JPL Publication 89-7, Vol. I

Proceedings of the NASA Conference on Space Telerobotics

Volume I

G. Rodriguez
H. Seraji
Editors

(NASA-CR-186856) PROCEEDINGS OF THE NASA CONFERENCE ON SPACE TELEROBOTICS, VOLUME 1 (JPL) 461 p CSCL 05H

N90-29000
--THRU--
N90-29043
Unclas
G3/54 0295935

January 31, 1989

NASA
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 89-7	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle PROCEEDINGS OF THE NASA CONFERENCE ON SPACE TELEROBOTICS (VOLUMES I-V)		5. Report Date January 31, 1989
		6. Performing Organization Code
7. Author(s) G. Rodriguez and H. Seraji (editors)		8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109		10. Work Unit No.
		11. Contract or Grant No. NAS7-918
		13. Type of Report and Period Covered JPL Publication
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

These proceedings contain papers presented at the NASA Conference on Space Telerobotics held in Pasadena, January 31 - February 2, 1989. The Conference was sponsored by the NASA Office of Aeronautics and Space Technology, together with ARC, LRC, GSFC, JSC, MSFC, KSC and JPL. The theme of the Conference was man-machine collaboration in space. The Conference provided a forum for researchers and engineers to exchange ideas on the research and development required for application of telerobotics technology to the space systems planned for the 1990s and beyond. The Conference: (i) provided a view of current NASA telerobotic research and development; (ii) stimulated technical exchange on man-machine systems, manipulator control, machine sensing, machine intelligence, concurrent computation, and system architectures; and (iii) identified important unsolved problems of current interest which can be dealt with by future research. There were about 500 international participants including about 100 from abroad.

17. Key Words (Selected by Author(s)) Engineering	18. Distribution Statement Unclassified; unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 2,386	22. Price

JPL Publication 89-7, Vol. I

Proceedings of the NASA Conference on Space Telerobotics

Volume I

G. Rodriguez
H. Seraji
Editors

January 31, 1989

NASA

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

This publication was prepared by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

ABSTRACT
NASA Conference on Space Telerobotics

These proceedings contain papers presented at the NASA Conference on Space Telerobotics held in Pasadena, January 31-February 2, 1989. The Conference was sponsored by the NASA Office of Aeronautics and Space Technology, together with ARC, LRC, GSFC, JSC, MSFC, KSC and JPL. The theme of the Conference was man-machine collaboration in space. The Conference provided a forum for researchers and engineers to exchange ideas on the research and development required for application of telerobotics technology to the space systems planned for the 1990s and beyond. The Conference: (i) provided a view of current NASA telerobotic research and development; (ii) stimulated technical exchange on man-machine systems, manipulator control, machine sensing, machine intelligence, concurrent computation, and system architectures; and (iii) identified important unsolved problems of current interest which can be dealt with by future research. There were about 500 international participants including about 100 from abroad.

An international program committee was established for the conference. A.K. Bejczy and H. Seraji of JPL acted as co-chairs for this committee. Members of the committee were

J. Amat, University of Barcelona, Spain
G.A. Bekey, University of Southern California
P.R. Belanger, McGill University, Canada
R.C. Bolles, Stanford Research Center
J.G. Bollinger, University of Wisconsin
W.J. Book, Georgia Institute of Technology
J.M. Brady, Oxford University, UK
F.E.C. Culick, California Institute of Technology
R.J.P. deFigueiredo, Rice University
W.R. Ferrell, University of Arizona
E. Freund, University of Dortmund, FRG
A.A. Goldenberg, University of Toronto, Canada
R. Jain, University of Michigan
T. Kanade, Carnegie-Mellon University
I. Kato, Waseda University, Japan
A.J. Koivo, Purdue University
P.D. Lawrence, University of British Columbia
J.Y.S. Luh, Clemson University
H.E. Rauch, Lockheed Palo Alto Research Lab
A. Rovetta, Polytechnic University of Milan
G.N. Saridis, Rensselaer Polytechnic Institute
T.B. Sheridan, Massachusetts Institute of Technology
L. Stark, University of California, Berkeley
D. Tesar, University of Texas at Austin
H. Van Brussel, Catholic University of Leuven
R.A. Volz, Texas Tech University

The Conference was organized by the Telerobotics Working Group of the NASA Office of Aeronautics and Space Technology. M. Montemerlo of NASA Headquarters and S.Z. Szirmay co-chair this working group. Representatives to this group from NASA centers and other research organizations are

D. Akin, Massachusetts Institute of Technology
J. Bull, Ames Research Center
R. Davis, Kennedy Space Center
S. Fisher, Ames Research Center
J. Haussler, Marshall Space Flight Center
A. Meintel, Langley Research Center
J. Pennington, Langley Research Center
D. Provost, Goddard Space Flight Center
C. Price, Johnson Space Center
L. Purves, Goddard Space Flight Center
C. Ruoff, Jet Propulsion Laboratory
E.C. Smith, Marshall Space Flight Center
M. Zweben, Ames Research Center

ACKNOWLEDGMENTS

Acknowledgments are due to R. Doshi, D. Diner, J. Barhen and A. Fijany of JPL and Professors L. Stark of UCB and H. Stephanou of George Mason University for their help in organizing invited technical sessions and discussion panels. Appreciation is due to P. McLane for setting up registration and conference facilities, and to L. Anderson for technical editing of the proceedings. Acknowledgments are also due Donna L. Milton of JPL for handling the local arrangements and coordination and administrative aspects of the Conference.

CONTENTS

Volume I

OPENING SESSION

REMARKS MADE AT THE BEGINNING OF THE NASA CONFERENCE ON SPACE TELEROBOTICS*

G. Varsi
Program Manager, Space Automation & Robotics Program
Jet Propulsion Laboratory
California Institute of Technology

Welcome to Pasadena and to the NASA Conference on Space Telerobotics. I am Giulio Varsi, Program Manager for the Space Automation and Robotics Program at the Jet Propulsion Laboratory. We at JPL are honored to be your hosts at this NASA Conference, and to open its first session. This Conference is similar to one successfully held two years ago. It has more than doubled in size since then and now enjoys the participation of a significant contingent of international participants. Over 10% of the papers are from other countries such as Germany, Japan, Italy, England, and France. We also have an enlarged participation from the NASA Centers which are active in NASA programs in space telerobotics. The establishment of programs in this technology is due to the foresight of the Office of Aeronautics and Space Technology at NASA. It started activities in this area about 10 years ago, and five years ago established a full-fledged program. We will be hearing more about that in the course of the opening session. This session consists of three speakers.

The conference has grown not only in size but also in a better understanding of the technology and a better understanding of its limitations. There is more maturity now, and understanding of the drawbacks. There is also the wisdom born of failure. Some of this is reflected in the structure of the Conference. In addition to the traditional technical sessions on specific topics, such as vision, control and manipulation, etc., there are specialized panels on some topical issues. We also have plenary sessions that are organized by NASA centers. Those are designed to give a perspective of the special areas of interest of the various centers. Finally, we have a panel at the end on Thursday where we are going to find out what has happened. If you are interested in knowing what the accomplishments of the conference are, you ought to stay for the final panel. I think the final panel should take to heart the three rubrics mentioned in M. Montemerlo's paper: the context, the vision, and the reality, as a way to guide thinking and to extract information from the technical sessions of the Conference.

* This is a summary of oral remarks made at the opening session.

CONFERENCE WELCOME*

Dr. Thomas E. Everhart
President, California Institute of Technology

It is a pleasure for me to be here this morning and to welcome all of you to this Conference on Space Telerobotics. I understand this is the second in a planned series. This new technology promises to change the way humanity will operate in space and extend our presence there in the next century. I think you are in a terrific field, because it is a field that is going to get more and more exciting as time goes on. I wish I were about 30 years younger and could sit in the audience throughout your conference.

I am also very pleased to see that the Campus and the Jet Propulsion Laboratory, which are both parts of Caltech, are working in this field and at the forefront of developments which NASA had the foresight to initiate and promote almost a decade ago. I am very pleased about the cooperation between JPL and the Campus. We have hired new faculty, and JPL has hired new people as well. They are cooperating, meeting once a week, talking with graduate students, getting courses and a new laboratory started which will benefit students a great deal. That interaction has been one of the successful aspects of JPL/Caltech Campus cooperation. As you may know, at JPL, a new directive has been established to integrate all technology activities within technology thrusts. Microelectronics and automation and robotics have been recognized as two of the main thrusts in this organization. Dr. Lew Allen, Director of JPL, and I have devoted substantial parts of our discretionary funds to pioneering work in autonomous rovers, computer vision, and intelligent machines. This has contributed to building a technical foundation for the ambitious efforts NASA has underway. We also have a Summer Undergraduate Research Fellowship program in which undergraduate students become involved in the research at JPL. I learned just a few weeks ago that when people at JPL have problems that they do not think they can solve, the word is: "Well, just leave it until summer and get some smart Caltech students up here. They will at least think of approaches that we have never thought of before." I think that is working out very well, from both an educational and a research point of view.

In a broader sense, planetary exploration by unmanned spacecraft has led the way in the development and implementation of intelligent automation (what some have termed "autonomous automation"). Intelligent automation could be thought of as the ability to accomplish assigned objectives or goals by devising an efficient course of action, utilizing available resources on the basis of the perceived environment, and reporting externally the status of accomplishment and system capabilities. This development at JPL has been driven by the special requirements of planetary exploration: communication delay, telemetry bandwidth restrictions, interruptions of the communication link, inaccessibility by humans, and complexity and time criticality of the science acquisition sequences. Thirty years ago this started a revolutionary approach to human exploration, when we dared to separate the human explorer from the machine by entrusting the latter with enough intelligent capability to make it able to perceive its surroundings and react appropriately. Progress is continuing and its development is shown by landmark advances in Mariner IV, launched in 1964; Voyager, launched in 1977; Galileo, to be launched this year; Mariner Mark II and CRAF, to be launched in 1994; and Mars Rover Sample Return in 1998. As examples, Mariner IV could achieve its required orientation by acquiring the Sun and Canopus, and it had some functional redundancy. Voyager, in 1977, had block redundancy and "tree-search" reprogrammability. This has been vital. As it nears a Neptune encounter this year, it has been totally reprogrammed from Earth, something that I found extremely impressive. Galileo will have a star-map scanner, autonomous self-calibration and

* This is a summary of oral remarks made at the opening session.

on-board attitude determination. The Mariner Mark II, planned for 1994, will have target body tracking and error detection and correction. This field is advancing rapidly. You will make advances which will enable us to go into the next century in a much better position than we are now.

One of the projects that we are working on at Campus, which will help that, is what we call CNS (Computation and Neural Systems). This new field will be discussed here at your Conference. We are trying to develop electronic analogs to the way neurons process information in the human nervous system (including the brain). This promises much faster processing of information in a more parallel way than in discrete and serial digital computers. Carver Mead, Christof Koch, and others are working on vision and hearing systems which will augment the way we can acquire information in space and allow the robots that work for us there to improve their capability.

I might make a couple of observations about scientific aspects. There is a growing sense that an international space policy, which encourages the exploration of the world outside of the earth, has given us a much better perspective over the last 20 years than we have had before. This policy eventually needs to be grounded in the economically justifiable uses of space. In view of the risks and costs of providing human presence in orbit, it is becoming mandatory that the tools be developed and the investment made so that humans have the ability to operate in orbit and perform evermore complex manipulations from the ground. Technology at this conference together with that of automated launch vehicles is a basis of this sort of capability. In the centuries to come, humanity will progress from the present of aerospace exploration and will go into the era of space utilization. Telerobotic technology will enable this transition to more efficient space and planetary operations.

You have a wonderful opportunity as you proceed into a new era. The work you do here will have extremely important implications for space. You will hear really forward-looking notions of trying to do things in space, where you cannot exert remote control directly. There will also be tremendous consequences for how we do things on Earth. Perhaps that is the reason why so many people are here, because it is obvious how much this can help how we work on Earth.

In closing, I would like to once again welcome you here. You are in the center of Pasadena. I hope you will progress a little farther south and east to the Caltech Campus and have a chance to stroll around there. I am sure you will have an opportunity to visit some of the laboratories of JPL. Welcome and have a good Conference.

Dr. Thomas E. Everhart is the President of Caltech and also Professor of Electrical Engineering. Dr. Everhart is a Fellow of the Institute of Electrical and Electronics Engineers, and he is a winner of the IEEE Centennial Medal for 1984. He is also a Fellow of the American Association for the Advancement of Science and a member of the National Academy of Engineering. His fame, however, is built not only on his academic achievements as a leader in education, but also as a scientist and engineer. As a Professor and Department Chair at Berkeley, he pioneered a number of applications of electronbeam technology (such as electronbeam microscopy and lithography) to the study of microstructures. Subsequently, as Dean of Engineering at Cornell, he established the Center for the Study of Microstructures. Before joining Caltech as its President, he was Chancellor of the University of Illinois.

EVOLVING SPACE TELEOPERATION TO SPACE TELEROBOTICS: RESEARCH AND SYSTEMS CONSIDERATIONS*

M. Montemerlo
NASA Headquarters, Office of Automation and Space Technology

I am not going to tell you about the OAST Program, and I am not going to tell you what wonderful things it contains. What I would like to do instead is to give you just a little perspective or way of looking at what you are going to see. In the next three days, you are going to see many papers and listen to many talks. I believe there are about 250 presentations that will be given. When you look at this wealth of material, you need some sort of perspective or context within which to grasp as much as possible. I will try in this talk to provide some thoughts on how you might want to look at what you are going to see and then maybe how you might want to help us in terms of where to go from here.

To establish this perspective, I want to mention three words: "context," "vision," and "reality." The context. What is the context of where NASA is coming from and going to? When you see all of the papers which talk about particular end-effectors and control laws and AI techniques, you have to think about them in terms of where they are coming from in NASA and where they are going. That is context. The second is vision. With regard to vision, you are going to see a number of component parts. What is your vision as to how all those would come together? Without some vision of where we are going, it is hard to determine what to do now. The third word is reality. When you look at where you think you might want to go or where we might want to go, how would you do a "reality" check on that? How can you tell if we can go as far as we would like to go, and if we can achieve in a reasonable amount of time the vision that you have put together? I debated slightly whether to mention the 25-million-dollars-a-year number (where our program is right now); then I thought, yes, that provides part of the perspective in checking reality.

In 1984-85, when Congress was really encouraging NASA on A & R, a group was formed called the Automation and Robotics Panel. This panel made a number of recommendations. One was to develop a new research and development program in automation. NASA would need to spend about 100 to 190 million dollars on research. That never came about. We have significantly greater resources than we had at the time. We have quadrupled and more, from about 4 million to about 25 million for research. But that is still limited. So, let us charge on: let us get a vision of where we would like to go, realize the context we are in, and then perform a reality check on what can we really do, so we do not over-sell and then get the baby thrown out with the bath water.

First of all, I would like to congratulate three people and their team: H. Seraji, A. Bejczy and G. Rodriguez. They have put together a phenomenal conference here for the next three days. You can tell just by looking around the room; seeing who is here and what they are talking about.

We had Giulio Varsi chair the session. My last name is Montemerlo. Here is a third Italian: Machiavelli. To add a little more context, those of us who want to push toward higher degrees of automation ought to remind ourselves of this thought from a great man:

> "It must be considered that there is nothing more difficult to carry out, nor more doubtful of success, nor more dangerous to handle, than to initiate a new order of things. For a reformer has enemies of all those who would profit by the old order and only lukewarm defenders of all those who would profit from the new order." -- Machiavelli

* This is a summary of oral remarks made at the opening session.

And that is what we are trying to do -- to establish a new order.

Part of my job is to listen to a lot of proposals, a lot of ideas on where we are going, and a lot of visions. I am supposed, from all those, to pick out what was meant and what was not meant. In other words, I have to determine what the reality is. So, in helping form a vision, I picked from the files a number of viewgraphs of visions that people have had. They are mostly cartoons. However, in them there is a grain of truth and reality, and there is a grain of a few other things. Glad to see that everybody is up and has a sense of humor ....

1) This one came in 1977. You will notice that most of the pictures you are going to see are anthropomorphic in a way. You have a couple of arms and a couple of eyes. This is one that is less anthropomorphic. This looks more like a snow-mobile, but it had some interesting ideas. For instance, there is a mobile remote manipulator system going along what might be a space station. That was quite advanced for 1977.

2) This was something that was demanded that we put together back in 1985. We were told to develop the technology for multiple interactive intelligent learning robots in space by the year 2000. At that time, we were thinking of $190 million a year.

3) Then we were told that those last two look too much like humans. We were told to put together a picture of a robot which does not look like a human. So we called JPL, and they had their artist put this one together. When I showed this to my supervisors, they said,"That looks less like a human, but I cannot see the robot anywhere."

4) Here is another possibility. These are two mobile remote manipulator systems working in tandem.

5) This is probably the most used picture in the history of NASA for telerobotics. It is from Martin Marietta, about 1982.

6) Back in 1985, we had a lot of proposals for multiple interactive intelligent robots which moved around space stations.

7) This was a proposal for an in-bay, free-floating telerobot experiment.

8) Some of the things that we might want telerobots to do: fly high, clean windows, inspection, servicing, realign mirrors, and change fluid couplers.

9) You will see. Try and find the robots as we go through. This one is sort of looking down (two arms pointing down).

10) This one is looking up. Again, there are two arms, two eyes and the middle attached to something with another arm.

11) This one has evolved quite a bit. You will see it has multifingered hands.

Okay, I did not want to go through those in great detail. But remember I said context, vision and reality. Try and keep all those in mind as we go through, because we need to divulge strong visions of where we want to go. However, we need to have some measure of reality as well.

Now to place a little bit of context. Where is NASA going? The way I see it, NASA has three major goals: (1) to monitor changes in the Earth's environment, (2) to establish permanent presence of man in space with the space station, and (3) to explore the solar system.

Automation will play a big role in that, or these goals will not be met very well. But then again, automation is not new at NASA. It has been a NASA goal since the start of the space program to extend human capability in space and to free humans to do what they do best. Here is what Congress wanted us to do with the new automation. They wanted to develop a new generation of automation which is qualitatively different from the old one. But remember, we need a realistic development path from where we are to where we want to go.

Traditional automation (which is wonderful, as without it we could not have gotten here) is preprogrammed and inflexible. Where we would like to go for the next generation of automation is more supervisory control. Prof. T. Sheridan of MIT, who is here, is, I believe, the father of the phrase "supervisory control." We would like to get to supervisory control where we tell a machine what to do, but not how. This is where we would like to go. And we need a realistic path to get there. We need to get an automation capability which is more adaptive to the environment, which can decompose higher level commands. We would like to give the machine subtask level and task level commands. Then it can decompose these commands down into primitive actions it can do. The machine needs to be able to plan these primitive actions and then to replan when problems occur. It needs to know, once in a while, how to report back to the supervisor, "Hey boss, I have a problem here that I cannot solve" or "I have a problem that I can solve, but I thought I ought to let you know about it." It ought to be able to degrade gracefully to teleoperation so that we can get a human to fix it up.

The context of humans in automation is that they are flip sides of the same coin. They are not in competition. We need the proper mix, and we need to see what each can do best and how to work together. Humans are good at creativity, high manual dexterity, perceptual skills, setting goals and values, and complex decision-making. Now, we are beginning to get automation to make strides in these areas, but for a while humans are going to do those a lot better than machines. There are many things that machines do a lot better than humans: precision, repeatability, handling large quantities of data, and working in hazardous environments. Those are the general things we have got to work together on. Now we have looked at automation. Let us look at telerobotics and how that fits in.

Why telerobotics at NASA? Some people might take the hard line and say, "It is to replace EVA." I do not think so. It is to enhance space operations and ground operations. In space operations is an alternative to EVA (sometimes). It is a co-worker with EVA (again, sometimes) to permit space operations when EVA is not possible - polar and geosynchronous orbits. It is also to enhance ground operations. Most of the talk today will focus on space operations. Generally, these operations consist of assembly, disassembly, rendezvous and docking, resupply, changeout, calibration and checkout, reboost and redeploy. That covers it. When we come up with a new system, we need to compare it to a current system to be able to say that it is better in some ways than this current system. We need to be able to explain where it is better, where it is now, what the costs are to do it. The current way of doing space operations is shuttle-based. It is EVA (extravehicular activity). We use a lot of acronyms. "RMS" stands for remote manipulator system. We do some servicing IVA (intravehicular activity), but that is the baseline for right now. It is not just that, because we put the astronaut on the end of the remote manipulator system. So, in a way, the human is just part of a robotic system right now. It depends what you consider a system. But we often consider the human on the end of the RMS as the intelligent end-effector.

Now, this is an intriguing time at NASA for robotics because a lot of these pictures that I showed you, the cartoons, came before we had real plans. I tried to advocate just a teleoperation program at NASA in 1982. I got hit with a lot of statements like, "Why bother, there is no project that is saying that we need it. There is no one screaming for this technology." Well, we got some activities going, but now we have the FTS and the OMV (this is exciting) - the Orbital Maneuvering Vehicle. A servicer kit will be based on FTS. We have the SSS, the Satellite Servicer System, which is still a dream, still a hope. It is a proposal from the Johnson Space Center (Code M). The servicer kit on that will be based on FTS, but a high degree of automation is desired, if not full automation. This is to work with the SDI and the Air Force and others who do not want to communicate much so others can listen in. Here are the FTS, the Flight Telerobotic Servicer on station; the SPDM, which is the Canadian version of the FTS; the Japanese robot arm on space station; the MRMS, the Mobile Remote Manipulator System on space station, which will be built by Canada. There are many possibilities here.

Definitions. I think some of these are becoming more accepted now. The word "telerobot" means a lot of different things. Let me just try something. To me, the generic term is a "remote manipulator." A remote manipulator is an electromechanical arm and gripper. Teleoperators, robots, and telerobots, to me, are just three types of remote manipulators. A teleoperator is a remote manipulator that is continuously controlled by a human some distance away (in terms of space). A robot is a remote manipulator that is controlled by a human some time earlier. This is a time distance. Both are controlled by humans, but they are distant in different ways - time and space. A telerobot is a remote manipulator as well. It is one that is operable both as a teleoperator and a robot, depending on what is called for at a given time. It can smoothly transition back and forth. Though all three require human control, the only difference is when and how.

You have all seen this. Just as an introduction, teleoperation is more appropriate for tasks which we do infrequently, because if you did them a lot, you would learn an automatic way to do them. Environments that are not well defined cause robots to have a hard time. If an environment is not controllable, and the task requires dexterity and problem-solving, we cannot achieve the task with robots now. There are times when one might do a thing with the robot, but because of the cost or value of logic that is being worked on, this may not be allowed.

The advantages of the robot over teleoperators are to save the cost of a human operator, bypass communications time delay, and increase precision, power and repeatability. The robot does not become bored, distracted or tired.

Now, this is a vision that I helped a group put together. D.D. Myers, our Deputy Administrator, asked the various codes at NASA Headquarters to give him a briefing on where servicing is going. One of the most difficult things we had to do was to get a group of people together and come up with one viewgraph which said, "When do we project we could do what?" That was an interesting event. I wish I had videotaped it. What came out was EVA Servicing in 1980 and Teleoperation in 1995. Now, 1995 means that if we took everything we know now and flew it, without any new technology, we might get it up by 1995. So that is taking today's technology. When can we evolve from teleoperation to something with increased local autonomy? Telerobotics. Well, in the next generation after what we put up in '95. We cannot change these things every year, so maybe 2000. What about autonomous things? Well, that is farther out than many people would hope, we thought. So we put the number at 2000+, 2010+. Now, you may argue with those numbers, and I hope you do. This is one vision put together by a group of bureaucrats, so you can do better. We would like your help in doing better; we would like some feedback. What is reasonable? The plus signs between EVA and Teleoperation mean that when we go to teleoperation we do not throw out EVA. When we go to telerobotics we do not throw out EVA in teleoperation. It is the same when we go to higher degrees with autonomy. That is an important point. The reason we put this viewgraph together was there is another dimension to that problem of evolution, which is that EVA and teleoperation and telerobotics and autonomy are not points. They evolve as well, so EVA means new technologies as it is being developed. Teleoperation can improve and has improved - look at the history of it. We can do better and more. We can not only increase performance by getting more capabilities. Each of these brands of capability can be improved and will be improved.

Now, talk about alternatives: there really is a wide variety of alternatives. Servicing. You want to service. I am not sure of everything the Soviet Union does, but I understand that they put up many satellites. They may not last long, but they put a lot of them up. We may not want to service all the time. We may want to build and throw away sometimes, although not all of the time. We may want to build in redundancy so that when a part fails that we expected to fail, there is already a replacement there, and it is hooked up so that it can be switched over. Now, you can do that to a point where it gets too heavy to fly, but the idea with all of this is: it is like pepperoni pizza: you have to have just so much of it. You have to take it in context. It is not black and white, it is all shades of gray. So, when do you determine how much to use of each of these alternatives?

Now, we discuss servicing. There is still a spectrum of servicing we can do with teleoperation, robotics and EVA. One is a low level of servicing complexity which assures that the satellite is designed to be serviced up there, and nothing goes wrong. There are no bolts which are stuck and nothing is off-nominal. Everything is like the CAD/CAM system. Well, that is simple. It has not happened yet. A high level of complexity is where you have a non-cooperative satellite like Solar Max. It was not designed to be serviced and things happened that were not expected. EVA can handle a lot of that, a lot more than you could with the teleoperator and more than you could handle with the robot. However, keep in mind that we do not have to service everything all the time. What we need to figure out is what teleoperation, telerobotics, and robotics can service and when. In other words, we need to determine the costs and risks that are relative to our present ways of doing them. The dilemma is that we need to assume that on-orbit servicing tasks will encounter problems, as we have faced in the past, and be prepared to cope with them. Look at all the past servicing that has been done EVA, and remember one fact - in EVA we fixed a number of satellites and we brought a number back. Here is a fact. Every time we developed either a tool or a jig based on a CAD/CAM database, to fix one of those satellites or hold it down, it did not work because the CAD/CAM database was wrong, every time, 100% of the time. If that is the case, we probably can expect some of that in the future.

The other part of the horn of the dilemma is you can hope that that is not the case. The trade-off is: we can and should advocate more autonomy, in which case we push robots; or we can push (and should push) for more versatility, which means improving teleoperation capability. But what we would like to do, and we just do not know how, is to work up the middle and have something which trades the two and which allows us to go back and forth with the same machine. If there is one fact in this business, it is that we are not going to have 50, 60, 100 robots and teleoperators so that we can figure which one we need and when. We are going to have a few, and so we need to have things which are more versatile. There is the crux, the interesting issue. How do we envision something? How do we put together a system which goes more up the middle and gives us more autonomy and more versatility at the same time?

Now, one of the big problems we have had is a definition of the state of the art. You tell me the 20 things that we do not know and then develop a 5-year plan to fill in those boxes. I have not been able to do that. That is very difficult in this area of space servicing for lots of reasons. I wrote a few of them down because I think some of you have run across the same problems because I have read a number of your papers. The difficulty in defining the state of the art is that we have no experience with small light-arm control in space. There are very few teleoperation or robot experts at NASA. We are growing them. We are getting more. A.K. Bejczy is a pioneer. We have very little teleoperation experience in a neutral buoyancy facility to tell us more about lack-of-gravity and systems problems. The problem is that each subsystem of a potential telerobot is evolving. Another real problem is that we have no stated requirements for teleoperator or telerobotic servicing for which we can design teleoperators/telerobots. There is some competition among the Centers. There is a lot of overselling. Many are overbought, and there is the problem of politics, which keeps us from saying everything we know. But those have made it difficult to state the 15 or 20 things that we need to develop in the research program.

This has made it a character-building opportunity to develop a program like this. What is the state of the art for servicing for telerobotics? Well, on Earth, robots are not used for servicing, as far as I know. Teleoperators are used for servicing in nuclear and subsea oil industries. In orbit, we have no robots. We have a teleoperator, the remote manipulator system, which can be used as a platform for an EVA astronaut. Now, there is something else on the bottom. An engineering test unit of a simple space robot for servicing specially designed satellites was delivered to MSFC in 1978. That is the In-Orbit Servicing System (IOSS). If you have a specially designed satellite, the IOSS will remove and replace modules and has been sitting there doing it since 1978. So, that is part of the state of the art. That idea has never caught on.

Here it is in one viewgraph - the goal of telerobotics - the unification of teleoperation and robotics such that telerobotics is more capable than either robotics or teleoperation because the same machine will have to be able to operate as both. Now, that machine is going to be more complex and is going to cost more; we have to do it in such a way that the increased complexity is offset by increased capability, versatility, robustness and safety. If we do not come up with something which is robust, safe and reliable, we will have a hard time getting and keeping it up there. There are a lot of paradigms for telerobots. You may like all of them. We have machines monitoring humans and humans monitoring machines. We have the power-steering approach, simultaneous or shared control between humans and machines. You have traded control, in which the controls pass back and forth between humans and machines. You have hybrid control, both traded and shared; and probably the neatest is all of those plus human supervisory control at modifiable levels of depth. That is what my boss does to me. Sometimes he says go forth and do good things, and sometimes he comes through and says this is how you shall do it. And I think we would like to treat machines like we treat humans, which is, every once in a while give it a big job and let it do it and have it report back if it is having problems. Every once in a while you like to get into the nitty-gritty. You treat people like that; you would probably like to treat machines like that.

Now, in selecting where we would like to go with this paradigm for telerobotics, we can think about it. That is what analysis is: systems analysis. Systems are things and analysis is thinking. So, if systems analysis is thinking about things, analysis is not enough. If we think about it a lot, we try and integrate everything, and early attempts to integrate everything will probably weight us down and kill us. So, how do we figure out where we ought to go? First of all, do not ever make viewgraphs with this much information on them. A big point is to focus on a system for space operation, a system, an overall system, not just an end-effector or a control law, not just a teleoperator or telerobot, but a system within which that is going to have to work. That is a big system. Focus on a system. Think about what it might look like. Think about lots of things -- that is systems analysis. Think about the experience of nuclear power and subsea operations. Consider NASA's experience in space operations and the problems with CAD/CAM databases. Think about the need for capability, versatility and reliability. Think about the state of the art of your true remote manipulator on the ground. Think about how you test systems like that in neutral buoyancy.

Once you have thought about those things, we have got to come up with a useful set of tasks. That has been a bear - getting various Centers to come up with a set of tasks which we can use as criterion tasks. Determine how well we can do those now. Have an anchor. If we are going to come up with this vision, determine what is reasonable to expect in five years. Five years is not a long way off. Stop to think. You know where we were five years ago and how much technology we developed from 1984 to 1989. Think about that increment from now to five years hence. Here is a picture that came out of Aviation Week of the system proposed by Canada for space station. It is an old picture. I do not know if it is up to date or not, but, as a system, it has a couple of interesting thoughts. You will notice that it has a long arm. The small arm is at the end and can be controlled from inside. There is also a human in the upper left-hand corner. I am not sure if that human is in an enclosed environment or not, but that is a different system. The system that you would use to have a person do that would be vastly different from the system for a person working from the inside. You have got to get that person out there, and it looks like it is too small a tube to crawl through. Think of the all-over system considerations to make one of these work. Think of the trade-offs.

For example, it is a lot easier to work with what you can see than to work through a video system. Even if it has stereo. Here you would have direct vision, but you have an extra problem. You have not figured out what those trades are. That is a difficult one. We do not have a good database for that. So one of the lessons for this next couple of days is, "Keep the mind wide open." Do not narrow down yet. Think of wide ideas. Now, the process, here it is -- 0, 1, 2, 3 -- the process for developing space telerobots. First, think of alternative visions. Think of how you would put them all together. We have to develop the component technologies. You will hear a lot about that. That is where we stand. That is the guts of this - this is where the rubber meets the road. We have to have the components! We have to integrate those components in laboratories. We have to also evaluate those integrated components in laboratories. And then a big one: we have to take those integrated systems and think about how they would fit in space.

Here is a teleoperator controller. That is all you see. It does not look like much. What is hidden is all those people I just mentioned. When we develop a teleoperator to do robotic tasks, we have to consider ourselves the hidden cost of doing that and compare it to EVA and teleoperation. When we go to robotics, where we have gotten rid of the astronaut or the teleoperator controller, we have introduced a big piece at the bottom which is software development. We have to be thinking of the whole system to see what the costs and risks are.

I am an advocate. I want to point these things out now so that we can take care of them ahead of time. Let me give you a quick fact. These are not real numbers. It is hard to find real, recorded numbers that everybody agrees to (these are informal). I was in Canada and I asked about the SPDM. Here is the question: Once your SPDM is up in space in 1996 or 7 (it is up there, it has been validated, it has been checked out, and is operational), when you are operational, you do not need those who developed it. About how many people on the ground are going to be required to develop software and end-effectors, to do specialized hardware, and to prepare and validate your scenarios? They did not have a number prepared, but in informal conversations a number came up, and (do not hold anybody to this) it was nice of them to say, "Maybe 140." I asked the same question of the FTS people. Do not hold anybody to this, since it is an informal answer not written down anywhere. The answer was: "Maybe about 100 people." If that is the case, we may have in the neighborhood of 240 people taking care of two teleoperators on space station. That is a large hidden cost, not hidden to the people who are involved in figuring out the overall system, but it is hidden to the bureaucrats who later might be disappointed. So, I think we have to make clear to ourselves and to everyone else ahead of time what we need in terms of an overall system. For example, maybe some of our AI effort ought to be aimed at helping reduce the number of people on the ground. That is something I do not think has been thought enough about. Where can we use AI besides as task planners and planners of movement?

The bottom line - this is the last viewgraph. Remember my three words were <u>context</u>, <u>vision</u>, and <u>reality</u>. As you listen to the papers over the next three days, take a systems view as well as a component view. See how it is all going to fit together in your mind; realizing we do not have all the answers. You probably have more. Think of the question of setting an anchor. What is the anchor? The anchor is: what could we do in space with no new technology? Think of everything you are going to hear. What can we fly right now? That is actually the FTS question. Now when you form a vision of what is likely in 1995, (remember, it is no new technology) it is essentially what we can do now. Remember some points:

- Remember the history of teleoperation. This is a part of the "reality" of robotics and AI. Remember how they developed and the pace of development. Now figure ahead with your visions for the years 1995 and 2000.

- Remember that requirements follow capability; nobody who owns a satellite or plans to build a satellite is going to require servicing by robots if the robots are not there to service them. So, we have to push.

- Remember that if someone says we can do this, say to yourself, "Well, if we can do it, we will do it on Earth before we do it in space." Is it being used on Earth anywhere? That is not always true, but it is heuristic.

- There is always a wisdom lag in technology between when technology is available and when it is used. People have to get comfortable with this, and project managers tend to be very conservative people.

- Remember what I said about the errors in the CAD/CAM databases. We are going to have to have robust systems.

- Remember that increasing automation may not reduce the number of people, but it may decrease the overall cost of the program; it may be best overall.

- Remember the servicing infrastructure.

Now, given all that, think about what you hear. Be creative and aggressive in developing some alternatives, especially in how you integrate all these things to achieve supervisory control in telerobotics. And think about how we evaluate these systems and make sure we know their costs, risks, and advantages. Think about how we can develop a good technology base for these systems.

Dr. Montemerlo is the Manager of the Automation and Robotics Program at NASA in the Office of Aeronautics and Space Technology. He is also the principal architect of the program as we see it now within NASA. More importantly he is the staunch and tireless advocate for it, which is an essential function. He has been in this position since the beginning of the program in 1984. Since then, as a result of his efforts, the program has grown by a factor of 5 or 6 from about 4 million dollars a year to the current level of about 25 million dollars a year, about equally apportioned between the cognitive tasks and the manipulative tasks, which are the subject of this conference. More recently, Dr. Montemerlo has tried to extract out of this program information and data to use as a base for users of this technology, in future decades and perhaps in the next century.

SPACE TELEROBOTICS CONFERENCE OBJECTIVES*

Dr. A.K. Bejczy
Jet Propulsion Laboratory
California Institute of Technology

I would like to make a very short program introduction. We have three major objectives. Our first major objective is to congregate the NASA centers at this conference to gain better insight into the robotics and teleoperation research and development. This is based on our conviction that the research and development work has to be done at different centers from different perspectives and viewpoints. We will have sessions each day on work done at the different NASA centers. And we are very grateful to all the NASA participants.

The next major objective is to congregate people who are interested in the field of space telerobotics as a new technology. We need researchers, developers, interested individuals, and interested groups. We need researchers and engineers from industry. Space is an international area. We are very grateful for the many international participants who came to this conference from Asia and Europe.

The third major objective is to look into the future. We would like to help NASA, as was pointed out by Dr. Montemerlo, by looking into something which is ahead of us. So, at the end of the conference, during the final panel discussion, we would like to identify problems which may have to be solved or which will have to be tackled.

One of the major challenges in space telerobotics is the very nature of the work. It is a multidisciplinary activity. There are three major areas: the actual hardware doing the action, the sensing which senses the consequences of the action, and all the local intelligence, which coordinates action with sensing with perception. There is also the intelligent human interface. What is an interface to a telerobot? This interface is a tool in the hands of someone, a person, who is handling a remote tool, a remote robot. The question is how to build these intelligent tools, which are intelligent in the hands of the operator and intelligent enough to be interfaced to the remote robot. There is also the knowledge, the database, what we usually call CAD/CAM. But knowledge is more than that. It is also represented by running human brains, running even during operation. This is the kind of perspective which was indicated by Dr. Montemerlo.

How many people are needed to operate the intelligent space telerobot system of the future? This question is expressed in our basic thrust of the conference. We talk about remote- or robot-site and local- or operator-site automation. And, we will ask the basic question: "What is the interaction of human and machine intelligence, not only during the development but during the operation?" That is the depth of the supervisory control concept.

All these questions are motivated by target NASA missions: the space transportation system, the great observatories like the Hubble space telescope and the gamma ray observatory, the space station project, the free-flying polar orbiting platform, Mars exploration, etc. These are the major NASA target missions. These are referred to as task drivers. Here we list the major tasks: in-space assembly, material processing, materials handling, system operations, satellite servicing and explorations.

Based upon its technical content, I believe we have a strong conference program in many topics. We are listing a number of them that address the needs of the hardware, the needs of sensing, and

* This is a summary of oral remarks made at the opening session.

the needs of computing and system architecture. Everyone will have a fair chance to present results and ideas in cooperation.

Cooperation is a major thrust of this conference: cooperation between NASA centers, cooperation between individuals at different academic institutions, and cooperation between individuals in different countries.

Dr. Bejczy is a pioneer in teleoperation technology and its applications to space. He is a Fellow of the IEEE, and past president in 1987 of the IEEE Council on Robotics and Automation. During his tenure, this council became a full-fledged technical society within IEEE.

REDUNDANT MANIPULATORS 1

N90-29001

A 17 Degree of Freedom Anthropomorphic Manipulator

Håvard I. Vold, James P. Karlen, Jack M. Thompson, James D. Farrell, Paul H. Eismann
Robotics Research Corporation
5400 DuPont Circle, TechneCenter
Milford, Ohio 45150

Abstract

A 17 axis anthropomorphic manipulator, providing coordinated control of two seven degree of freedom arms mounted on a three degree of freedom torso-waist assembly, is presented. This massively redundant telerobot, designated the Robotics Research K/B-2017 Dexterous Manipulator, employs a modular mechanism design with joint-mounted actuators based on brushless motors and harmonic drive gear reducers. Direct joint torque control at the servo level causes these high-output joint drives to behave like direct-drive actuators, facilitating the implementation of an effective impedance control scheme. The redundant, but conservative motion control system models the manipulator as a spring-loaded linkage with viscous damping and rotary inertia at each joint. This approach allows for real time, sensor-driven control of manipulator pose using a hierarchy of competing rules, or objective functions, to avoid unplanned collisions with objects in the workplace, to produce energy-efficient, graceful motion, to increase leverage, to control effective "impedance" at the tool or to "favor" overloaded joints.

1.0 MODULAR SYSTEM CONCEPT

Since forming the company in 1983, we and our colleagues at Robotics Research Corporation have focused our efforts on the design and manufacturing of high-performance modular manipulators and motion controllers for advanced applications in the industrial, space and defense sectors. Our goal is to offer a configurable and open architecture system of mechanical, electronic and software modules that can readily be adapted to suit specific user requirements. The company's commercial line of hardware and software modules is now reasonably extensive, permitting the assembly of a number of novel and promising system configurations.

2.0 SERVOMECHANISM DESIGN

Family of Joint Drive Modules Our current line of robotic servomechanisms, the K-Series and B-Series Dexterous Manipulators, are all assembled from a family of unitized joint drive modules[1]. Each module in this family includes a complete joint actuator and structural system for one degree of freedom. In existing units, individual joint drive modules do not contain the signal conditioning, control and servo power electronics. These components are housed in a control cabinet and connected to the modules with a highly flexible internal wiring harness. Each joint module is designed around a particular size (and thereby torque capacity) harmonic drive reducer and incorporates an electric servomotor with appropriate characteristics. Modules containing identical actuator elements are built in two forms, to serve either as "roll" or "pitch"-type joints. Roll modules effect rotary motions about the axis of the module interface flanges (typically +/-180^o or +/-360^o), while pitch modules effect rotary motions about an axis perpendicular to the normal vector of the attachment flanges (+/-180^o). Modules are joined to each other in manipulator assemblies by quick-disconnect band clamps, secured by a single tangent bolt. An extensive family of different joint module sizes is now in production, permitting the assembly of a wide range of manipulator scales and kinematic configurations, including systems with as few as three and more than 17 degrees of freedom (DOF).

At present, roll and pitch-type modules are available in seven increments of peak torque capacity-- 17,000 lb-in, 8,000 lb-in, 4,500 lb-in, 2,500 lb-in, 1,400 lb-in, 600 lb-in and 150 lb-in, as illustrated in Figure 1. (We will be expanding the family of joint modules in future to include both larger and smaller modules. Joints having as much as 100,000 lb-in peak output torque could be built using Robotics Research's actuator design.) While a variety of three-to-six axis devices might be constructed from the existing module set, only kinematically-redundant units, i.e., ones having seven or more joints, have been manufactured to date.

Manipulator Topologies Three 7 degree of freedom manipulator arm models that have been configured from this set of joint modules are shown in accompanying photographs (Figure 2, below). The K-2107HR is a seven foot long, seven axis manipulator for applications which require a light-weight tool or sensor to be conveyed about a large working envelope with dexterity and speed, and with high repeatability. Operating in a stable temperature state, the K-2107HR has a measuring repeatability at the toolpoint of 5/10,000ths of an inch. The K-1607HP is a five foot long, seven axis unit with a 50 lb. payload configured for general-purpose factory and laboratory use. The K/B-1207 unit is a light-weight (160 lbs.), four foot long, seven-axis arm utilizing brushless motors to achieve a very high payload-to-arm weight ratio.

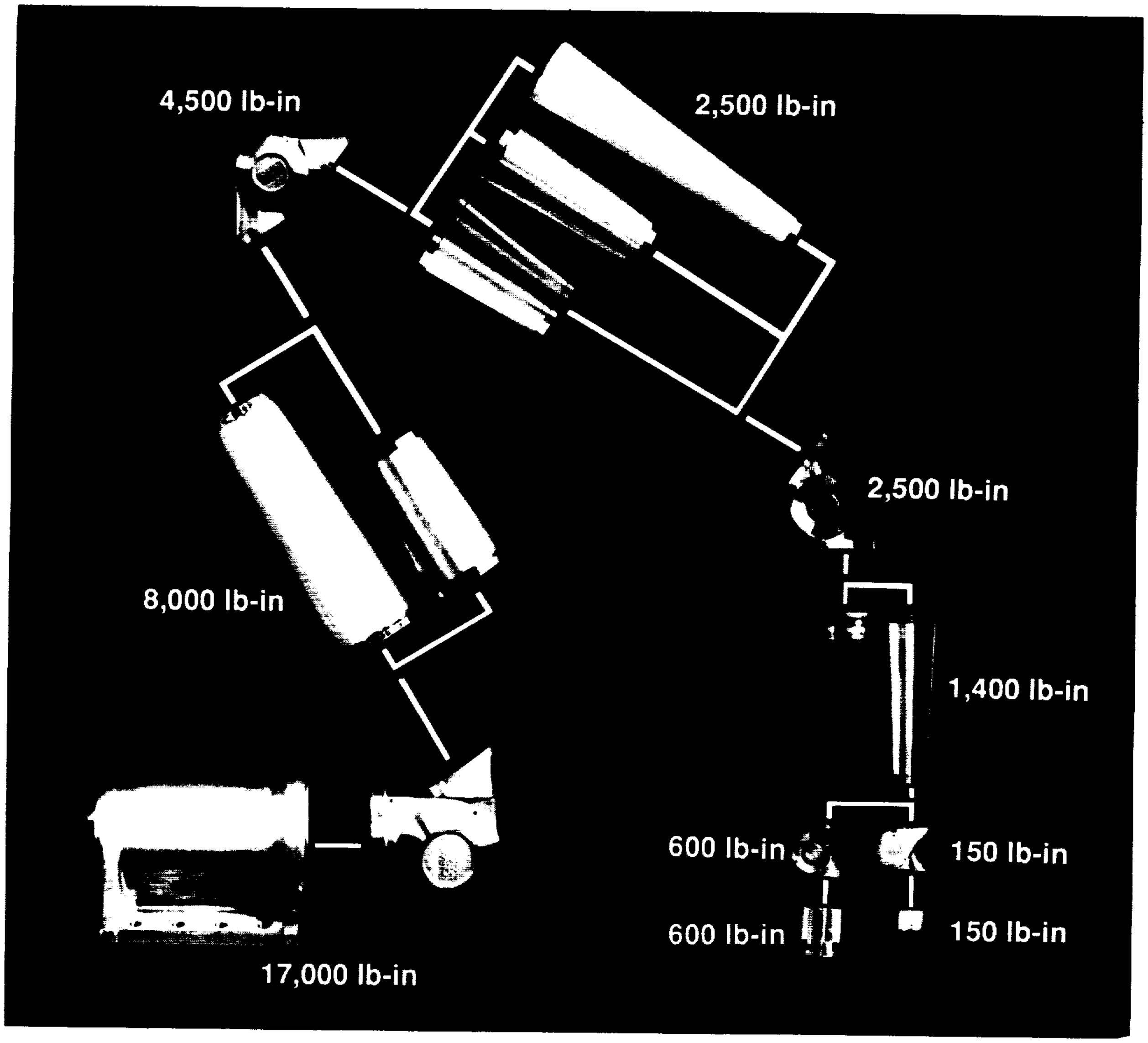


Figure 1:
Family of Joint Modules Currently in Production

In addition to those described above, a variety of alternative serial-chain configurations incorporating more than seven axes could be assembled using our existing joint modules. One example is a nine-axis manipulator arm, created by adding the K/B-1207 wrist roll, wrist pitch and toolplate roll modules on to the first six joints of a K-1607HP arm. An articulated, "snake-like" configuration of this sort might be of use in certain inspection tasks in confined work sites, as in nuclear reactor servicing, where a camera or other light-weight sensor package must be inserted into a narrow space.

Potential manipulator configurations need not be simple serial chains. Indeed, certain branching topologies offer important possibilities. These include the 17-axis configuration that is the subject of this report, and other arrangements, such as those having three or four manipulator arms branching from a common torso-waist link. Such configurations could be constructed and controlled in a similar fashion.

Our 17 DOF model, designated the K/B-2017, has two 7 DOF manipulator arms mounted on and operating in concert with a 3 DOF torso-waist assembly (Figure 3). A natural extension of our modular family, the unit was assembled by affixing two standard K/B-1207 arms onto the end of the first three joints of a standard K-1607.

A branching, 17 degree of freedom kinematic configuration offers several fundamental advantages over a pair of 7-axis manipulator arms of equivalent reach and payload operating from a fixed base. In addition to incorporating more redundancy, which may be used by the sensor-driven controller to mitigate singularities, to avoid obstacles in the workplace and to manage manipulability factors, of particular note is the fact that short tool-handling arms are, by nature, more dexterous and efficient, and less obtrusive and dangerous, than long arms. The 17 DOF configuration employed in this device seems, to the authors, a good compromise. The torso-waist link provides a long overall reach and a large working envelope, while preserving all of the advantages of relatively short, responsive, maneuverable arms for manipulating tools.

Actuator Design The generic actuator design utilized in all K-Series and B-Series modules, including those which comprise the K/B-2017 model, consists of a harmonic drive and a high performance brush-type or brushless samarium-cobalt DC servomotor located on the joint axis, directly coupled to the proximal and distal castings of the module. The flexspline of the harmonic drive is connected to the structure through a metal-to-metal overload clutch and torque transducer. The clutch is generally adjusted to slip at a torque greater than required for full machine performance, but less than would damage the drive or other manipulator components. An independent, high precision instrument gear system, mounted directly on the joint housing, drives a brushless resolver which is utilized to provide joint position and non-quanitized velocity feedback by means of an advanced R-to-D chip. By this arrangement, the servo system has: applied actuator torque, joint velocity and joint position feedback available to control axis behavior. Any or all variables may be commanded or electronically limited by the servo-control system.

Integral Structural System The K-Series module system utilizes an exoskeleton structural approach. The exoskeleton structure provides favorable structural dynamics of the overall manipulator, with low overall weight. It also provides a strong, durable and clean exterior, enclosing all wiring and actuator componentry.

3.0 TORQUE LOOP SERVO-CONTROL SYSTEM

The harmonic drive is unrivaled for compact, light, backlash-free torque multiplication, but its application as a mechanism for directly driving joints in high-performance spatial manipulators is complicated by two factors intrinsic to the device. Besides being relatively compliant, it exhibits a two-per-input-revolution transmission error which excites the inevitable resonance resulting from the inherent reducer compliance. The resonance phenomenon has prevented the widespread application of this otherwise attractive actuator package in robots. K-Series manipulators utilize a servo-control approach which overcomes this problem and has important attributes as regards manipulator control and performance. The conventional approach in robot arms is to control velocity and position of the motor shaft, while assuming that the transmission elements are nearly ideal in their translation of shaft motion into joint motion. The approach taken by two of the authors (Thompson and Eismann) in K-Series servo drives is to treat the motor and harmonic drive as a torque producer[1]. The control feedback parameters are all measured at the interface between proximal and distal joint elements. The joint position commands joint velocity, which commands applied actuator torque. The innermost loop is thus a *torque loop*, capable of bandwidth which

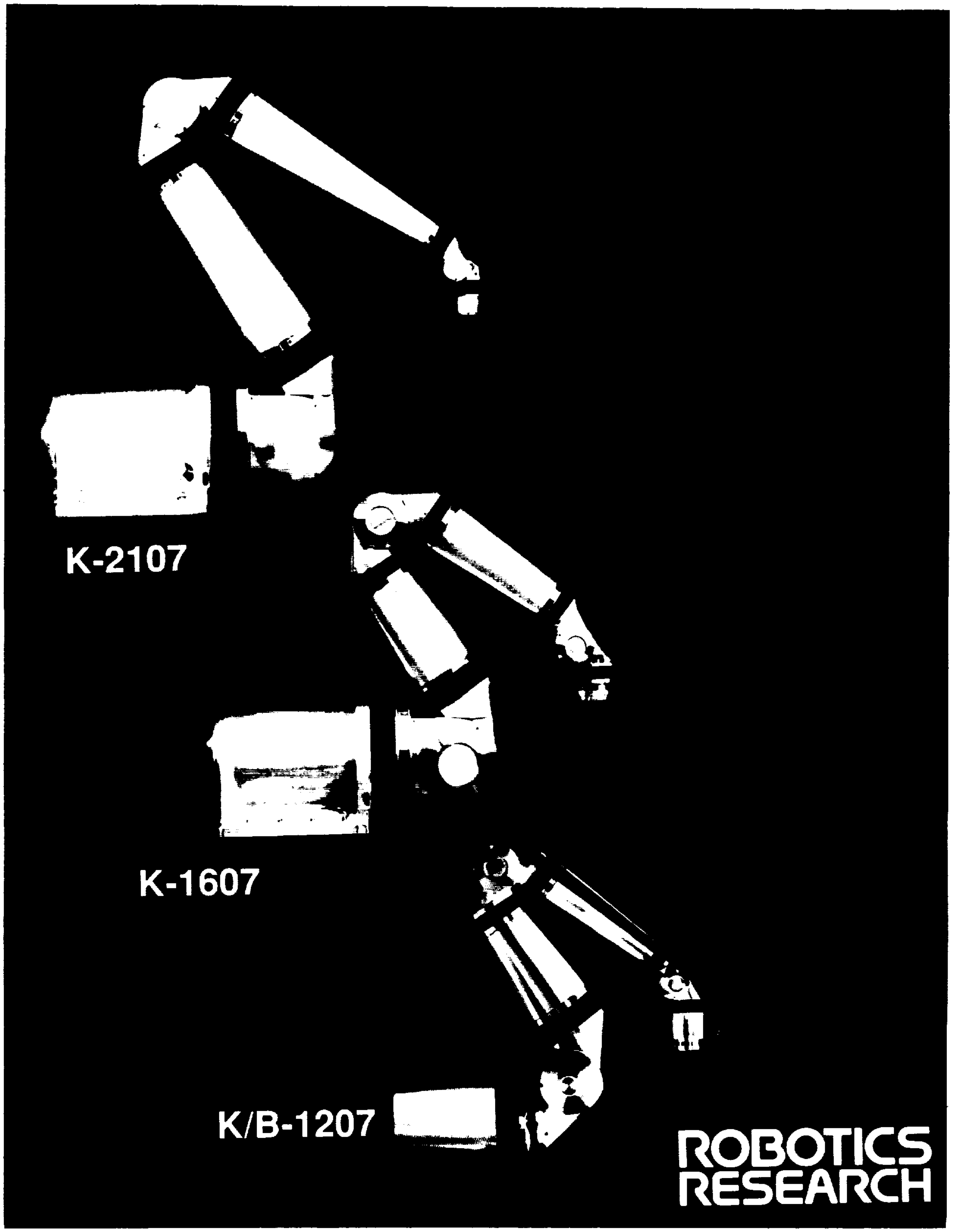


Figure 2:
Configuration of Modules in Three 7 DOF Manipulator Arm Models

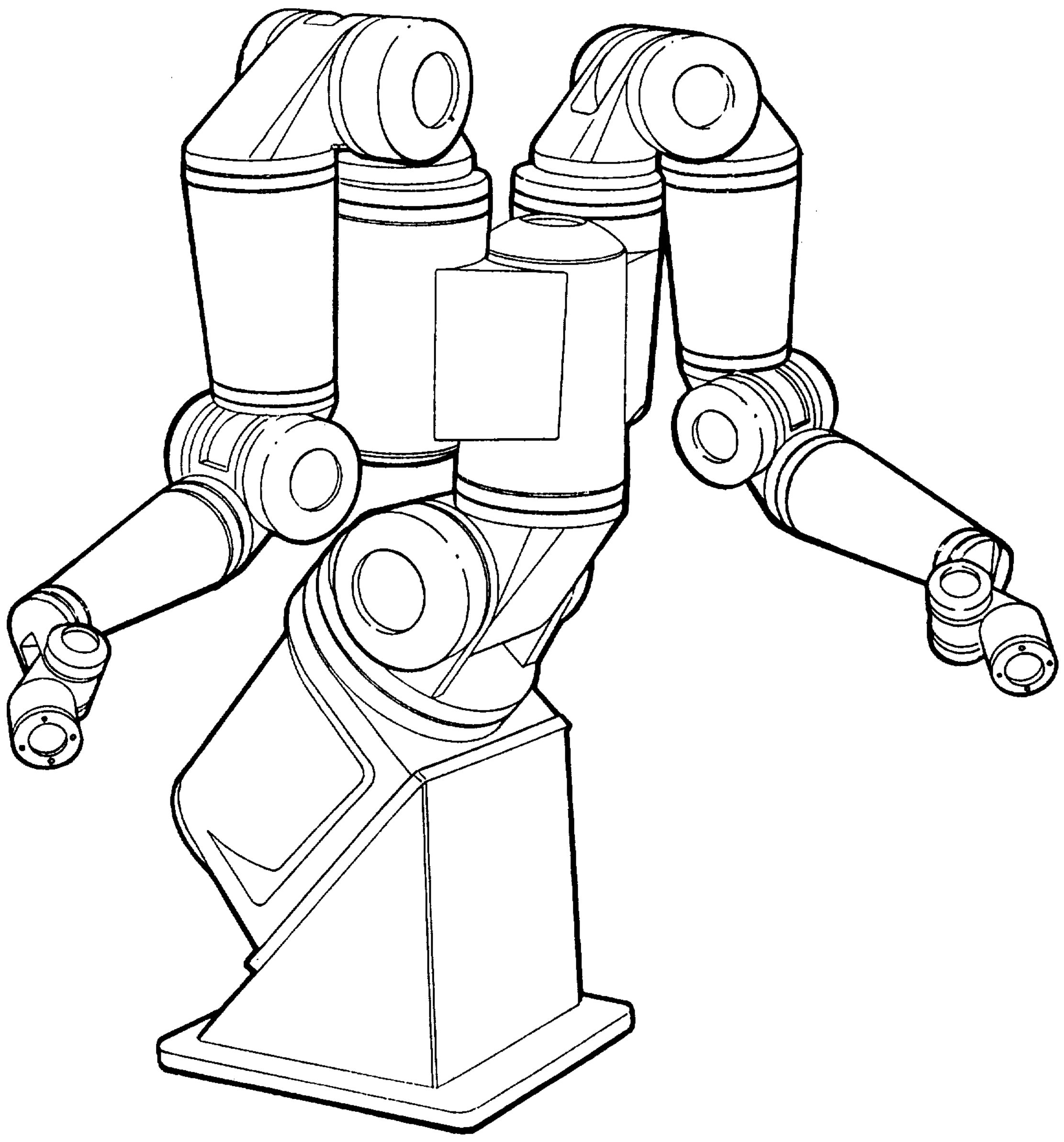


Figure 3:
K/B-2017 Dexterous Manipulator
17 Degrees of Freedom

encompasses the normal resonant frequency range of the actuator package. This torque loop functions to position the motor inertia, however it must, to cause the drive to provide the desired applied joint torque.

As previously mentioned, the torque-loop system provides significant advantages beyond eliminating the harmonic drive resonant response to provide smooth motion. A promising avenue in research on manipulator control is commanding axis torques to achieve high-bandwidth tool force control, or "impedance control"[2]. In K-Series arms, the fastest loop at work in the servo-control system is the torque loop. This innermost loop can remain in operation while mode switching. The torque loop also encompasses and compensates for motor, motor seal and drive friction. Viewed from outside the loop, it imparts to the system many of the attributes of best direct-drive manipulators, while avoiding the relatively poor torque-to-mass ratios of the direct-drive motors.

4.0 MOTION CONTROL SYSTEM

Type 2 Motion Controller The Type 2 Motion Control system configured for the K/B-2017 from our standard hardware and software modules provides an open and modular architecture which enables the user to operate the manipulator in a number of different control modes and at various control levels[1,3]. (Refer to Figure 4.) Like Type 2 Motion Controllers supplied with Robotics Research's 7 DOF arms, this 80386/80387-based hierarchical control system is structured following principles set forth in the NASREM architecture, developed by Dr. James S. Albus, et al, at the National Institute of Standards and Technology (formerly National Bureau of Standards)[4]. The Type 2 system handles trajectory control, inverse kinematics and servo-control functions for the manipulator. The trajectory level accepts Cartesian toolpoint commands from the user's host for each manipulator arm and moves each tool centerpoint from its current to its commanded positions. The kinematics level executes Cartesian-to-joint and joint-to-Cartesian transformations for the 17 DOF system on a 50 millisecond basis. The servo-control level accepts position, velocity, torque or current commands for each joint in the manipulator, interchangeably and independent of the mode of other joints, and closes all 17 servoloops on a 5 millisecond basis. A bus-to-bus interface is provided for high-speed host communications.

For this 17 DOF configuration, the Type 2 provides means to control the orientation of the elbow of each manipulator arm, independent of toolpoint position and orientation, using "Elbow-orbit" commands (Figure 5), as well as means to control torso posture using "Torso-orbit" commands (Figure 6). All joints in the manipulator arms and torso-waist assembly are coordinated and continuously participate in a move, i.e., a commanded motion at the toolpoint of either arm causes all 17 axes to move appropriately under the redundancy criteria which are in effect. In addition, the system can be controlled in any desired topological degeneration of its initial 17-axis geometry.

Control Philosophy Investigators at Robotics Research Corporation believe that complex robotic systems will increasingly utilize the standardized NASREM hierarchical control architecture, in which each successively higher level has a broader purview with respect to space and time, and is equipped with sensory feedback, memory and logical functions appropriate to its level of responsibility. In this context, authority over how to use the kinematic redundancy in the manipulator will not reside within any single level of the control system, but will be affected by decisions made at all levels.

Robotics Research is principally concerned with those levels of the robot control system responsible for making "reflexive motion control" decisions based on local, kinesthetic sensors mounted on the manipulator. These might be viewed as "brainstem" functions, analogous to the autonomic or sympathetic divisions of the central nervous system in biological models. (They are encompassed by Levels 1, 2 and 3 in the NASREM model-- "Servo", "Primitive", and "Elemental Move".)

The reflexive motion control approach developed by Robotics Research has been designed to accommodate the simultaneous operation of a wide range of potential redundancy criteria, including--

1. Reflexive Collision Avoidance
2. Joint Travel Limit Avoidance
3. Impedance Control
4. Torque Management and Redistribution
5. Velocity Management and Redistribution
6. Pose Optimization
7. Mechanical Advantage and Positioning Resolution Management
8. Suspension Emulation
9. Graceful Degradation of Kinematics

A basic strategy is to ensure that the system remains sufficiently redundant to satisfy all of the objective functions in force. A hierarchy of competing rules, or objective functions, can then be defined to make a balanced decision at each clock cycle about how best to dispose manipulator redundancy. We propose that, in general, the robot should attempt to execute the commanded toolpoint trajectory,

1. while avoiding collisions with itself, and
2. while avoiding collisions with objects that are detected in the robot's working envelope, and
3. while recognizing singularities intrinsic to its mechanical geometry and using them appropriately,

a) to produce energy-efficient, graceful motion, or
b) to increase leverage (mechanical advantage), or
c) to control "impedance" at the toolpoint, and

4. while "favoring" joints whose actuators are sensed to be closer to their thermal limits than others.

Obviously, while a higher level in the hierarchical control system may elect to override or reprioritize these objectives based on its broader view of the situation, in normal operation, no one criterion is ever permitted to monopolize the available redundancy. Competing functions coexist. An exceptionally computationally-efficient generalized inverse solver for the Jacobian of redundant systems, devised by one of the authors (Vold), provides means to reduce to practice such a philosophy, even for massively-redundant manipulator configurations.

Robotics Research's Paradigm for Redundant Motion Control The manipulator is construed in our control mathematics as a mechanism where, to each joint, there is associated a spring value with specified stiffness and origin, a viscous damper value, and an inertia value. To each link there is a center of gravity, a mass value, and an inertia tensor. Robotics Research's Spring-Mass-Damper (SMD) model is not a simulation-- dynamic linkage parameters are chosen to provide desirable manipulator behavior, but do not correspond to its actual physical properties.

Motion is accomplished by prescribing incremental Cartesian motion at the end-effector(s), and imposing dynamic joint loads generated by the damping and inertia factors. Additional joint loads may be superposed to accomplish subgoals. Motion may be biased by modifying spring origins. Joint increments are found which are compatible with the instantaneous Jacobian matrix of the system. Using Robotics Research's proprietary algorithm, the solution time for the 17 DOF manipulator shown herein is less than 50 milliseconds operating in C code on a single 20 mHz Intel 80386 cpu with a 80387 coprocessor. Optionally, a basis for the 5 dimensional nullspace of the Jacobian is also computed. The form of our procedures is inherently amenable to parallel processing architectures when higher update rates are required.

Joint position commands are generated that tend to be compatible with the real actuator's capabilities by employing a mass and inertia distribution that approximates the physical system. One may view the inclusion of damping and inertia factors in this model as providing tunable low-pass filters for the actuator velocities and torques. The specified masses and inertias need not, however, be exact. Indeed, by setting those values to zero, a quasi-static solution is generated which is quite satisfactory for slow speed motion.

Reflexive Collision Avoidance Sensory-interactive collision avoidance for redundant systems is implemented by associating with each point obstacle a repellent Cartesian force field 5. Integrating this force field results in a torque load per joint that tends to push the manipulator away from the obstacle. Multiple force fields are superposed in a linear fashion. Joint limits are avoided by applying counteracting torque loads applied to each axis as it approaches its end-of-travel.

Suspension Emulation During abrupt end-effector maneuvers, peak actuator torque requirements may be reduced substantially by dissipating the manipulator's kinetic energy over longer periods of time. This is implemented by using realistic mass and inertia distributions in the motion control model. In effect, the end-effectors behave like the only unsprung masses in the system.

Impedance Control Impedance control is effected by specifying desired end-effector impedance, position and velocity. As previously discussed, the joint actuators in this 17 DOF manipulator utilize torque loops at the servo level which provide many of the beneficial characteristics of direct-drive motors; in this context, a reasonably simple scheme can be employed to implement impedance control.

The specification of desired end-effector position and velocity translates into an equivalent desired position and velocity in joint space. The specification of end-effector stiffnesses translates directly into joint space by forming the congruent transform of the Cartesian stiffness matrix with the Jacobian of the kinematic transformations. In a redundant manipulator, this joint space stiffness suffers a rank loss equal to the redundancy of the manipulator, such that to the Jacobian must be adjoined a basis for the null space. A specification of impedance for orbit moves, such as torso pitch and shoulder pitch, is implicit.

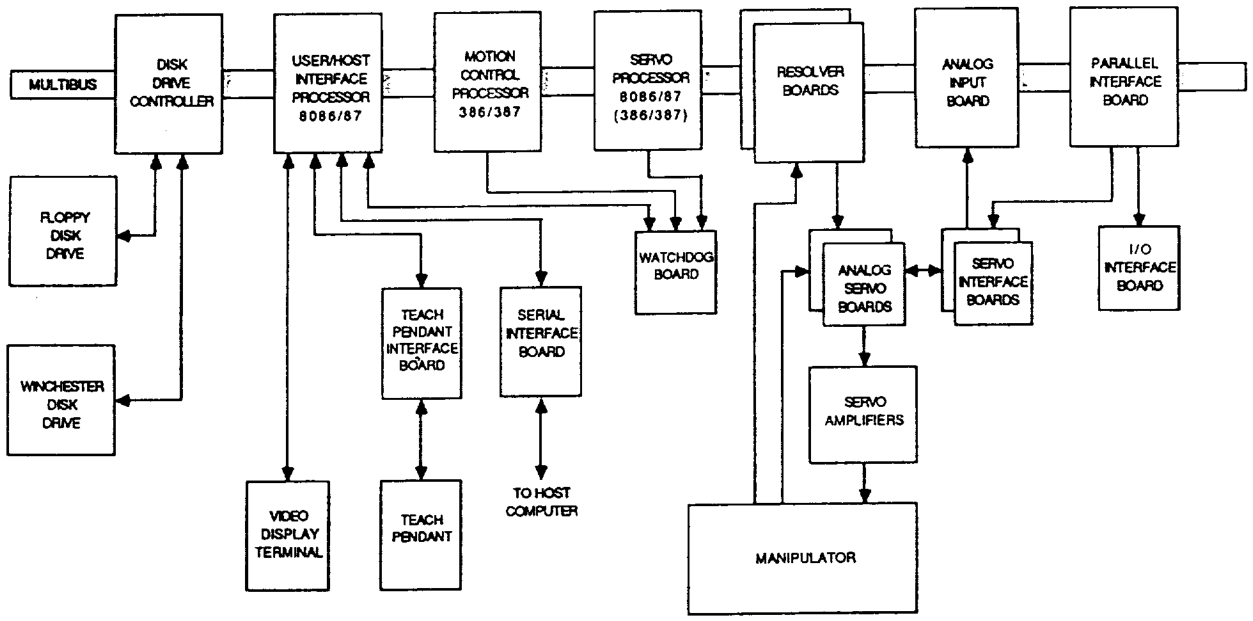


Figure 4:
Type 2 Motion Controller
Hardware Architecture

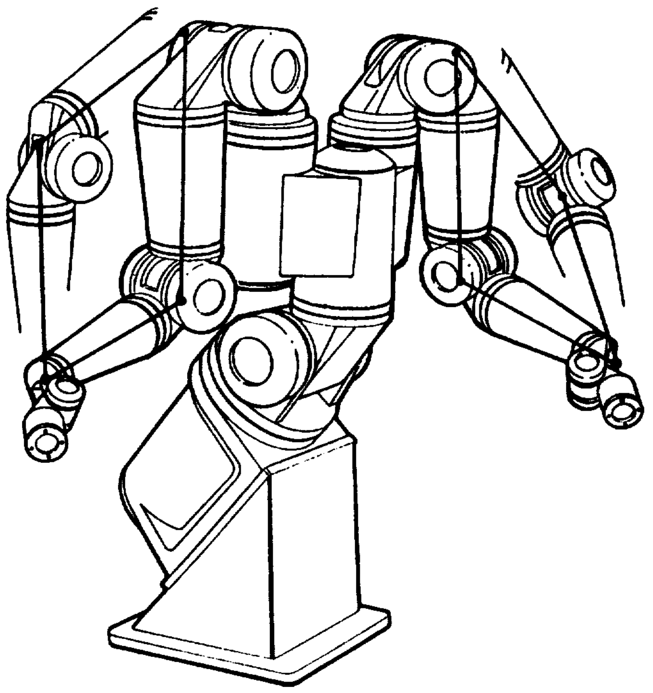


Figure 5:
Elbow "Orbit" Moves
with 17 DOF Manipulator

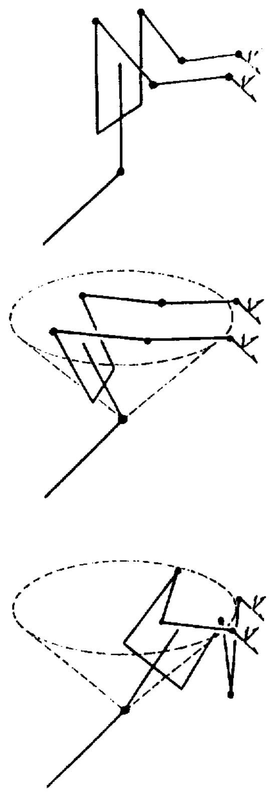


Figure 6:
Torso "Orbit" Moves
with 17 DOF Manipulator

Mechanical Advantage and Positioning Resolution Management In conventional manipulators, singular positions announce their presence by demanding high joint velocities to achieve small end-effector motions. While a near-singular position is not usually desirable, the end-effector enjoys considerable mechanical advantage at that instant. Also, precision of motion at the end-effector is greatly increased.

In a redundant manipulator controlled according to our Spring-Mass-Damper (SMD) model, the emulation of nature tends to mitigate excessive velocities and torques, such that singular regions in joint space are avoided. However, in order to achieve required precision for fine manipulation, or sufficient leverage for a high-force task, a redundant manipulator can be made to seek out near-singular poses using this technique while maintaining sufficient redundancy to break singular deadlocks.

Torque Management and Redistribution Torque management enables the manipulator to carry out its tasks, while controlling torques, velocities and power within safe limits and with economy. Higher level control functions may preselect desirable poses, while, at the reflexive motion control level, torque management is implemented through judicious selection of spring stiffnesses, damping and inertias in the SMD model. Favoring overloaded or temporarily "over-worked" joints is an example of torque management.

Pose Optimization Pose control allows the redundant manipulator to be configured for specific task objectives. Pose is normally thought of as a quasi-static configuration in joint space, but in our control philosophy, pose also includes velocity, such that the inertia in the manipulator can be exploited.

Pose is normally based upon experience or task planning, but may also be determined on-the-fly by the reflexive motion control level tracking the inertia distribution and second-order properties of the kinematic transformations. Pose is implemented by biasing the spring origins in the conceptual control model.

5.0 CONCLUSIONS

The 17 degree of freedom K/B-2017 Dexterous Manipulator and Type 2 Motion Controller described herein, assembled using standard hardware and software modules, represents a natural extension of Robotics Research's modular manipulator series. It is one of a number of new kinematic configurations for manipulators that can be practically implemented using these proven components.

The anthropomorphic branching topology used in this system appears to the authors to be a good solution to the problem of achieving long reach and a large working envelope, without sacrificing the intrinsic advantages of compact, highly dexterous tool-handling arms.

An exceptionally wide range of postures can be assumed by this massively redundant system and a variety of competing objective functions can be satisfied simultaneously. Posture control functions such as reflexive obstacle avoidance and the favoring of overloaded joints, of nominal utility in seven-axis arms, can be demonstrated in a 17 degree of freedom device to have great benefit for overall manipulation capability, reliability and task performance.

We believe this highly anthropomorphic device promises to be an ideal testbed for research in "man-equivalent" telerobots for space servicing, nuclear servicing and defense applications.

6.0 REFERENCES

1. J. P. Karlen, J. M. Thompson, Jr., J. D. Farrell, "Design and Control of Modular, Kinematically Redundant Manipulators", Second AIAA / NASA / USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program, March 9-11, 1987, Arlington, VA (Robotics Research Corporation).
2. N. Hogan, "Impedance Control: An Approach to Manipulation, Part 1: Theory", ASME Journal of Dynamic Systems Measurement and Control, Vol. 107, pp. 1-7, March, 1985.
3. J. D. Farrell, "Pragmatic Control of Kinematically Redundant Manipulators", 1988 American

Control Conference, July 15-17, 1988, Atlanta, GA (Robotics Research Corporation).

4. J. S. Albus, H. G. McCain, R. Lumia, "NASA / NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", National Bureau of Standards, Technical Note 1235, June, 1987.

5. J. P. Karlen, Jack M. Thompson, Jr., James D. Farrell, Håvard I. Vold, "Reflexive Obstacle Avoidance for Kinematically-Redundant Manipulators", NASA Conference on Space Telerobotics, January 31-February 2, 1989, Jet Propulsion Laboratory, Pasadena, CA (Robotics Research Corporation).

N90-29002

A NEW APPROACH TO GLOBAL CONTROL OF REDUNDANT MANIPULATORS

Homayoun Seraji
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

Abstract

A new and simple approach to configuration control of redundant manipulators is presented in this paper. In this approach, the redundancy is utilized to control the manipulator configuration directly in task space, where the task will be performed. A number of kinematic functions are defined to reflect the desirable configuration that will be achieved for a given end-effector position. The user-defined kinematic functions and the end-effector Cartesian coordinates are combined to form a set of task-related configuration variables as generalized coordinates for the manipulator. An adaptive scheme is then utilized to globally control the configuration variables so as to achieve tracking of some desired reference trajectories. This accomplishes the basic task of desired end-effector motion, while utilizing the redundancy to achieve any additional task through the desired time variation of the kinematic functions. The control law is simple and computationally very fast, and does not require the complex manipulator dynamic model.

1. Introduction

The remarkable dexterity and versatility that the human arm exhibits in performing various tasks can be attributed largely to the kinematic redundancy of the arm, which provides the capability of reconfiguring the arm without affecting the hand position. A robotic manipulator is called (kinematically) "redundant" if it possesses more degrees-of-freedom than is necessary for performing a specified task. For instance, in the three-dimensional space, a manipulator with seven or more joints is redundant since six degrees-of-freedom are sufficient to position and orient the end-effector in any desired configuration. Redundancy of a robotic manipulator is determined relative to the particular task to be performed. For example, in the two-dimensional space, a planar robot with three joints is redundant for achieving any end-effector position, whereas the robot is non-redundant for tasks involving both position and orientation of the end-effector. In a non-redundant manipulator, a given position and orientation of the end-effector corresponds to a single set of joint angles and an associated unique robot configuration (with distinct poses such as elbow up or down). Therefore, for a prescribed end-effector motion, the evolution of the robot configuration is uniquely determined. When this evolution is undesirable due to collision with obstacles, approaching kinematic singularities or reaching joint limits, there is no freedom to reconfigure the robot so as to reach around the obstacles, or avoid the singularities and joint limits.

Redundancy in the manipulator structure yields increased dexterity and versatility for performing a task due to the infinite number of joint motions which result in the same end-effector trajectory. However, this richness in choice of joint motions complicates the manipulator control problem considerably. In order to take full advantage of the capabilities of redundant manipulators, effective control schemes should be developed to utilize the redundancy in some useful manner. During recent years, redundant manipulators have been the subject of considerable research, and several methods have been suggested to resolve the redundancy. In 1969, Whitney [1] suggested the use of Jacobian pseudoinverse to resolve the redundancy. Over the past two decades, most of the research on redundant manipulators has been explicitly or implicitly based on the pseudoinverse approach for the utilization of redundancy through *local* optimization of some criterion functional. Furthermore, most proposed methods resolve the redundancy in joint space and are concerned solely with solving the inverse kinematic problem for redundant manipulators.

In this paper, a new and conceptually simple approach for *configuration control* of redundant manipulators is presented, which takes a complete departure from the conventional pseudoinverse methods. In this approach, the redundancy is utilized for *global* control of the manipulator configuration directly in *task space*, where the task will be performed, thus avoiding the complicated inverse kinematic transformation. A set of kinematic functions is chosen to reflect the desired additional task that will be performed due to the redundancy. The kinematic functions succinctly characterize the "self-motion" of the manipulator, in which the internal movement of the links does not move the end-effector. In other words, the kinematic functions are used to "shape" the manipulator configuration, given the end-effector position and orientation. The end-effector Cartesian coordinates and the kinematic functions are combined to form a set of "configuration variables" which describe the physical configuration of the entire manipulator in a task-related coordinate system. The control scheme then ensures that the configuration variables track some desired trajectories as closely as possible, so that the evolution of the manipulator configuration meets the task requirements. The control law is adaptive and does not require knowledge of the complex dynamic model or parameter values of the manipulator or payload. The scheme can be implemented either in a centralized or a decentralized control structure, and is computationally very fast as a real-time algorithm for on-line control of redundant manipulators.

2. Configuration Control Scheme

The mechanical manipulator under consideration consists of a linkage of rigid bodies with n revolute or prismatic joints. Let T be the $n \times 1$ vector of torques or forces applied at the joints and θ be the $n \times 1$ vector of the resulting relative joint rotations or translations. The dynamic equation of motion of the manipulator which relates T to θ can be represented in the general form [2]

$$M(\theta)\ddot{\theta} + N(\theta, \dot{\theta}) = T \tag{1}$$

where the matrices M and N are highly complex nonlinear functions of θ, $\dot{\theta}$, and the payload. Let the $m \times 1$ vector Y (with $m < n$) represent the position and orientation of the end-effector (last link) with respect to a fixed Cartesian coordinate system in the m-dimensional task space where the task is to be performed. The $m \times 1$ end-effector coordinate vector Y is related to the $n \times 1$ joint angle vector θ by the forward kinematic model

$$Y = Y(\theta) \tag{2}$$

where $Y(\theta)$ is an $m \times 1$ vector whose elements are nonlinear functions of the joint angles and link parameters and embodies the geometry of the manipulator. For a redundant manipulator with $m < n$, a Cartesian coordinate vector (such as Y) that specifies the end-effector position and orientation does not constitute a set of generalized coordinates to completely describe the manipulator dynamics. Nonetheless, equations (1) and (2) form a valid dynamic model to describe the *end-effector* motion itself in the task space. The desired motion of the end-effector is represented by the reference position and orientation trajectories denoted by the $m \times 1$ vector $Y_d(t)$, where the elements of $Y_d(t)$ are continuous twice-differentiable functions of time. The vector $Y_d(t)$ embodies the information on the "*basic task*" to be accomplished by the end-effector in the task space.

We shall now discuss the definition of configuration variables and the adaptive control of redundant manipulators in the subsequent sections.

2.1 Definition of Configuration Variables

Let $r = n - m$ be the "degree-of-redundancy" of the manipulator, i.e. the number of "extra" joints. Let us define a set of r kinematic functions $\{\phi_1(\theta), \phi_2(\theta), \ldots, \phi_r(\theta)\}$ to reflect the "*additional task*" that will be performed due to the manipulator redundancy. Each ϕ_i can be a function of the joint angles $\{\theta_1, \ldots, \theta_n\}$ and the link geometric parameters. The choice of the kinematic functions can be made in several ways to represent, for instance, the coordinates of any point on the manipulator, or any combination of the joint angles. The kinematic functions succinctly characterize the "self-motion" of the manipulator, in which the internal movement of the links does not move the end-effector.

For the sake of illustration, let us consider a planar three-link arm as shown in Figure 1(i). The basic task is to control the end-effector position coordinates $[x, y]$ in the base frame. Suppose that we fix the end-effector position and allow internal motion of the links so that the arm takes all possible configurations. It is found that the locus of point A is an arc of a circle with center O and radius ℓ_1 which satisfies the distance constraint $AC \leq (\ell_2 + \ell_3)$. Likewise, the locus of point B is an arc of a circle with center C and radius ℓ_3 which satisfies $OB \leq (\ell_1 + \ell_2)$. The loci of A and B are shown as hatched arcs in Figure 1(i), and represent the self-motion of the arm. Now, in order to characterize the self-motion, we can select a kinematic function $\phi(\theta)$ to represent, for instance, the terminal angle $\phi = \theta_1 + \theta_2 + \theta_3$, or alternatively we can designate the wrist height y_B as the kinematic function $\phi = \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2)$. The choice of ϕ clearly depends on the particular task that we wish to perform by the utilization of redundancy, in addition to the end-effector motion. Let us now consider a spatial 7 dof arm [3] as shown in Figure 1(ii), in which the end-effector position and orientation are of concern. The self-motion of this arm corresponds to rotation of the elbow point A about the line OB joining the shoulder to the wrist. We can now define the kinematic function $\phi(\theta) = \alpha$, where α is the angle between a normal line from A to OB and a line perpendicular to OB in the vertical plane, as shown in Figure 1(ii). The kinematic function ϕ then succinctly describes the redundancy and gives a simple characterization of the self-motion.

Once a set of r task-related kinematic functions $\phi = \{\phi_1, \phi_2, \ldots, \phi_r\}$ is defined, we have partial information on the manipulator configuration. The set of m end-effector position and orientation coordinates $Y = \{y_1, y_2, \ldots, y_m\}$ provides the remaining information on the configuration. Let us now combine the two sets ϕ and Y to obtain a complete set of n configuration variables as

$$\begin{aligned} X &= \{Y, \phi\} = \{y_1, y_2, \ldots, y_m \vdots \phi_1, \phi_2, \ldots, \phi_r\} \\ &= \{x_1, x_2, \ldots, x_n\} \end{aligned} \tag{3}$$

The $n \times 1$ vector X is referred to as the *"configuration vector"* of the redundant manipulator and the elements of X, namely $\{x_1, \ldots, x_n\}$, are called the *"configuration variables."* The configuration variables $\{x_1, \ldots, x_n\}$ constitute a set of generalized coordinates for the redundant manipulator. Using the configuration vector X, the manipulator is fully specified and is no longer redundant in this representation. It is noted that in some applications, certain end-effector coordinates are not relevant to the task, for instance, in a spot welding task the orientation of the end-effector is not important. In such cases, the present approach allows the designer to replace the insignificant end-effector coordinates with additional kinematic functions which are more relevant to that particular application. In fact, if $m'(< m)$ end-effector coordinates are specified, then $n - m' = r'(> r)$ kinematic functions can be defined.

The augmented forward kinematic model which relates the configuration vector X to the joint angle vector θ is now given by

$$X = \begin{pmatrix} Y(\theta) \\ \cdots\cdot \\ \phi(\theta) \end{pmatrix} = X(\theta) \tag{4}$$

From equation (4), the differential model which relates the rates of change of X and θ is obtained as

$$\dot{X}(t) = J(\theta)\dot{\theta}(t) \tag{5}$$

where

$$J(\theta) = \begin{pmatrix} J_e(\theta) \\ \cdots\cdots \\ J_c(\theta) \end{pmatrix} = \begin{pmatrix} \frac{\partial Y}{\partial \theta} \\ \cdots \\ \frac{\partial \phi}{\partial \theta} \end{pmatrix} \tag{6}$$

is the $n \times n$ ***augmented Jacobian matrix***. The $m \times n$ submatrix $J_e(\theta) = \frac{\partial Y}{\partial \theta}$ is associated with the end-effector, while the $r \times n$ submatrix $J_c(\theta) = \frac{\partial \phi}{\partial \theta}$ is related to the kinematic functions. The two submatrices J_e and J_c combine to form the square Jacobian matrix J.

The augmented Jacobian matrix J can be used to test the functional independence of the kinematic functions $\{\phi_1, \ldots, \phi_r\}$ and the end-effector coordinates $\{y_1, \ldots, y_m\}$. For the set of configuration variables $X = \{x_1, \ldots, x_n\}$ to be functionally independent throughout the workspace, it suffices to check that det $[J(\theta)]$ is not identically zero for *all* θ, [4]. In other words, when the augmented Jacobian matrix J is rank-deficient for *all* values of θ, the kinematic functions chosen are functionally dependent on the end-effector coordinates and a different choice of ϕ is necessary. When det $[J(\theta)]$ is not identically zero, the configuration variables $\{x_1, \ldots, x_n\}$ are not functionally dependent for all θ. Nonetheless, there can be certain joint configurations $\theta = \theta_o$ at which det $[J(\theta_o)] = 0$, i.e., the augmented Jacobian matrix J is rank-deficient. This implies that the rows J^i of J satisfy the linear relationship $\sum_{i=1}^{n} c_i J^i = 0$, where c_i are some constants not all zero. Since the changes of the configuration variables and joint angles are related by $\Delta x = J(\theta)\Delta\theta$, we conclude that at $\theta = \theta_o$,

$\sum_{i=1}^{n} c_i \Delta x_i = 0$. Therefore at a Jacobian singularity, the changes in the configuration variables $\{\Delta x_1, \ldots, \Delta x_n\}$ must satisfy the constraint relationship $\sum_{i=1}^{n} c_i \Delta x_i = 0$, and hence the configuration vector X cannot be changed arbitrarily.

From expression (6), it is clear that the Jacobian matrix J will be singular at any joint configuration for which the submatrix J_e is rank-deficient; i.e., at any end-effector singular configuration. In addition, the Jacobian J will be singular at those values of θ for which the submatrix J_c loses full rank. The latter singularities of J, which are due to the kinematic functions, are inevitably introduced whenever an additional task is employed to utilize the redundancy. However, by judicious choice of the kinematic functions, some of the singularities due to J_c may be avoided, and the singularities of J may be shifted to the unusable part of the workspace. Note that even when J_e and J_c have full ranks individually, the augmented matrix J may still be rank-deficient.

2.2 Adaptive Configuration Control

Suppose that a user-defined *"additional task"* can be expressed by the following kinematic equality constraint relationships

$$\begin{aligned} \phi_1(\theta) &= \phi_{d1}(t) \\ \phi_2(\theta) &= \phi_{d2}(t) \\ \vdots \quad & \quad \vdots \\ \phi_r(\theta) &= \phi_{dr}(t) \end{aligned} \tag{7}$$

where $\phi_{di}(t)$ denotes the desired time variation of the kinematic function ϕ_i and is a user-specified continuous twice-differentiable function of time. The kinematic relationships (7) can be represented collectively in the vector form

$$\phi(\theta) = \phi_d(t) \tag{8}$$

where ϕ_d is an $r \times 1$ vector. Equation (8) represents a set of "kinematic constraints" on the manipulator and defines the task that will be performed *in addition to* the basic task of desired end-effector motion. The kinematic equality constraints (8) are chosen to have physical interpretations and are used to formulate the desirable characteristics of the manipulator configuration in terms of motion of other members of the manipulator. For instance, in the 7 dof arm of Figure 1(ii), by controlling the elbow height as well as the hand coordinates, we can ensure that the elbow avoids collision with vertical obstacles (such as walls) in the workspace while the hand tracks the desired trajectory. Alternatively, a particular posture of the manipulator which represents a singular configuration can be avoided by an appropriate choice of the kinematic constraints in terms of the joint angles. The proposed formulation appears to be a highly promising approach to the additional task performance in comparison with the previous approaches which attempt to minimize or maximize criterion functionals, since we are now able to make a more specific statement about the evolution of the manipulator configuration. The present approach also covers the intuitive solution to redundant arm control in which certain joint angles are held constant for a portion of the task in order to resolve the redundancy. The functional forms of the kinematic functions ϕ_i and their desired behavior ϕ_{di} may vary widely for different additional tasks, making the approach unrestricted to any particular type of application.

Based on the foregoing formulation, we can now consider the manipulator with the $n \times 1$ configuration vector $X = \binom{Y}{\phi}$ and the $n \times n$ augmented Jacobian matrix $J = \binom{J_e}{J_c}$. Once the desired motion of the end-effector $Y_d(t)$ is specified for the particular basic task and the required evolution of the kinematic functions $\phi_d(t)$ is specified to meet the desired additional task, the $n \times 1$ desired configuration vector $X_d(t) = \binom{Y_d(t)}{\phi_d(t)}$ is fully determined. The configuration control problem for the redundant manipulator is to devise a dynamic control scheme as shown in Figure 2 which ensures that the manipulator configuration vector $X(t)$ tracks the desired trajectory vector $X_d(t)$ as closely as possible. In the control system shown in Figure 2, the actual end-effector position $Y(t)$ and the current value of the kinematic functions $\phi(t)$ are fed back to the controller. The controller uses this feedback information together with the commanded end-effector motion $Y_d(t)$ and the desired time variation $\phi_d(t)$ to compute the driving torques $T(t)$ that are applied at the manipulator joints so as to meet the basic and additional task requirements simultaneously.

Different control strategies can be improvised to meet the above tracking requirement, taking into account the dynamics of the manipulator given by equation (1). There are two major techniques for the design of tracking controllers in task space, namely model-based control and adaptive control. For the model-based control [5], the manipulator dynamics is first expressed in task space as

$$M_x(\theta)\ddot{X} + N_x(\theta, \dot{\theta}) = F \tag{9}$$

where F is the $n \times 1$ "virtual" control force vector in the task space, and M_x and N_x are obtained from equations (1)-(6). The control law which achieves tracking through global linearization and decoupling is given by

$$F = M_x(\theta)\left[\ddot{X}_d(t) + K_v\left(\dot{X}_d(t) - \dot{X}(t)\right) + K_p\left(X_d(t) - X(t)\right)\right] + N_x(\theta, \dot{\theta}) \tag{10}$$

where K_p and K_v are constant position and velocity feedback gain matrices. This control formulation requires precise knowledge of the full dynamic model and parameter values of the manipulator and the payload. The alternative approach is the adaptive control technique in which the on-line adaptation of the controller gains eliminates the need for the complex manipulator dynamic model. In this section, we adopt an adaptive control scheme which has been developed recently and validated experimentally on a PUMA industrial robot [6-8]. The adaptive controller produces the control signal based on the observed performance of the manipulator and has therefore the capability to operate with minimal information on the manipulator/payload and to cope with unpredictable gross variations in the payload. The proposed adaptive control scheme is developed within the framework of Model Reference Adaptive Control (MRAC) theory, and the adaptive tracking control law in the task space is given by [6]

$$F(t) = d(t) + [K_p(t)E(t) + K_v(t)\dot{E}(t)] + [C(t)X_d(t) + B(t)\dot{X}_d(t) + A(t)\ddot{X}_d(t)] \tag{11}$$

as shown in Figure 3. This control force is composed of three components, namely:

(i) The *auxiliary signal* $d(t)$ is synthesized by the adaptation scheme and improves transient performance while resulting in better tracking and providing more flexibility in the design.

(ii) The term $[K_p(t)E(t) + K_v(t)\dot{E}(t)]$ is due to the PD *feedback controller* acting on the position tracking-error $E(t) = X_d(t) - X(t)$ and the velocity tracking-error $\dot{E}(t) = \dot{X}_d(t) - \dot{X}(t)$.

(iii) The term $[C(t)X_d(t)+B(t)\dot{X}_d(t)+A(t)\ddot{X}_d(t)]$ is the contribution of the PD^2 *feedforward controller* operating on the desired position $X_d(t)$, the desired velocity $\dot{X}_d(t)$, and the desired acceleration $\ddot{X}_d(t)$.

The required auxiliary signal and feedback/feedforward controller gains are updated based on the $n \times 1$ "weighted" error vector $q(t)$ by the following simple adaptation laws [6]:

$$q(t) = W_pE(t) + W_v\dot{E}(t) \tag{12}$$

$$d(t) = d(0) + \delta_1 \int_0^t q(t)dt + \delta_2 q(t) \tag{13}$$

$$K_p(t) = K_p(0) + \alpha_1 \int_0^t q(t)E'(t)dt + \alpha_2 q(t)E'(t) \tag{14}$$

$$K_v(t) = K_v(0) + \beta_1 \int_0^t q(t)\dot{E}'(t)dt + \beta_2 q(t)\dot{E}'(t) \tag{15}$$

$$C(t) = C(0) + \nu_1 \int_0^t q(t)X_d'(t)dt + \nu_2 q(t)X_d'(t) \tag{16}$$

$$B(t) = B(0) + \gamma_1 \int_0^t q(t)\dot{X}_d'(t)dt + \gamma_2 q(t)\dot{X}_d'(t) \tag{17}$$

$$A(t) = A(0) + \lambda_1 \int_0^t q(t)\ddot{X}_d'(t)dt + \lambda_2 q(t)\ddot{X}_d'(t) \tag{18}$$

In equations (13)-(18), $\{\delta_1, \alpha_1, \beta_1, \nu_1, \gamma_1, \lambda_1\}$ are any positive scalar integral adaptation gains, and $\{\delta_2, \alpha_2, \beta_2, \nu_2, \gamma_2, \lambda_2\}$ are zero or any positive scalar proportional adaptation gains. In equation (12), $W_p = \text{diag}_i\{w_{pi}\}$ and $W_v = \text{diag}_i\{w_{vi}\}$ are constant $n \times n$ weighting matrices chosen by the designer to reflect the relative significance of the position and velocity errors E and $\dot{E}$ in forming the vector q. The values of the adaptation gains and weighting matrices determine the rate at which the tracking-errors converge to zero.

Since the control actuation is at the manipulator joints, the control force F is implemented as the joint torque T where

$$T(t) = J'(\theta)F(t) \tag{19}$$

The augmented Jacobian matrix $J(\theta)$ is used in equation (19) to map the task forces $F(t)$ to the joint torques $T(t)$. Equation (19) represents the fundamental relationship between the task and joint spaces and is the basis for implementation of any task-based control scheme [5]. Equation (19) can be rewritten as

$$T(t) = \left[J_e'(\theta) \vdots J_c'(\theta) \right] \begin{bmatrix} F_e(t) \\ \cdots\cdots \\ F_c(t) \end{bmatrix} = J_e'(\theta)F_e(t) + J_c'(\theta)F_c(t) \tag{20}$$

where F_e and F_c are the $m \times 1$ and $r \times 1$ control force vectors corresponding to the basic task and the additional task, respectively. It is seen that the total control torque is the sum of two components: $T_e = J_e' F_e$, for the end-effector motion (basic task), and $T_c = J_c' F_c$, for the kinematic constraints (additional task). Equation (20) shows distinctly the contributions of the basic and the additional tasks to the overall control torque. Under the joint control law (20), the desired end-effector trajectory $Y_d(t)$ is tracked, and the "extra" degrees-of-freedom are conveniently used to control the evolution of the manipulator configuration through tracking of the desired kinematic functions $\phi_d(t)$. In other words, the self-motion of the manipulator is controlled by first characterizing this motion in terms of a set of kinematic functions and then controlling these functions through trajectory tracking.

The adaptive control scheme presented in this section is extremely simple since the auxiliary signal and controller gains are evaluated from equations (12)-(18) by simple numerical integration by using, for instance, the trapezoidal rule. Thus the computational time required to calculate the adaptive control law (11) is extremely short. As a result, the scheme can be implemented for on-line control of redundant manipulators with high sampling rates, resulting in improved dynamic performance. This is in contrast to most existing approaches which require time-consuming optimization processes unsuitable for fast on-line control implementation. It is important to note that the adaptation laws (12)-(18) are based solely on the observed performance of the manipulator rather than on any knowledge of the complex dynamic model or parameter values of the manipulator and the payload.

3. Conclusions

A simple formulation for configuration control of redundant manipulators has been developed in this paper. The controller achieves trajectory tracking for the end-effector directly in the Cartesian space to perform some desired basic task. In addition, the redundancy is utilized by imposing a set of kinematic constraints on the manipulator to accomplish an appropriate additional task. The proposed formulation incorporates the kinematic constraints (additional task) and the end-effector motion (basic task) in a conceptually simple and computationally efficient manner to resolve the redundancy. Furthermore, the adaptive controller has a very simple structure and the controller gains are adjusted in a simple manner to compensate for changing dynamic characteristics of the manipulator. The adaptation laws are based on the observed performance of the manipulator rather than on any knowledge of the manipulator dynamic model. Thus, the adaptive controller is capable of ensuring a satisfactory performance when the payload mass is unknown and time-varying. Any approach used to resolve redundancy should be implementable as a real-time algorithm, and therefore the speed of computation is a critical factor. The small amount of computations required by the proposed method offers the possibility of fast real-time control of redundant manipulators.

4. Acknowledgement

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

5. References

[1] D.E. Whitney: "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man-Machine Systems*, 1969, Vol. MMS-10, No. 2, pp. 47-53.

[2] J.J. Craig: *Robotics - Mechanics and Control*, Addison-Wesley Publishing Company, Reading, MA, 1986.

[3] J.M. Hollerbach: "Optimum kinematic design for a seven degree of freedom manipulator," *Proc. 2nd Intern. Symp. on Robotics Research*, Kyoto, August 1984.

[4] R. Courant: *Differential and Integral Calculus*, Vol. II, Interscience Publishers Inc., New York, 1961.

[5] O. Khatib: "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, 1987, Vol. RA-3, No. 1, pp. 43-53.

[6] H. Seraji: "Direct adaptive control of manipulators in Cartesian space," *Journal of Robotic Systems*, 1987, Vol. 4, No. 1, pp. 157-178.

[7] H. Seraji: "A new approach to adaptive control of manipulators," *ASME Journ. Dynamic Systems, Measurement and Control*, 1987, Vol. 109, No. 3, pp. 193-202.

[8] H. Seraji: "Adaptive independent joint control of manipulators: Theory and experiment," *Proc. IEEE Intern. Conf. on Robotics and Automation*, Philadelphia, April 1988, pp. 854-861.

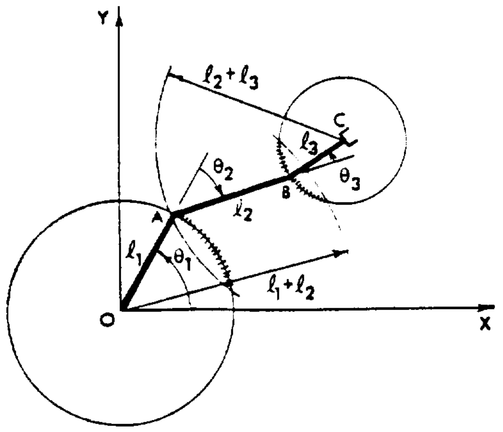


Figure 1(i). Self-motion of Planar 3 dof Arm

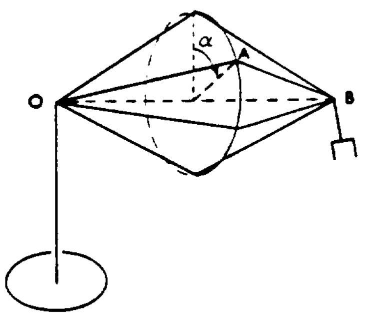


Figure 1(ii). Self-motion of Spatial 7 dof Arm

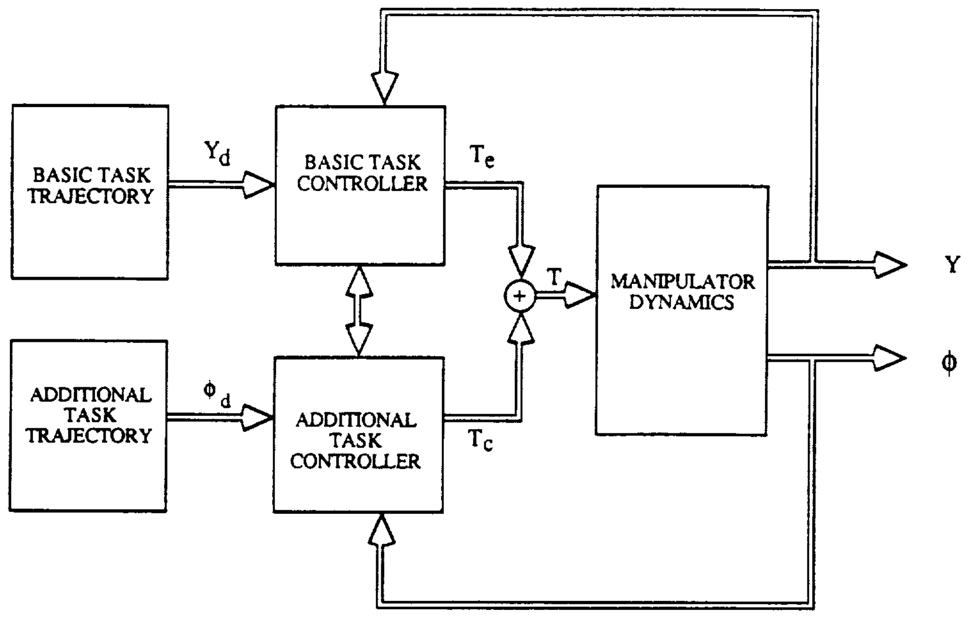


Figure 2. Architecture of Configuration Control Scheme

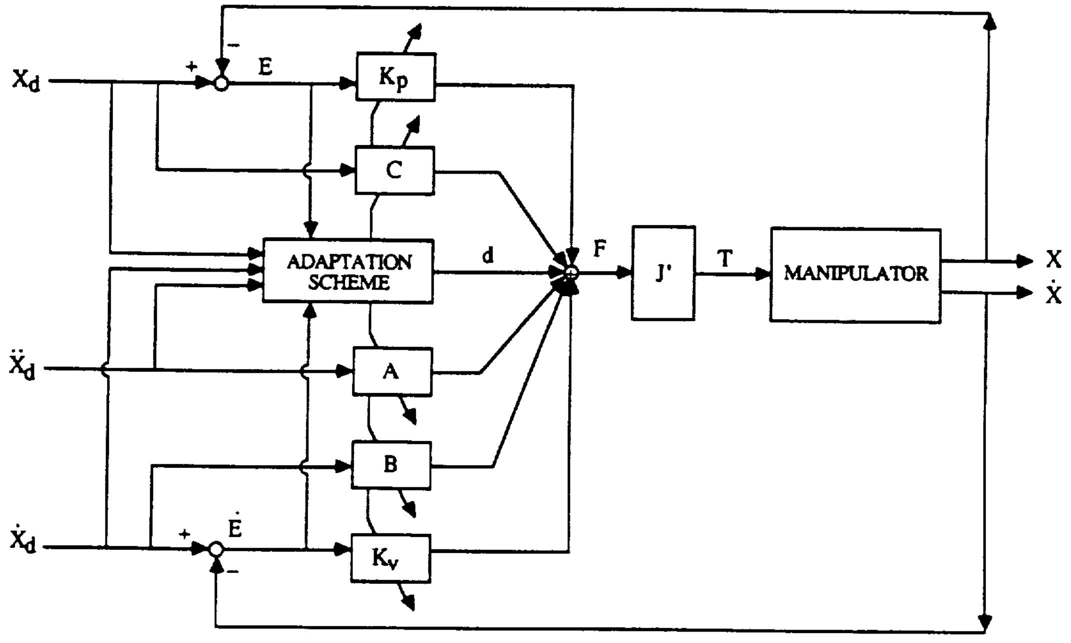


Figure 3. Adaptive Manipulator Control System

KINEMATIC FUNCTIONS FOR THE 7 DOF ROBOTICS RESEARCH ARM

K. Kreutz, M. Long, H. Seraji
Jet Propulsion Laboratory
California Institute of Technology

Abstract

The Robotics Research Model K–1207 manipulator is a redundant 7R serial link arm with offsets at all joints. To uniquely determine joint angles for a given end–effector configuration, the redundancy is parameterized by a scalar variable which corresponds to the angle between the manipulator elbow plane and the vertical plane. The forward kinematic mappings from joint–space to end–effector configuration and elbow angle, and the augmented Jacobian matrix which gives end–effector and elbow angle rates as a function of joint rates, are also derived.

1. Introduction

The Robotics Research Model K–1207 arm is a seven degree–of–freedom serial link manipulator which offers one extra degree of joint–space redundancy over that needed for the fundamental task of end–effector placement and orientation. In this paper, a reasonable task–space parameterization, ψ, is first given of the redundancy, and the forward kinematic mappings from joint space to end–effector configuration and ψ are then derived. We also give the augmented Jacobian, J^A, which gives end–effector rates and ψ as a function of joint rates. A longer and more complete version of this paper is available which contains proofs, as well as an analysis of the kinematic and algorithmic singularities of the augmented Jacobian.

2. Forward Kinematics

2.1. Mapping from Joint–Space to End–Effector Configuration

The Robotics Research Model K–1207 arm is essentially a 7R spherical–revolute–spherical manipulator, but with additional nonzero offsets (denoted by the link lengths a_i, $i = 1, \cdots, 6$) at each of the joints, as shown in Figures 1–3. Denavit–Hartenberg (D–H) link frame assignments are given in accordance with the convention described in [1]. This assignment results in the following general form of the interlink homogeneous transformation matrix:

$$ {}^{i-1}T_i = \begin{pmatrix} {}^{i-1}R_i & {}^{i-1}P_i \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & \alpha_{i-1} \\ \sin\theta_i \cdot \cos\alpha_{i-1} & \cos\theta_i \cdot \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \cdot \sin\alpha_{i-1} \\ \sin\theta_i \cdot \sin\alpha_{i-1} & \cos\theta_i \cdot \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cdot \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

where θ_i denotes the i^{th} joint angle. The D–H parameters for the K–1207 arm are given in Table 1. The link frame assignments for the K–1207 are given in Figure 2, where the arm is shown in its zero configuration. The link i coordinate frame is denoted by $\mathcal{F}_i$, with coordinate axes $(\widehat{x}_i, \widehat{y}_i, \widehat{z}_i)$ and origin O_i. The associated interlink transformation matrices, ${}^{i-1}T_i$, $i = 1, \cdots 7$, are easily found from the above expression evaluated for the D–H parameter values listed in Table 1. If the link length parameters a_i, $i = 1, \cdots, 6$ are set to zero, the 7R anthropomorphic arm described in [2] is retrieved and we call this arm the "zero–offset" arm.

The forward kinematic function, ${}^0T_7$, which gives the position and orientation of the end-effector as a function of joint angles $\theta = (\theta_1, \cdots, \theta_7)^T$, is ${}^0T_7 = {}^0T_1 \cdots {}^6T_7$. If these multiplications are performed to obtain a symbolic form for ${}^0T_7$, the resulting expression will be complex due to the multitude of nonzero link offsets and the fact that no two consecutive joint axes are parallel. Rather than construct and implement the symbolic expression, it is more efficient to compute the forward kinematic function ${}^0T_7$ via a link-by-link iteration of the form

$$ {}^0T_i = {}^0T_{i-1} \cdot {}^{i-1}T_i \,, \quad i = 1, \cdots, 7 \tag{1} $$

thus exploiting special structural properties of the homogeneous transformation matrices during each link update. Furthermore, it is useful to explicitly have the interlink homogeneous transformations, ${}^{i-1}T_i$, since important quantities — such as the vectors w, e, and p defined later — can then be computed. In fact, such quantities are often a direct result of the intermediate steps of the iteration (1).

2.2. Mapping from Joint-Space to Elbow Angle

When the arm is in a kinematically nonsingular configuration, there will generally exist one excess joint degree-of-freedom for the task of end-effector control since there are seven joint angles available to orient and position the end-effector — a task which requires only six degrees of freedom. As a result, for a fixed end-effector configuration there is generally a one-dimensional subset of joint space (a "self-motion manifold") which maps to this configuration. Actually, there are finitely many, up to 16 in the most general case, such manifolds or "poses" [3, 4]. The extra degree of freedom represented by a self-motion manifold can be used to attain some additional task requirement, provided that this task can be performed independently of end-effector placement [5, 6]. Furthermore, the imposition of an auxiliary task constraint can provide sufficient additional information to uniquely determine the joint angles (modulo the remaining finitely many-to-one mapping property represented by the pose [3, 4]). This scalar additional task variable is denoted by ψ and is assumed to be a meaningful parameterization of the self-motion manifolds which map to a given end-effector configuration. We say that the "basic" task of end-effector placement has been *augmented* by the additional task represented by ψ. In essence, the concept of the forward kinematic map is generalized to be the (finitely many-to-one) mapping from $\theta \in R^7$ to $({}^0T_7, \psi)$.

Although ψ can be any additional scalar parameter which is independent of end-effector configuration, we define and use the "elbow angle" to resolve the manipulator redundancy. Refer to Figures 3 and 4 where $S = O_1$, $E = O_4$, and $W = O_7$ denote the origins of link frames 1, 4, and 7 respectively. ψ is defined by the angle from the vertical plane containing the shoulder-wrist line (line SW) to the shoulder-elbow-wrist plane (plane SEW) in the right-hand sense about the vector $w = W - S$. Assuming that the elbow angle ψ is a meaningful parameterization of manipulator redundancy, a self-motion is described by a rotation of the plane SEW about the line SW. Note that the elbow angle ψ is undefined when the wrist point W is anywhere on a line above the shoulder point S — even though this is generally not a singular configuration — since in this case the vertical plane is not uniquely defined. ψ is also undefined when e and w are collinear since then the plane SEW is not uniquely defined. In the latter case, the arm is either nearly fully outstretched, or folded, and is therefore near or at an "elbow singular" configuration [3, 7].

To derive the forward kinematic function which gives ψ as a function of joint angles, again consider Figure 4. Let $w = W - S$, $e = E - S$, and let $\widehat{V}$ denote the unit vector in the vertical direction of the base frame. Let the projection of e onto w be given by $d = \widehat{w}(\widehat{w}^T e)$, $\widehat{w} = w/\|w\|$. The minimum distance from the line SW to the point E is along the vector $p = e - d = (I - \widehat{w}\widehat{w}^T)e$. The vertical plane is the plane which contains both w and the vertical unit vector $\widehat{V}$. The unit vector in the vertical plane which is orthogonal to w is given by $\widehat{\ell} = \ell/\|\ell\|$, with $\ell = (w \times \widehat{V}) \times w$. We also define the unit vector $\widehat{p} = p/\|p\|$. Note that e, w, $\widehat{w}$, d, p, $\widehat{p}$, ℓ, and $\widehat{\ell}$ can be computed during the forward kinematics iteration (1) (see the discussion following equation (6) below).

The vector ℓ, or equivalently $\widehat{\ell}$, is treated as a free vector which can slide along the line SW. In particular, ℓ is moved along the line SW until its base is in contact with the base of vector p at the point d (see Figure 4), so that ψ is the angle from ℓ to p. This construction results in

$$c_\psi = \widehat{\ell}^T \widehat{p}, \quad s_\psi \widehat{w} = \widehat{\ell} \times \widehat{p}, \quad s_\psi = \widehat{w}^T(\widehat{\ell} \times \widehat{p}) \tag{2}$$

where $c_\psi = \cos\psi$ and $s_\psi = \sin\psi$. This gives

$$\tan\psi = \frac{\widehat{w}^T(\widehat{\ell} \times \widehat{p})}{\widehat{\ell}^T \widehat{p}} = \frac{\widehat{w}^T(\ell \times p)}{\ell^T p} \tag{3}$$

The result (3) can be simplified somewhat. Defining $g = w \times \widehat{v}$, we have $\ell = g \times w$, and we note that $\ell^T g = \widehat{V}^T g = 0$. This means that ℓ and $\widehat{V}$ are coplanar, both lying in the vertical plane. Since, in general, the vertical plane is spanned by $\widehat{V}$ and $\widehat{w}$, we have

$$\widehat{\ell} = \alpha\widehat{V} + \beta\widehat{w}, \quad \alpha = 1/(\widehat{\ell}^T\widehat{V}), \quad \beta = -\alpha\widehat{w}^T\widehat{V} \tag{4}$$

Substituting this result into (2) gives

$$c_\psi = \alpha\widehat{V}^T\widehat{p}, \quad s_\psi = \alpha\widehat{w}^T(\widehat{V} \times \widehat{p})$$

which can be used with (3) to obtain

$$\tan\psi = \frac{\widehat{w}^T(\widehat{V} \times \widehat{p})}{\widehat{V}^T\widehat{p}} = \frac{\widehat{w}^T(\widehat{V} \times p)}{\widehat{V}^T p} \tag{5}$$

Equation (5) immediately gives the forward kinematic function which maps the joint angles θ to the elbow angle ψ:

$$\psi = \operatorname{atan2}(\widehat{w}^T(\widehat{V} \times p), \widehat{V}^T p) \tag{6}$$

Note that (6) is undefined when both arguments are simultaneously zero. This occurs when the arm is in a configuration for which e and w are collinear, or for which the wrist point W is directly above the shoulder point S on the line through $\widehat{V}$. These indeterminacies are discussed above, and are due to the inability to uniquely define the elbow plane SEW or the reference vertical plane, respectively.

The augmented forward kinematics mapping $\theta \to ({}^0T_7, \psi)$ is given by (1) and (6). The quantities $\widehat{w}$ and $p = e - d = (I - \widehat{w}\widehat{w}^T)e$ are first computed during the iteration (1), after which ψ is computed by (6). Note that, with

$${}^0T_4 = \begin{pmatrix} {}^0R_4 & {}^0P_4 \\ 0^T & 1 \end{pmatrix} \quad \text{and} \quad {}^0T_7 = \begin{pmatrix} {}^0R_7 & {}^0P_7 \\ 0^T & 1 \end{pmatrix}$$

quantities which are directly computed during the iteration (1), the representations of e and w in the base (link 0) frame $\mathcal{F}_0$ are precisely ${}^0e = {}^0P_4$ and ${}^0w = {}^0P_7$. Also note that $\widehat{V}$ is a constant vector which is usually expressed in a frame which gives it a particularly simple form such as $(0,0,1)^T$ or $(1,0,0)^T$.

3. Differential Kinematics

3.1. Manipulator End–Effector Jacobian, J^{ee}

To present actual values for the end–effector Jacobian, J^{ee}, it is first necessary to choose a "velocity reference point," as well as a frame in which to represent the vectorial quantities which define the columns of the Jacobian. In this section, to simplify notation, we will suppress the trailing superscript and write the end–effector Jacobian simply as $J = J^{ee}$. When a velocity reference point, a, and a representation frame, $\mathcal{F}_r$, have been chosen (as discussed immediately below), we write ${}^rJ_a = {}^rJ_a^{ee}$.

Let ω_a and v_a be the angular and linear velocities of a coordinate frame, $\mathcal{F}_a$, located at a point a and fixed with respect to the manipulator end–effector. The point a is known as a "velocity reference point" of the end–effector. The Jacobian, $J_a(\theta) \in R^{6\times7}$, relates joint rates to the frame $\mathcal{F}_a$ rate of change via the linear relationship $(\omega_a^T, v_a^T)^T = J_a(\theta)\dot{\theta}$ and is given by [8]

$$J_a = \begin{pmatrix} \hat{z}_1 & \cdots & \hat{z}_7 \\ \hat{z}_1 \times P_{a,1} & \cdots & \hat{z}_7 \times P_{a,7} \end{pmatrix} \tag{7}$$

In (7), $\hat{z}_i$ denotes the unit vector corresponding to the z-axis of link frame i (i.e. of $\mathcal{F}_i$) while $P_{a,i} \equiv P_{a,O_i} \equiv a - O_i$ is the vector from the origin, O_i, of link frame i to the point a. Note that $P_{i,i} = 0$.

Let $\mathcal{F}_b$ denote an alternative frame fixed with respect to the end–effector and located at the velocity reference point b. The relationship between joint rates and the rate of change of $\mathcal{F}_b$ is given by $(\omega_b^T, v_b^T)^T = J_b\dot{\theta}$. Let $\mathcal{F}_r$ and $\mathcal{F}_s$ be frames which are *not* necessarily fixed with respect to the end–effector. The representations of ω_a and v_a in frame $\mathcal{F}_r$ are denoted by ${}^r\omega_a$ and ${}^rv_a$. Similarly, ${}^s\omega_b$ and ${}^sv_b$ are the representations of ω_b and v_b in $\mathcal{F}_s$. Note that we have defined a and b to be end–effector reference points, i.e. to be fixed with respect to the end–effector, while we have placed no constraints on r and s.

The Jacobian, ${}^rJ_a$, giving the rate of change of $\mathcal{F}_a$ represented in $\mathcal{F}_r$, is related to ${}^sJ_b$, the Jacobian giving the rate of change of frame $\mathcal{F}_b$ represented in frame $\mathcal{F}_s$, by [1, 9]

$${}^rJ_a = \begin{pmatrix} {}^rR_s & 0 \\ 0 & {}^rR_s \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ {}^s\tilde{P}_{a,b} & I \end{pmatrix} \cdot {}^sJ_b \tag{8}$$

where, for a 3-vector x, $\tilde{x}$ denotes the 3×3 skew symmetric matrix defined by $\tilde{x}y = x \times y$ for every $y \in R^3$ and $P_{a,b} = a - b$. When $r = b$, we write ${}^bP_a \equiv {}^bP_{a,b}$. ${}^rR_s \in R^{3\times3}$ is a rotation matrix, represented in frame $\mathcal{F}_r$, which gives the orientation of frame $\mathcal{F}_s$ with respect to frame $\mathcal{F}_r$. Common choices of ${}^rJ_a$ are given by ${}^7J_7$, and ${}^0J_7$. It is straightforward to show from (8) that

$$\det {}^rJ_a(\theta)\,{}^rJ_a(\theta)^T = \det {}^sJ_b(\theta)\,{}^rJ_b(\theta)^T \tag{9}$$

for every a, b, r, and s. Since an $m \times n$ matrix M, $m < n$, is full rank if and only if $\det MM^T \neq 0$, eq. (9) shows that the singularity of a manipulator Jacobian is independent of the choice of velocity reference point and representation frame, and is a function purely of the manipulator configuration variables θ.

An important aspect of the decomposition (8) is that s and b can often be chosen to make the Jacobian matrix have a particularly simple structure for the purposes of singularity analysis, efficient evaluation, and efficient inversion. For example, in [10] an algorithm for the efficient computation of ${}^0J_0$ is given. Note that J_0 does *not* give the velocity of the base frame, $\mathcal{F}_0$, as a function of joint rates — indeed, in most cases the base is assumed fixed and the base frame origin, O_0, cannot be a velocity reference point for the moving end–effector. Instead, J_0 is viewed as giving the velocity of a reference frame fixed with respect to the end–effector and instantaneously coincident with the base frame origin, O_0. The computation of

$$ {}^0J_0 = \begin{pmatrix} {}^0\widehat{z}_1 & {}^0\widehat{z}_2 & \cdots & {}^0\widehat{z}_7 \\ {}^0\widehat{z}_1 \times {}^0P_{0,1} & {}^0\widehat{z}_2 \times {}^0P_{0,2} & \cdots & {}^0z_7 \times {}^0P_{0,2} \end{pmatrix} \tag{10} $$

where ${}^kP_{i,j} = {}^{O_k}P_{i,j}$ and $P_{i,j} = P_{O_i,O_j} = O_j - O_i$, naturally fits in with the forward kinematics iteration (1), since from

$$ {}^0T_i = \begin{pmatrix} {}^0R_i & {}^0P_i \\ 0^T & 1 \end{pmatrix} $$

${}^0P_i \equiv {}^0P_{i,0}$, we can obtain ${}^0P_{0,i} = -{}^0P_i$ and ${}^0\widehat{z}_i \times {}^0P_{0,i}$ where ${}^0\widehat{z}_i = {}^0R_ie_3$, $e_3 = (0,0,1)^T$. Having ${}^0J_0$, ${}^0J_7$ can then be found from (see (8))

$$ {}^0J_7 = \begin{pmatrix} I & 0 \\ {}^0\tilde{P}_7 & I \end{pmatrix} {}^0J_0 \tag{11} $$

The symbolic forms of ${}^0J_0$ and ${}^0J_7$ can be found from this procedure, but these expressions are complex and provide little insight.

In [9], the results in [10] are extended to show that taking $s = O_i$ and $b = O_j$ for an appropriate choice of link frames i and j can result in an expression ${}^iJ_j \equiv {}^{O_i}J_{O_j}$ which is not only efficient to compute, but which simplifies singularity analysis and (for nonredundant manipulators) inversion. In particular, to gain insight into the singularity structure of the K–1207 end–effector Jacobian (and to obtain alternative ways of constructing ${}^0J_0$ and ${}^0J_7$) we will let $b = 3$ (i.e., let the velocity reference point be the origin of link frame 3) and $s = 3$ (let the reference frame be link frame 3) in (9) to arrive at an expression for ${}^3J_3$. J_3 should be interpreted as giving the velocity of a fictitious tool frame which is instantaneously coincident with link frame 3. ${}^3J_3$ is found from eq. (7) by taking $P_{a,i} = P_{3,i} = P_{O_3,O_i} = O_3 - O_i$ and representing $\widehat{z}_i$ and $P_{3,i}$ in link frame 3 to obtain ${}^3\widehat{z}_i$ and ${}^3\widehat{z}_i \times {}^3P_{3,i}$, ${}^3P_{3,i} = {}^{O_3}P_{O_3,O_i}$. The symbolic expression for ${}^3J_3$ found in this manner is given by

$$ {}^3J_3 = \left(\begin{matrix} -S_2C_3 & S_3 & 0 & 0 & S_4 \\ S_2S_3 & C_3 & 0 & 1 & 0 \\ C_2 & 0 & 1 & 0 & C_4 \\ d_3S_2S_3 + (a_2C_2 + a_1)S_5 & d_3C_3 & 0 & 0 & 0 \\ (d_3S_2 + a_2C_2 + a_1)C_3 & -d_3S_3 & 0 & 0 & -a_3C_4 - a_4 \\ 0 & -a_2 & 0 & a_3 & 0 \end{matrix} \right. $$
$$ \left. \begin{matrix} -C_4S_5 & C_4C_5S_6 + S_4C_6 \\ C_5 & S_5S_6 \\ S_4S_5 & C_4C_6 - S_4C_5S_6 \\ S_4(a_4C_5 + a_5) - d_5C_4C_5 & S_5[C_4(a_5C_6 - d_5s_6 + a_6) + a_4S_4S_6] \\ -S_5[a_3S_4 + d_5] & C_5[S_6(a_3S_4 + d_5) - a_6] - (a_5C_5 + a_4 + a_3C_4)C_6 \\ C_4(a_4C_5 + a_5) + C_5(d_5S_4 + a_3) & S_5[(a_4C_4S_6 + a_3S_6) + S_4(d_5S_6 - a_5C_6 - a_6)] \end{matrix} \right) \tag{12} $$

Having ${}^3J_3$, ${}^0J_0$ is found from (see eq. (8))

$$ {}^0J_0 = \begin{pmatrix} I & 0 \\ {}^0\tilde{P}_3 & I \end{pmatrix} \cdot {}^0J_3 = \begin{pmatrix} I & 0 \\ {}^0\tilde{P}_3 & I \end{pmatrix} \cdot \begin{pmatrix} {}^0R_3 & 0 \\ 0 & {}^0R_3 \end{pmatrix} \cdot {}^3J_3 \tag{13} $$

with ${}^0P_3 \equiv {}^0P_{3,0}$ given by

$$ {}^0P_3 = \begin{pmatrix} C_1(d_3S_2 + a_2C_2 + a_1) \\ S_1(d_3S_2 + a_2C_2 + a_1) \\ d_3C_2 - a_2S_2 \end{pmatrix} \tag{14} $$

and ${}^0R_3$ by

$$ {}^0R_3 = \begin{pmatrix} C_1C_2C_3 - S_1S_3 & -C_1C_2S_3 - S_1C_3 & C_1S_2 \\ C_1S_3 + S_1C_2C_3 & C_1C_3 - S_1C_2S_3 & S_1S_2 \\ -S_2C_3 & S_2S_3 & C_2 \end{pmatrix} \tag{15} $$

The relative simplicity of (12) not only enables one to efficiently compute ${}^0J_7$ via eqs. (11)-(15), but also allows one to gain insight into conditions leading to Jacobian singularity. In the special case of the zero-offset arm discussed in [2], corresponding to $a_1 = \cdots = a_6 = 0$, (12) simplifies to

$$ {}^3J_3 = \begin{pmatrix} -S_2C_3 & S_3 & 0 & 0 & S_4 & -C_4S_5 & C_4C_5S_6 + S_4C_6 \\ S_2S_3 & C_3 & 0 & 1 & 0 & C_5 & S_5S_6 \\ C_2 & 0 & 1 & 0 & C_4 & S_4S_5 & C_4C_6 - S_4C_5S_6 \\ d_3S_2S_3 & d_3C_3 & 0 & 0 & 0 & -d_5C_4C_5 & -d_5C_4S_5S_6 \\ d_3S_2C_3 & -d_3S_3 & 0 & 0 & 0 & -d_5S_5 & d_5C_5S_6 \\ 0 & 0 & 0 & 0 & 0 & d_5S_4C_5 & d_5S_4S_5S_6 \end{pmatrix} \tag{16} $$

3.2. Elbow Angle Jacobian, J^ψ, and the Augmented Jacobian, J^A

Let the relationship between the rate of change of a scalar additional task variable, ψ, and the joint rates be given by $\dot\psi = J^\psi\dot\theta$. The "augmented" Jacobian is given by

$$ J^A = \begin{pmatrix} J^{ee} \\ J^\psi \end{pmatrix} $$

where J^{ee} is the end-effector Jacobian discussed in Section 3.1. For the task of positioning and orienting the end-effector *augmented* by an additional task represented by ψ, the augmented Jacobian relates joint rates to the simultaneous rates of change of the end-effector and ψ. Given the end-effector Jacobian, J^{ee}, the augmented Jacobian J^A is obtained once J^ψ has been determined for a given task variable ψ. In this section, J^ψ is constructed for the case where ψ describes the angle between the vertical plane and the elbow plane SEW as defined in Section 2.

Before proceeding, it is necessary to define the Jacobians $\mathbf{E}$ and $\mathbf{W}$ which relate joint rates to $\dot e$ and $\dot w$ respectively via $\dot e = \mathbf{E}\dot\theta$ and $\dot w = \mathbf{W}\dot\theta$, where e and w are defined in Section 2.2. $\dot e$ is the linear velocity of the manipulator elbow point $E = O_4$, and $\dot w$ is the linear velocity of the wrist point $W = O_7$. We have

$$ \mathbf{E} = (\,\hat z_1 \times P_{4,1}\,, \quad \hat z_2 \times P_{4,2}\,, \quad \hat z_3 \times P_{4,3}\,, \quad 0, \quad \cdots, \quad 0\,) \tag{17} $$

$$ \mathbf{W} = (\,\hat z_1 \times P_{7,1}\,, \quad \cdots, \quad \hat z_6 \times P_{7,6}\,, \quad 0\,) \tag{18} $$

where $P_{i,j} = O_j - O_i$. Note that eqs. (17) and (18) are given in coordinate-free form and that to provide values for $\mathbf{E}$, or $\mathbf{W}$, a choice of reference frame for representing $\hat z_j$ and $P_{i,j}$ must be made. Also note that (compare eqs. (7) and (18)) $J_7^{ee} = \begin{pmatrix} \cdots \\ \mathbf{W} \end{pmatrix}$, so that any procedure for producing a value for $J_7 = J_7^{ee}$ (such as the one discussed following eq. (10)) automatically results in a value for $\mathbf{W}$. Furthermore, just as one can construct ${}^0\mathbf{W}$ from knowledge of ${}^{i-1}T_i$, $i = 1, \cdots, 7$ (say in the manner discussed after eq. (10)), one can readily compute values for **E** given the interlink homogeneous transformations ${}^{i-1}T_i$.

Recall the definitions of ℓ, $\widehat{\ell}$, $\widehat{V}$, p, $\widehat{p}$, w, $\widehat{w}$, and e given in Section 2.2. Also recall, as discussed in Section 2.2, that these quantities can all be computed from knowledge of the interlink homogeneous transformations ${}^{i-1}T_i$.

Lemma 3.1: The relationship between $\dot{\theta}$ and $\dot{\psi}$, where ψ is the elbow angle as defined in Section 2.2, is given by

$$\dot{\psi} = \frac{1}{\|p\|}(\widehat{w} \times \widehat{p})^T \dot{p} - \frac{1}{\|\ell\|}(\widehat{w} \times \widehat{\ell})^T \dot{\ell} \tag{19a}$$

$$= \frac{(\widehat{w} \times \widehat{p})^T}{\|p\|}\left\{\mathbf{E} - \frac{\widehat{w}^T e}{\|w\|}\mathbf{W}\right\}\dot{\theta} + \frac{\widehat{V}^T w}{\|\ell\|}(\widehat{w} \times \widehat{\ell})^T \mathbf{W}\dot{\theta} \tag{19b}$$

which results in

$$J^{\psi} = \frac{(\widehat{w} \times \widehat{p})^T}{\|p\|}\mathbf{E} + \left\{\frac{\widehat{V}^T w}{\|\ell\|}(\widehat{w} \times \widehat{\ell})^T - \frac{\widehat{w}^T e}{\|w\|\,\|p\|}(\widehat{w} \times \widehat{p})^T\right\}\mathbf{W} \tag{20}$$

●

Since the elbow angle ψ is given by the angle from ℓ to p, it is natural that $\dot{\psi}$ should depend only on $\dot{\ell}$ and $\dot{p}$ as in eq. (19a). Equation (19a) says that only the components of $\dot{\ell}$ and $\dot{p}$ which result in an instantaneous motion of ℓ and p directly towards or away from each other can produce a change in the elbow angle, ψ. Based on our earlier discussions, it should be obvious that J^{ψ} can be constructed from knowledge of the interlink homogenous transformations ${}^{i-1}T_i$. Also note that J^{ψ} is independent of the reference frame chosen to represent the quantities in the right hand side of eq. (20).

4. Conclusions

In this paper the forward kinematic functions which give end–effector configuration and elbow angle as a function of joint angles for the Robotics Research Model K–1207 manipulator have been derived. Also given is the augmented Jacobian which relates joint rates to end–effector and elbow angle rates. Omitted derivations can be found in a longer and more complete version of this paper available from the authors. The fuller version of this paper also contains a detailed singularity analysis of the augmented Jacobian.

Acknowledgement

This research was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The suggested use of intermediate Jacobians by Dr. Amir Fijany is gratefully acknowledged.

References

[1] J.J. Craig, *Introduction to Robotics*, Addison–Wesley, Reading, Mass., 1986.

[2] J.M. Hollerbach, "Optimum Kinematic Design for a Seven Degree of Freedom Manipulator," 2^{nd} *Int. Symp. Robotics Research*, Kyoto, Japan, Aug. 20–23, 1984, pp. 215–222.

[3] J.W. Burdick, "Kinematic Analysis and Design of Redundant Robot Mechanisms," Ph.D. Thesis, Dept. of ME, Stanford, March 1988.

[4] J.W. Burdick and H. Seraji, "Characterization and Control of Self-Motions in Redundant Manipulators," *NASA Conference on Space Telerobotics*, January 1989, Pasadena.

[5] H. Seraji, "A New Approach to Global Control of Redundant Manipulators," *NASA Conference on Space Telerobotics*, Jan. 31–Feb. 2, 1989, Pasadena, California.

[6] O. Egeland, "Task-Space Tracking with Redundant Manipulators," *IEEE J. Robotics and Automation*, Vol. RA-3, 1987, pp. 471–475.

[7] C.W. Wampler, II, "Inverse Kinematics of a 7 DOF Manipulator," NATO Advanced Research Workshop on Robots with Redundancy: Design, Sensing, and Control, June 27–July 1, 1988, Lago di Garda, Italy.

[8] H. Asada and J.-J.E. Slotine, *Robot Analysis and Control*, Wiley–Interscience, New York, 1986.

[9] A. Fijany and A.K. Bejczy, "Efficient Jacobian Inversion for the Control of Simple Robot Manipulators," *1988 IEEE Int. Conf. Robotics and Automation*, Philadelphia, 1988, pp. 999–1007.

[10] D.E. Orin and W.W. Schrader, "Efficient Computation of the Jacobian for Robot Manipulators," *Int. J. Robotics Research*, Vol. 3, 1984, pp. 66–75.

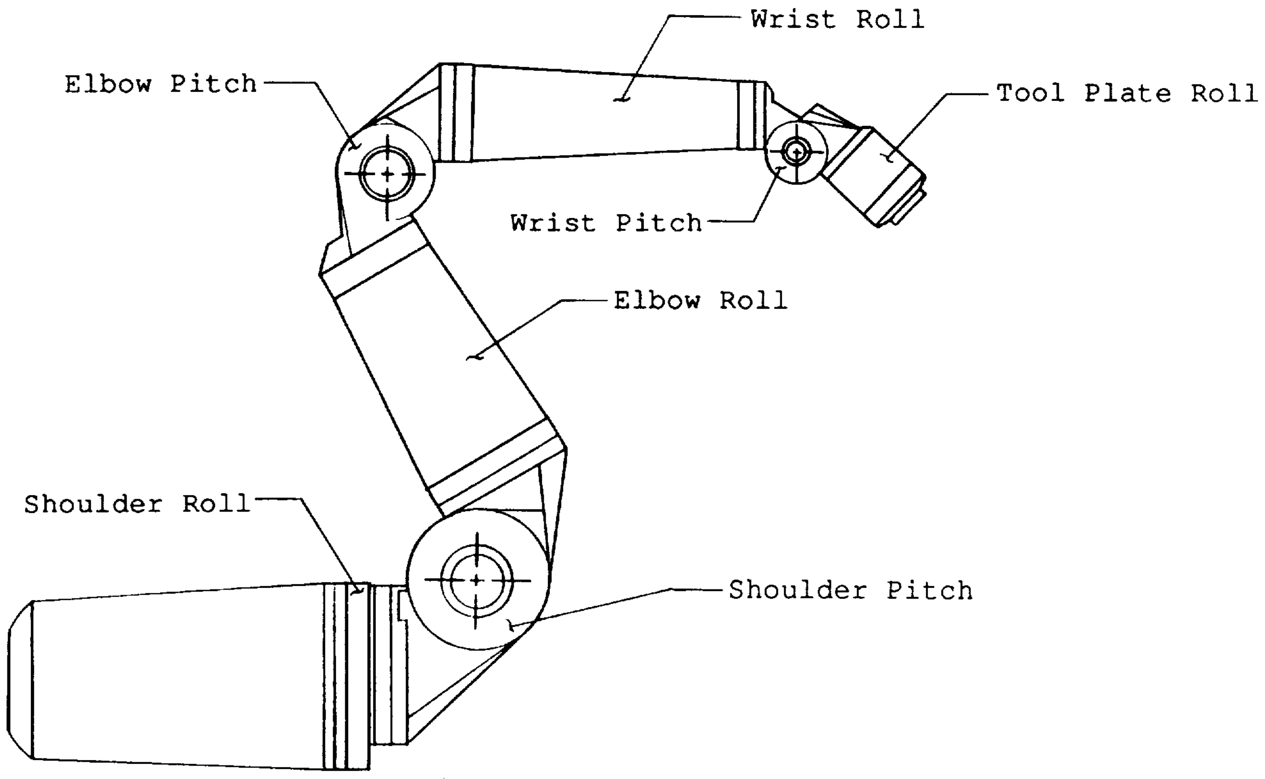


FIGURE 1: Robotics Research Model K-1207 Arm

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0°	0	0	θ_1
2	-90°	a_1	0	θ_2
3	+90°	a_2	d_3	θ_3
4	-90°	a_3	0	θ_4
5	+90°	a_4	d_5	θ_5
6	-90°	a_5	0	θ_6
7	+90°	a_6	0	θ_7

$a_1 = +4.850$ in $= +12.319$ cm
$a_2 = -4.250$ in $= -10.795$ cm
$a_3 = -3.125$ in $= -\ 7.938$ cm
$a_4 = +3.125$ in $= +\ 7.938$ cm
$a_5 = -1.937$ in $= -\ 4.920$ cm
$a_6 = +1.937$ in $= +\ 4.920$ cm
$d_3 = d_5 = 21.5$ in $= 54.61$ cm

TABLE 1: Denavit-Hartenberg Parameters of the K-1207 Arm

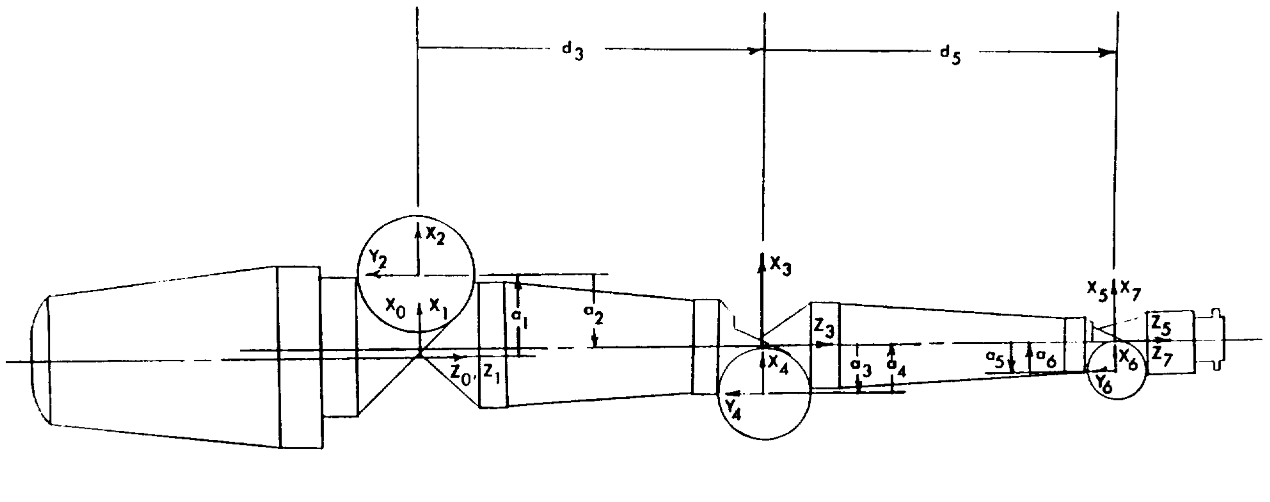


FIGURE 2: Robotics Research Model K-1207 Link Frame Assignment

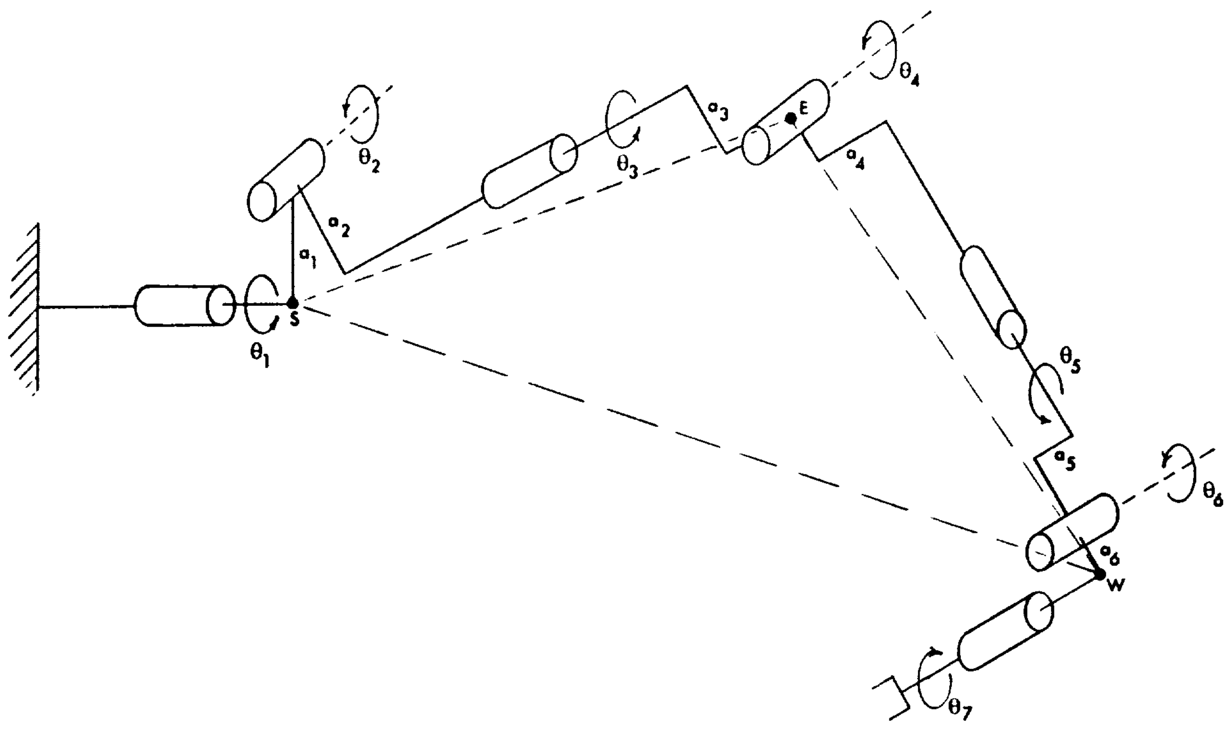


FIGURE 3: Definition of the Elbow Plane SEW

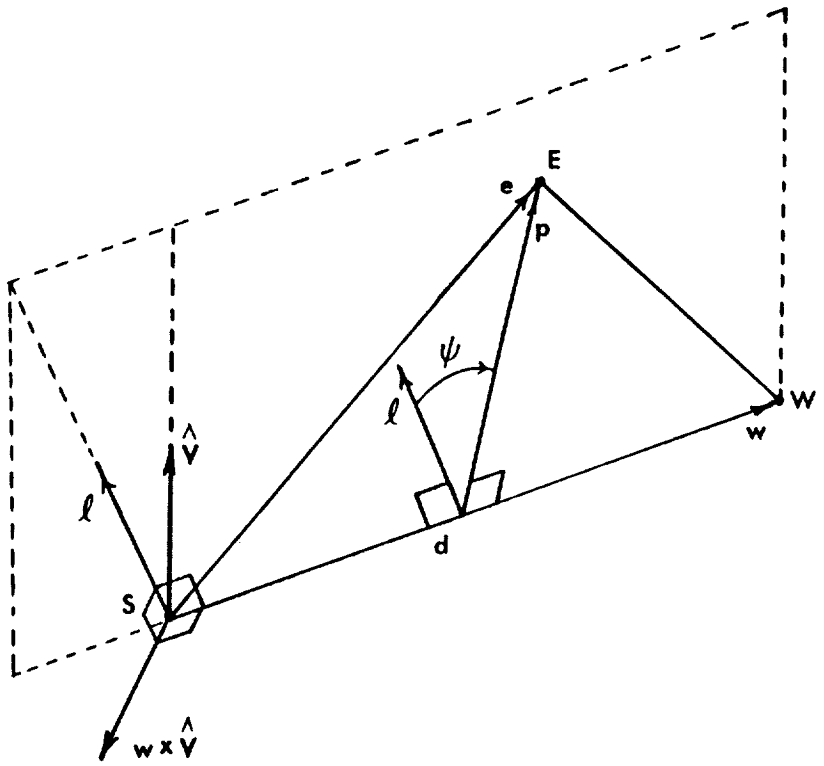


FIGURE 4: Definition of the Elbow Angle ψ

N90-29004

CARTESIAN CONTROL OF REDUNDANT ROBOTS

R. Colbaugh **K. Glass**
New Mexico State University, Las Cruces, NM. 88003

Abstract

This paper presents a Cartesian-space position/force controller for redundant robots. The proposed control structure partitions the control problem into a nonredundant position/force trajectory tracking problem and a redundant mapping problem between Cartesian control input $F \in R^m$ and robot actuator torque $T \in R^n$ (for redundant robots, $m < n$). The underdetermined nature of the $F \longrightarrow T$ map is exploited so that the robot redundancy is utilized to improve the dynamic response of the robot. This dynamically optimal $F \longrightarrow T$ map is implemented locally (in time) so that it is computationally efficient for on-line control; however, it is shown that the map possesses globally optimal characteristics. Additionally, it is demonstrated that the dynamically optimal $F \longrightarrow T$ map can be modified so that the robot redundancy is used to simultaneously improve the dynamic response and realize any specified kinematic performance objective (e.g., manipulability maximization or obstacle avoidance). Computer simulation results are given for a four degree of freedom planar redundant robot under Cartesian control, and demonstrate that position/force trajectory tracking and effective redundancy utilization can be achieved simultaneously with the proposed controller.

1. INTRODUCTION

It is predicted that in the near future robot manipulators will be required to perform complex tasks that demand great dexterity and versatility in both position control and force control applications. Such tasks will require performance superior to that obtainable with conventional six degree of freedom (DOF) robots under the control of joint-space position servo loops. This fact has motivated increased research activity in the area of redundant robot manipulators. Redundant robots possesses more DOF than are necessary to achieve the desired position and orientation of the end-effector, and it is expected that the "extra" degrees of freedom can be used to improve the robot's performance.

Most of the research on the control of redundant robots reported to date has focused on the inverse kinematics problem, which involves the calculation of the joint-space trajectory that provides the desired end-effector motion and in addition satisfies some side criterion. The majority of this work has involved using redundancy to realize some kinematic performance objective. A partial list of kinematic performance criteria that have been studied includes singularity avoidance [1,2], obstacle avoidance [3,4], joint limit avoidance [5,6], repetitive motion conservation [6,7], and achievable accuracy [8]. Research in which manipulator redundancy is utilized to achieve a dynamic performance objective has been more limited, and includes studies of minimizing joint torque requirements [9,10], minimizing manipulator energy consumption [11,12], and increasing the robot's dynamic response [13,14].

It has only been very recently that researchers have considered the complete redundant robot control problem [14-19]. The controllers described in [14-16] are model-based control schemes which require complete knowledge and calculation of the complex robot dynamic model. In addition, each of these control algorithms requires either explicit or implicit calculation of the inverse kinematics of the robot. Alternatively, the control strategy presented in [17-19] is an adaptive Cartesian-space control algorithm which does not require calculation of either the robot dynamic model or the inverse kinematics, and which has been shown through simulations and experiments to perform well. However, this controller has thusfar been applied only to control problems in which the redundancy is utilized to realize kinematic performance objectives.

This paper presents an adaptive Cartesian position/force controller for redundant robots. The proposed control strategy is to partition the control problem into a nonredundant trajectory tracking problem and a redundant mapping problem between Cartesian control input $F \in R^m$ and robot actuator torque $T \in R^n$ (for redundant robots, $m < n$). The underdetermined nature of the $F \longrightarrow T$ map is exploited to allow the redundancy to be effectively utilized directly in Cartesian-space. Computer simulation results are given for a four DOF planar redundant robot under the control of the proposed algorithm, and demonstrate that accurate position/force trajectory tracking and effective utilization of redundancy can be achieved simultaneously with the controller.

The paper is organized as follows. In Section 2, the redundant robot position/force control problem is formulated in the partitioned form indicated above. The $F \longrightarrow T$ map which uses the robot redundancy to increase the robot's dynamic response is constructed in Section 3. This dynamically optimal $F \longrightarrow T$ map is modified in Section 4 so that the robot redundancy can be used to simultaneously improve the dynamic response and realize any specified kinematic performance objective. The performance of the controller is illustrated in Section 5 through a computer simulation study. Section 6 summarizes the paper and draws some conclusions.

2. PROBLEM FORMULATION

2.1 Basic Theory

Consider an n DOF robot manipulator performing tasks in an m-dimensional Cartesian-space (with $m < n$). These tasks will, in general, involve motion of the robot end-effector in certain directions and exertion of force by the end-effector on the environment in the remaining directions. The particular directions of motion and force exertion depend on the nature of the task. Consider now a task-related "constraint frame" which is defined by the particular end-effector/environment contact situation [20]. In the constraint frame, the m-dimensional Cartesian-space can be decomposed into an l-dimensional "position subspace" and a j-dimensional "force subspace", where $l + j = m$ and where the position subspace and force subspace are orthogonal. The position subspace contains the l directions in which the robot end-effector is free to move and along which the end-effector position is to be controlled, while the force subspace contains the j directions in which the robot end-effector is constrained by the environment and along which the contact force is to be controlled. For convenience it will be assumed in the following that all quantities are expressed in terms of the constraint frame unless otherwise noted.

Let $y \in R^m$ define the position (and orientation) of the end-effector in Cartesian-space and $\theta \in R^n$ be the vector of joint coordinates. The relationship between end-effector position y and joint-space position θ is

$$y = f(\theta) \tag{1}$$

$$\dot{y} = J(\theta)\dot{\theta} \tag{2}$$

where $f : R^n \rightarrow R^m$ represents the forward kinematics of the robot and $J = \partial f/\partial\theta \in R^{m\times n}$ is the end-effector Jacobian matrix. It may be assumed without loss of generality that the elements of y are ordered such that $y = [x^T \mid z^T]^T$, where $x \in R^l$ and $z \in R^j$ are the end-effector position (and orientation) vectors in the position subspace and force subspace, respectively. Given this partitioning for y, the following partitioning for J may be defined:

$$\begin{bmatrix} \dot{x} \\ -- \\ \dot{z} \end{bmatrix} = \begin{bmatrix} J_p(\theta) \\ -- \\ J_f(\theta) \end{bmatrix} \dot{\theta}$$

where $J_p \in R^{l\times n}$ and $J_f \in R^{j\times n}$ are termed the "position subspace Jacobian" and "force subspace Jacobian", respectively. The dynamic model of the robot with its end-effector in contact with the environment may be written in joint-space as [e.g., 21]

$$T = H(\theta)\ddot{\theta} + V_{cc}(\theta,\dot{\theta}) + V_f(\theta,\dot{\theta}) + G(\theta) + J_f^T(\theta)P \tag{3a}$$

where $T \in R^n$ is the vector of actuator torques and/or forces, $H \in R^{n\times n}$ is the robot inertia matrix, $P \in R^j$ is the end-effector/environment contact force and/or moment, and $V_{cc}, V_f, G \in R^n$ represent the torque vectors due to Coriolis and centripedal acceleration, friction, and gravity, respectively. Alternatively, the robot dynamic model can be expressed in Cartesian-space as [e.g., 16]

$$F = (JH^{-1}J^T)^{-1}[\ddot{y} - \dot{J}\dot{\theta}] + (JH^{-1}J^T)^{-1}JH^{-1}[V_{cc} + V_f + G] + P^* \tag{3b}$$

where $F \in R^m$ is the generalized force vector corresponding to the generalized coordinate y, and $P^* = [\,0^T \mid P^T\,]^T \in R^m$ with the zero denoting an l-dimensional zero vector.

The general Cartesian-space position/force control problem for the redundant robot described in (1)-(3) may be considered to consist of two steps:

1.) Cartesian position/force trajectory tracking:

compute the Cartesian control input $F = [\,F_p^T \mid F_f^T\,]^T \in R^m$ required to track the desired m-dimensional position/force trajectory, where $F_p \in R^l$ is the position control input that tracks the desired end-effector position trajectory $x_d \in R^l$ and $F_f \in R^j$ is the force control input that tracks the desired end-effector/environment contact force trajectory $P_d \in R^j$

2.) $F \rightarrow T$ mapping:

compute the joint torque vector $T \in R^n$ required to realize F while simultaneously accomplishing some desired kinematic and/or dynamic performance objective.

Each of the steps will now be considered individually.

2.2 Cartesian Position/Force Trajectory Tracking

Observe that the Cartesian position/force trajectory tracking problem is nonredundant since F_p and x_d are both of dimension l and F_f and P_d are both of dimension j. Thus many different control strategies could be improvised to compute the control input F that would ensure that the dynamics (3b) tracks the desired end-effector position/force trajectory. Here the adaptive Cartesian-space position/force controller recently developed by Seraji [22] for nonredundant robots will be adopted to accomplish this trajectory tracking. This control scheme was derived from an improved Model Reference Adaptive Control (MRAC) method, and requires no knowledge of the robot dynamic model or parameter values for the robot, the payload, or the environment. As a result, the controller is robust to both model and parameter uncertainties, and is computationally fast for on-line control applications with modest computing power.

The control algorithm computes the position control input F_p as follows:

$$F_p = d_p(t) + K_{pp}(t)E_p + K_{vp}(t)\dot{E}_p + C(t)x_d + B(t)\dot{x}_d + A(t)\ddot{x}_d \tag{4}$$

where $E_p = x_d - x$ is the position tracking error, and $d_p \in R^l$ and $K_{pp}, K_{vp}, C, B, A \in R^{l\times l}$ are controller gains which are updated adaptively. The adaptation laws for these gains are provided in [22] and are not repeated here. Note that the control input F_p is computed entirely based on the observed performance of the manipulator.

The control scheme computes the force control input F_f using the following algorithm:

$$F_f = d_f(t) + K_I(t)\int_0^t E_f dt + K_{pf}(t)E_f - K_{vf}(t)\dot{z} + P_d \tag{5}$$

where $E_f = P_d - P$ is the force tracking error, and $d_f \in R^j$ and $K_I, K_{pf}, K_{vf} \in R^{j\times j}$ are controller gains which are updated adaptively. Again, the adaptation laws for these gains are provided in [22] and are not repeated here. Note that the control input F_f is computed entirely based on the observed performance of the robot.

Finally, the position control input F_p computed in (4) and the force control input F_f computed in (5) are combined to form the Cartesian control input F:

$$F = [\, F_p^T \mid F_f^T \,]^T \tag{6}$$

2.3 Redundancy Resolution Through Construction of $F \longrightarrow T$ Map

Observe that the control input F cannot physically be applied to the robot end-effector; therefore this desired control input must be mapped to an actuator torque vector **T**. The $F \longrightarrow T$ mapping problem is underdetermined since $F \in R^m$ and $T \in R^n$ with $m < n$, so that it is at this stage of the control problem that the robot redundancy may be utilized to improve the robot's performance.

The problem of constructing an appropriate $F \longrightarrow T$ map may be formulated in terms of inverting the known $T \longrightarrow F$ map, which is unique even for redundant robots. The $T \longrightarrow F$ map may be shown to be [13,16]

$$F = (JH^{-1}J^T)^{-1}JH^{-1}T \equiv M(\theta)T \tag{7}$$

where it is easily verified that $M \in R^{m\times n}$. Inversion of the $T \longrightarrow F$ map (7) may be achieved in two ways:

1.) "direct" inversion of (7) using the theory of generalized inverses [23]
2.) "indirect" inversion of (7) by first augmenting both M and F with $r = n - m$ additional rows and then inverting the resulting fully determined system by standard methods

Each of these approaches is now briefly summarized. Additional details concerning each inversion method are provided in Sections 3 and 4 of this paper, and also in [13,24].

The direct approach to inverting (7) has proven useful for realizing dynamic performance objectives, primarily because inverting (7) using generalized inverse theory readily permits optimization of objective functions involving joint torque T and joint accelerations $\ddot{\theta}$. For example, the $F \longrightarrow T$ map which minimizes the norm of the joint torque vector $||T|| = (T^T T)^{1/2}$, subject to the constraint (7), may be easily derived using generalized inverses:

$$T = H^{-1}J^T(JH^{-2}J^T)^{-1}JH^{-1}J^TF \tag{8}$$

The indirect approach to inverting (7) has been utilized principally for realizing kinematic performance objectives. While the idea of augmenting M and F with r additional rows to obtain a fully determined system is conceptually simple, selecting these additional rows in such a way that some desired performance objective is realized is more difficult. The process of augmenting M in an appropriate manner can be simplified somewhat by choosing to augment J instead. Let $J_a = [J^T \mid J_c^T]^T \in R^{n\times n}$ be the matrix that results from augmenting J with $J_c \in R^{r\times n}$. Then, provided J_a is nonsingular, replacing J with J_a in (7) allows this $T \longrightarrow F$ map to be inverted by standard methods, yielding

$$T = J^T F + J_c^T F_c \tag{9}$$

where $F_c \in R^r$ is an appropriately chosen vector used to augment F. One method of specifying the terms J_c and F_c in (9) has been derived by Seraji [17], and is summarized in Section 4 of this paper.

3. OPTIMIZATION OF THE DYNAMIC RESPONSE OF THE ROBOT

One of the advantages of a redundant robot is the potential to use the "extra" DOF to improve the robot's dynamic response [13,14,25]. One approach to achieving this improved performance is to devise a strategy for allocating motion among the robot joints in such a way that the desired end-effector motion is tracked with minimum actuator torque. This strategy will increase the bandwidth of the robot for a given set of actuator torque limits, which in turn will lead to improved tracking of both position and force trajectories [26].

Local minimization of the (norm of the) joint torque vector required to provide the desired Cartesian control input F is achieved in Section 2.3 using the direct approach to inverting the $T \longrightarrow F$ map (7), and the result is given in (8). However, it has been found in previous investigations that local (in time) minimization of joint torques often leads to trajectories that are globally unstable [9,27]; thus implementation of the $F \longrightarrow T$ map (8) may be undesirable.

An alternative approach to reducing joint torque requirements is to consider the following constrained optimization problem:

$$\text{minimize} \quad \int_0^{t_f} \frac{1}{2}\dot{\theta}^T H\dot{\theta}dt \qquad \text{subject to the constraint} \quad y = f(\theta) \tag{10}$$

where t_f is the trajectory completion time. Note that (10) is a global optimization problem, and therefore its solution should possess the desirable characteristics of a globally optimal solution, such as trajectory stability. Note also that minimizing total robot kinetic energy integrated over the entire trajectory, subject to the constraint of desired end-effector motion, should lead to a uniform reduction in joint torques and a corresponding uniform increase in dynamic response. The optimization problem (10) may be analyzed using the calculus of variations [28]. First, the intermediate function $L(\theta, \dot{\theta}, \lambda) \in R$ is constructed:

$$L = \frac{1}{2}\dot{\theta}^T H \dot{\theta} + \lambda^T(t)[y - f(\theta)] \tag{11}$$

where $\lambda \in R^m$ is the Lagrange multiplier vector. The necessary conditions on (11) for optimality of (10) are

$$\frac{\partial L}{\partial \lambda} = 0 \quad , \quad \frac{\partial L}{\partial \theta} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} = 0 \tag{12}$$

Substituting (11) into the necessary conditions (12) yields, after some simplification,

$$\ddot{y} - J\ddot{\theta} - \dot{J}\dot{\theta} = 0 \quad , \quad H\ddot{\theta} + \dot{H}\dot{\theta} - J^T\lambda - \frac{1}{2}\partial(\dot{\theta}^T H \dot{\theta})/\partial\theta = 0 \tag{13}$$

The equations (13) may be solved for $\ddot{\theta}$ [13]:

$$\ddot{\theta} = H^{-1}J^T(JH^{-1}J^T)^{-1}[\ddot{y} - \dot{J}\dot{\theta}] - [I_n - H^{-1}J^T(JH^{-1}J^T)^{-1}J]H^{-1}V_{cc} \tag{14}$$

where $I_n \in R^{n \times n}$ is the identity matrix. Note that the solution (14) to the problem (10) has been obtained, independently, by Kazerounian and Wang [29].

Expressing the necessary condition for optimizing (10) as an $F \longrightarrow T$ map may be achieved by substituting the joint-space dynamic model (3a) and the Cartesian-space dynamic model (3b) into the necessary condition (14), and simplifying the result:

$$T = J^T F + [I_n - J^T(JH^{-1}J^T)^{-1}JH^{-1}](V_f + G) \tag{15}$$

A close approximation to the global minimum kinetic energy $F \longrightarrow T$ map (15) may be obtained as follows. Observe that the operator $[I_n - J^T(JH^{-1}J^T)^{-1}JH^{-1}]$ projects the vector $V_f + G$ into the null-space of JH^{-1} (this may be verified by pre-multiplying the projection $[I_n - J^T(JH^{-1}J^T)^{-1}JH^{-1}](V_f + G)$ by JH^{-1} and noting that the result is the zero vector). In fact, it is shown in [24] that this operator projects the vector $V_f + G$ onto only a portion of the null-space of JH^{-1}, and that typically the resulting projection is small compared to the term $J^T F$. These results imply that the $F \longrightarrow T$ map

$$T = J^T F \tag{16}$$

is a close approximation to the global minimum kinetic energy $F \longrightarrow T$ map. Note that the map (16) is computationally efficient and requires no knowledge of the robot dynamic model.

In summary, it is hypothesized that utilizing the robot redundancy to construct the $F \longrightarrow T$ map which minimizes robot kinetic energy integrated over the trajectory, subject to the constraint of desired end-effector motion, will lead to a uniform reduction in joint torques and a corresponding uniform increase in dynamic response. Moreover, the resulting robot trajectory should be stable because of the globally optimal nature of this $F \longrightarrow T$ map. In view of the fact that the $F \longrightarrow T$ map (16) is a close approximation to the minimum kinetic energy map (15), and possesses the desirable features of computational efficiency and robustness to dynamic model uncertainty, it is proposed that the map (16) be employed in the control algorithm. The performance of the control scheme (4)-(6) together with the $F \longrightarrow T$ map (16) is examined through computer simulation in Section 5.

4. CONSIDERATION OF KINEMATIC PERFORMANCE OBJECTIVES

In this Section, the control algorithm (4)-(6),(16) is modified so that the robot redundancy is used to simultaneously improve the robot's dynamic response and realize any specified kinematic performance objective.

It is shown in [18] that a redundant robot may be controlled to track a desired end-effector position/force trajectory and simultaneously satisfy an r-dimensional kinematic constraint of the form

$$\psi(t) = g(\theta) \tag{17}$$

where $g : R^n \longrightarrow R^r$ and $\psi \in R^r$ defines the evolution of g. The control algorithm developed to achieve this desired performance computes the Cartesian control input F using (4)-(6), and then maps this control input to the robot actuator torque T as follows:

$$T = J^T F + \rho(\partial g/\partial\theta)^T F_c \tag{18}$$

where $\partial g/\partial\theta \in R^{r\times n}$ is the constraint Jacobian, $\rho \in R^{+}$ was implicitly defined as $\rho = 1$ in [18], and $F_c \in R^r$ is the constraint control input required to track the desired evolution of (17), denoted as $\psi_d(t)$. The constraint control input F_c is computed as

$$F_c = d_p(t) + K_{pp}(t)E_c + K_{vp}(t)\dot{E}_c + C(t)\psi_d + B(t)\dot{\psi}_d + A(t)\ddot{\psi}_d \tag{19}$$

where $E_c = \psi_d - \psi$ is the constraint tracking error, and the adaptive gains $d_p \in R^r$ and $K_{pp}, K_{vp}, C, B, A \in R^{r\times r}$ are updated based on the constraint tracking error E_c.

Observe that setting $\rho = 0$ in (18) reduces that map to the (approximate) minimum kinetic energy $F \longrightarrow T$ map (16), while setting $\rho = 1$ in (18) causes the robot redundancy to be used to closely track the kinematic constraint (17). Thus the map (18) may be viewed as a modification of the map (16) to include the potential to satisfy kinematic constraints, and the parameter ρ may be used to specify the relative importance of dynamic response and constraint tracking accuracy. In typical applications (e.g., obstacle avoidance, joint limit avoidance), the constraint (17) need not be tracked with the same accuracy as the end-effector task. Then ρ can be chosen small, and adequate constraint tracking and improved dynamic response can be achieved simultaneously. The selection of an appropriate value for ρ and the effect of this choice on the performance of the robot is quantified through example in Section 5.

The control algorithm (4)-(6), (17)-(19) provides a method for controlling a redundant robot so that end-effector position/force trajectory tracking and general kinematic constraint satisfaction are achieved simultaneously. This control scheme can be extended to include utilizing the redundancy to optimize general kinematic performance objectives. Let the general kinematic performance optimization problem be formulated as

$$\text{maximize} \quad G(\theta) \qquad \text{subject to the constraint} \quad y = f(\theta) \tag{20}$$

where $G : R^n \longrightarrow R$ may be constructed to represent a measure of any desired kinematic performance objective. The solution to (20) can be obtained using Lagrange multipliers. Let the augmented scalar objective function $G^*(\theta, \lambda)$ be defined as

$$G^*(\theta,\lambda) = G(\theta) + \lambda^T[y - f(\theta)] \tag{21}$$

where $\lambda \in R^m$ is the vector of Lagrange multipliers. The necessary conditions for optimality of (20) may be written using (21):

$$\begin{aligned} \partial G^*/\partial\lambda = 0 \quad &\Rightarrow \quad y = f(\theta) \\ \partial G^*/\partial\theta = 0 \quad &\Rightarrow \quad \partial G/\partial\theta = J^T\lambda \end{aligned} \tag{22}$$

From (22), it is seen that a necessary condition for optimality of (20) is that $\partial G/\partial\theta \in R(J^T)$. This requirement may be written concisely as

$$A\partial G/\partial\theta = 0 \tag{23}$$

where $A \in R^{r\times n}$ is any matrix whose rows form a basis for the null-space of J. This result is a direct consequence of the fact that the row-space and the null-space of any matrix are orthogonal complements. Note that (23) can alternatively be obtained using gradient projection optimization theory [30], and that this approach was first proposed for redundancy resolution by Baillieul in his "extended Jacobian" method [4]. When $-G(\theta)$ is convex, the condition (23) is both necessary and sufficient to solve (20). This is of interest because in robotics applications it is usually possible to construct $G(\theta)$ so that $-G(\theta)$ is convex.

Observe that the optimality condition (23) is an r-dimensional kinematic equality constraint of the form (17) with $g = A\partial G/\partial\theta$ and $\psi_d(t) = 0$. Therefore the control law (4)-(6), (17)-(19) can be used for simultaneous end-effector trajectory tracking and optimization of any desired kinematic objective function $G(\theta)$. Indeed, assuming that $G(\theta)$ is defined (and differentiable), specification of the kinematic equality constraint that is to be tracked to achieve this optimization requires only that an appropriate A matrix be constructed and that the calculations specified in (23) be performed. The matrix A may be constructed in several ways; one formulation for A is [13]

$$A = [-J_2^T(J_1^{-1})^T \quad | \quad I_r] \tag{24}$$

where $J_1 \in R^{m\times m}$ and $J_2 \in R^{m\times r}$ are the partitions of J defined by $J = [J_1 \ | \ J_2]$. The validity of the construction (24) for A may be verified by observing that $AJ^T = 0$ and that the row rank of A is r for all θ due to the presence of the I_r partition in (24).

Summarizing, the control algorithm (4)-(6), (17)-(19) can be extended to include utilizing the redundancy to solve the kinematic performance optimization problem (20) by setting $g = A\partial G/\partial\theta$ and $\psi_d = 0$, where $A \in R^{r\times n}$ is given in (24). The comments made previously concerning the role of the parameter ρ in the control law apply here as well, of course. The use of the control algorithm (4)-(6), (17)-(19) for the case in which the kinematic performance objective is the optimization of a kinematic objective function is illustrated in Section 5.

5. SIMULATION RESULTS

5.1 Overview of Computer Simulation Study

The Cartesian-space position/force control scheme for redundant robots given in (4)-(6), (17)-(19) is now applied to a direct-drive four-link planar robot in two computer simulation examples. The results presented here are samples selected from a comprehensive computer simulation study which was carried out to test the performance of the proposed controller. Note that the results given here are selected because they are typical of the larger study, and not because they represent the best performance obtainable with the proposed control law.

Consider the four-link robot in a horizontal plane shown in Figure 1. The robot parameters are link lengths $l_1 = l_2 = l_3 = l_4 = 1.0$ m, link masses $m_1 = m_2 = m_3 = m_4 = 10.0$ kg, and joint viscous friction coefficients $c_1 = c_2 = c_3 = c_4 = 40.0$ Nt·m·sec; the link inertias are modeled by thin uniform rods. The frictionless reaction surface is located parallel to the x-axis at $z = 0.0$ and has a stiffness of 10^4Nt/m. Note that in this example the base frame is chosen as the constraint frame, so that the position subspace and the force subspace are each of dimension one and correspond to the x and z axes, respectively. The robot dynamic model which relates joint torques $T \in R^4$ and joint angles $\theta \in R^4$ is given by

$$T = H(\theta)\ddot{\theta} + V_{cc}(\theta,\dot{\theta}) + V_f(\dot{\theta}) + J_f^T P \tag{25}$$

In the dynamic model (25) the numerical values for the inertia matrix $H \in R^{4\times4}$, Coriolis and centrifugal torque vector $V_{cc} \in R^4$ and viscous friction torque vector $V_f \in R^4$ may be found in [19]. Note that the gravity vector is orthogonal to the plane of motion of the robot, so that no gravity torques appear in (25). It must be emphasized that the dynamic model (25) is used only to simulate the robot behavior and is not used in the control law formulation.

In the simulation study, the performance of the control scheme (4)-(6), (17)-(19) is evaluated through comparison with a commonly proposed approach to redundancy resolution, the *inertia-weighted pseudoinverse approach* [9]. Specifically, the performance of the proposed controller is compared with the performance of a controller which resolves the robot redundancy as follows:

$$\ddot{\theta} = H^{-1}J^T(JH^{-1}J^T)^{-1}[\ddot{y} - \dot{J}\dot{\theta}] \tag{26}$$

To allow a meaningful comparison to be made between the control law (4)-(6), (17)-(19) and the redundancy resolution scheme (26), the inverse kinematics algorithm (26) must be implemented as an equivalent $F \longrightarrow T$ map. This equivalent $F \longrightarrow T$ map may be derived using the same approach taken when rewriting the inverse kinematics algorithm (14) as the equivalent $F \longrightarrow T$ map (15), and yields the following result:

$$T = J^T F + [I_n - J^T(JH^{-1}J^T)^{-1}JH^{-1}](V_{cc} + V_f) \tag{27}$$

The $F \longrightarrow T$ map (27) may be combined with the control scheme (4)-(6) to yield a pseudoinverse-based position/force controller; this controller resolves the robot redundancy exactly as prescribed in the inverse kinematics algorithm (26). Note that in deriving the $F \longrightarrow T$ map (27) it is implicitly assumed that $G = 0$, since this is the case in the simulation study.

We now turn to the discussion of two computer simulation examples. Throughout this discussion, the control law (4)-(6), (17)-(19) will be referred to as the *proposed controller* while the control scheme given by (4)-(6) together with the $F \longrightarrow T$ map (27) will be called the *weighted pseudoinverse (WP) controller*. Additionally, in these simulations, the unit of length is meter, the unit of angle is radian, the unit of force is Newton, and the unit of time is second.

5.2 Simulation 1

The task requirements for this simulation are to have the robot end-effector track a straight-line position/constant force trajectory while utilizing the redundancy to improve the dynamic response of the robot. The desired end-effector position trajectory is $x_d(t) = 2.0 + A_o - A_o cos\omega t$, for $t \subset [0, \pi/\omega]$ and for different values of the trajectory parameters A_o and ω. The desired end-effector /environment contact force is $P_d(t) = 10.0$, for $t \subset [0, \pi/\omega]$. The initial configuration of the robot is $\theta(0) = [\ \pi/3 \quad -2\pi/3 \quad 2\pi/3 \quad -2\pi/3\ ]^T$, and the robot is initially at rest.

The proposed controller and the WP controller each accomplishes the required position/force trajectory tracking by employing the Cartesian control algorithm (4)-(6). The desired position trajectory is tracked using a scalar version of the position control algorithm (4), and the desired force trajectory is tracked using a scalar version of the force controller (5). The position control input F_p and force control input F_f are combined to form F as prescribed in (6).

Redundancy resolution is achieved when mapping the Cartesian control input F (computed in (4)-(6)) to joint actuator torque T. The map used in the proposed controller for increasing the robot's dynamic response in a stable manner is given in (18) with $\rho = 0$. The $F \longrightarrow T$ map used by the WP controller is given in (27). The algorithm (4)-(6) together with the appropriate $F \longrightarrow T$ map is applied to the dynamic model (25) through computer simulation on a SUN 3/50 computer with a sampling period of one millisecond.

In the first simulation, the end-effector trajectory parameters are assigned the values $A_o = 0.5$ and $\omega = 0.25$. The results of the simulation are shown in Figures 2a-2c, and indicate that both controllers perform well. This is as expected, because the required end-effector motion is slow and of moderate length.

In the next simulation, the end-effector trajectory is made both longer and faster by choosing trajectory parameter values of $A_o = 0.8$ and $\omega = 1.25$. The results of the simulation are given in Figures 2d and 2e, and show that the WP controller requires much larger torques than the proposed controller, and yet achieves poorer tracking accuracy.

5.3 Simulation 2

This simulation illustrates the proposed controller's capability to use the robot redundancy to improve dynamic response and achieve a desired kinematic performance objective simultaneously. The desired end-effector position/force trajectory to be tracked in this simulation is quantified by $x_d(t) = \sqrt{3} + 0.8 - 0.8cos1.25t$ and $P_d(t) = 10.0$, for $t \subset [0, 4\pi/5]$. The kinematic performance objective to be achieved simultaneously with improved dynamic response is the maximization of the "manipulability measure" $w : R^n \rightarrow R^+$, defined by Yoshikawa as follows [1]:

$$w(\theta) = (det[JJ^T])^{1/2} \tag{28}$$

Briefly, it has been proposed by Yoshikawa [1] and others that utilizing robot redundancy to maximize manipulability may be an effective means of increasing robot dexterity and avoiding kinematic singularities.

The proposed controller and the WP controller each accomplishes the required position/force trajectory tracking by employing the Cartesian control algorithm (4)-(6), as described in Section 5.2 for Simulation 1. Redundancy resolution is achieved in these controllers when mapping the Cartesian control input F to joint torque T. The map used in the proposed controller for increasing dynamic response and manipulability simultaneously is given in (17)-(19) with $g = A\partial w/\partial\theta$ and $\psi_d(t) = 0$, where the matrix A is constructed as in (24). The parameter ρ, which specifies the relative importance of increasing dynamic response and increasing manipulability, is chosen (heuristically) as $\rho = 0.1$. A measure of how effectively this proposed controller increases dynamic response is obtained through comparison with the WP controller, which maps control input F computed in (4)-(6) to joint torque T using (27). The effectiveness of the proposed controller at increasing manipulability is evaluated by comparing the evolution of $w(\theta)$ over the trajectory to the maximum possible values for w given the end-effector trajectory specified in this simulation.

In the simulation, the algorithm (4)-(6) together with the appropriate $F \rightarrow T$ map is applied to the robot dynamic model (25) through computer simulation on a SUN 3/50 computer with a sampling period of one millisecond. It can be shown that in order to maximize manipulability by tracking the optimality condition $A\partial w/\partial\theta = 0$, it is necessary that the initial robot configuration be the maximum manipulability configuration corresponding to the initial end-effector position [24]. One method of obtaining the optimal initial configuration $\theta^*(0)$ is to integrate the differential equation

$$\dot{\theta} = [I - J^T(JJ^T)^{-1}J]\partial w/\partial\theta \tag{29}$$

until it reaches equilibrium. The starting point for the integration may be any configuration θ which places the end-effector in the desired initial position, and the equilibrium configuration of (29) is the optimal configuration θ^*. Using this algorithm, the optimal initial configuration of the robot is obtained as $\theta(0) = [\ 1.697202 \quad -1.570791 \quad -0.252815 \quad -1.570791\ ]^T$. In this simulation the robot is initially at rest.

The results of the simulation are shown in Figures 3a-3d. These results indicate that the WP controller requires much larger torques than the proposed controller, and exhibits poorer tracking accuracy. Additionally, the results show that the manipulability is very nearly maximum over the entire trajectory. Thus the proposed controller accurately tracks the required trajectory and successfully increases both the robot's dynamic response and manipulability measure over the entire trajectory.

6. CONCLUSIONS

This paper presents a Cartesian-space position/force control scheme for redundant robots. The proposed control strategy is to partition the control problem into a nonredundant position/force trajectory tracking problem and a redundant mapping problem between Cartesian control input F and robot actuator torque T. The underdetermined nature of the $F \rightarrow T$ map is exploited to allow the redundancy to be effectively utilized directly in Cartesian-space. Computer simulation results are given for a four DOF planar redundant robot under the control of the proposed algorithm, and demonstrate that accurate position/force trajectory tracking and effective utilization of redundancy can be achieved simultaneously with the controller.

7. ACKNOWLEDGEMENTS

The research described in this paper was supported by Sandia National Laboratories under contracts 06-5630 and 05-9859. The authors gratefully acknowledge the many helpful conversations on topics of this paper with Homayoun Seraji of the Jet Propulsion Laboratory and Patrick Eicker of Sandia National Laboratories.

8. REFERENCES

1. Yoshikawa, T., "Analysis and Control of Robot Manipulators with Redundancy", *Robotics Research: The First Intern. Symp.*, eds. Brady and Paul, Cambridge, MA., MIT Press, 1984, pp. 735-748
2. Mayorga, R.V. and A.K.C. Wong, "A Singularities Avoidance Approach for the Optimal Local Path Generation of Redundant Manipulators", *Proc. IEEE Intern. Conf. on Robotics and Automation*, Philadelphia, PA., April 1988, pp. 49-54
3. Maciejewski, A.A. and C.A. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", *Intern. J. Robotics Research*, Vol. 4, No. 3, 1985, pp. 109-117
4. Baillieul, J., "Avoiding Obstacles and Resolving Kinematic Redundancy", *Proc. IEEE Intern. Conf. on Robotics and Automation*, San Francisco, CA., April 1986, pp. 1698-1704
5. Liegeois, A., "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms", *IEEE Trans. Sys., Man, and Cyber.*, Vol. SMC-7, No. 12, 1977, pp. 868-871
6. Klein, C.A. and C.H. Huang, "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators", *IEEE Trans. Sys., Man, and Cyber.*, Vol. SMC-13, No. 3, 1983, pp. 245-250

7. Baillieul, J., J.M. Hollerbach and R. Brockett, "Programming and Control of Kinematically Redundant Manipulators", *Proc. 23rd Conf. on Decision and Control*, Las Vegas, NV., December 1984, pp. 768-774
8. Chiu, S.L., "Kinematic Characterization of Manipulators: An Approach to Defining Optimality", *Proc. IEEE Intern. Conf. on Robotics and Automation*, Philadelphia, PA., April 1988, pp. 828-833
9. Hollerbach, J.M. and K.C. Suh, "Redundancy Resolution of Manipulators Through Torque Optimization", *IEEE J. of Robotics and Automation*, Vol. RA-3, No. 4, 1987, pp. 308-316
10. Colbaugh, R. and K. Glass, "Real Time Resolution of Robot Manipulator Redundancy Through Joint Torque Optimization", *Proc. ISMM Intern. Symp. on Computer Applications*, Honolulu, HA., February 1988, pp. 168-172
11. Vukobratovic, M. and M. Kircanski, "A Dynamic Approach to Nominal Trajectory Synthesis for Redundant Manipulators", *IEEE Trans. Sys., Man, and Cyber.*, Vol. SMC-14, No. 4, 1984, pp. 1016-1021
12. Colbaugh, R. and K. L. Phillips, "Energy Optimal Trajectory Planning for Robots", *Space Nuclear Power Systems 1988*, Orbit Book Company, 1988
13. Colbaugh, R., "Dynamic Performance Optimization of Redundant Robot Manipulators", *Preprints 842nd Meeting of the American Mathematical Society*, Las Cruces, NM., April 1988
14. Egeland, O., "Task-Space Tracking with Redundant Manipulators", *IEEE J. of Robotics and Automation*, Vol. RA-3, No. 5, 1987, pp. 471-475
15. Hsu, P., J. Hauser and S. Sastry, "Dynamic Control of Redundant Manipulators", *Proc. IEEE Intern. Conf. on Robotics and Automation*, Philadelphia, PA., April 1988, pp. 183-187
16. Khatib, O., "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation", *IEEE J. of Robotics and Automation*, Vol. RA-3, No. 1, 1987, pp. 43-53
17. Seraji, H., "Configuration Control of Redundant Manipulators", *Proc. NATO Advanced Research Workshop on Robots with Redundancy*, June 1988
18. Seraji, H., R. Colbaugh, K. Glass and T. Lee, "Experimental and Simulation Studies of Configuration Control for Redundant Robots", submitted to 1989 IEEE Intern. Conf. on Robotics and Automation, Scottsdale, AZ., May 1989
19. Colbaugh, R., H. Seraji and K. Glass, "Obstacle Avoidance for Redundant Robots Using Configuration Control" (to appear)
20. Raibert, M. H. and J. J. Craig, "Hybrid Position/Force Control of Manipulators", *ASME J. Dyn. Sys., Measurement, and Control*, Vol. 102, No. 2, 1981, pp. 126-133
21. Tarn, T.J., A.K. Bejczy and X. Yun, "Robot Arm Force Control through System Linearization by Nonlinear Feedback", *Proc. IEEE Intern. Conf. on Robotics and Automation*, Philadelphia, PA., April 1988, pp. 1618-1625
22. Seraji, H., "Adaptive Force and Position Control of Manipulators", *Journal of Robotic Systems*, Vol. 4, No. 4, 1987, pp. 551-578
23. Lawson, C. L. and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ., 1974
24. Colbaugh, R. and K. Glass, "Cartesian Position/Force Control of Redundant Robots", *Final Report for Contract 06-5630 with Sandia National Laboratories*, Engineering Research Center, New Mexico State University, November 1988
25. Salisbury, J.K. and J.D. Abramowitz, "Design and Control of a Redundant Mechanism for Small Motion", *Proc. IEEE Intern. Conf. on Robotics and Automation*, St. Louis, MO., March 1985, pp. 323-328
26. Whitney, D.E., "Historical Perspective and State of the Art in Robot Force Control", *Intern. J. of Robotics Research*, Vol. 6, No. 1, Spring 1987, pp. 3-14
27. Kazerounian, K. and A. Nedungadi, "An Alternative Method for Minimization of Driving Forces in Redundant Manipulators", *Proc. IEEE Intern. Conf. on Robotics and Automation*, Raleigh, NC., April 1987, pp. 1701-1706
28. Petrov, I., *Variational Methods in Optimum Control Theory*, Academic Press, New York, 1968
29. Kazerounian, K. and Z. Wang, "Global Versus Local Optimization in Redundancy Resolution of Robotic Manipulators", *Intern. J. Robotics Research*, 1988, Vol. 7, No. 5, pp.3-12
30. Kirk, D.E., *Optimal Control Theory*, Prentice-Hall, Englewood Cliffs, NJ., 1970

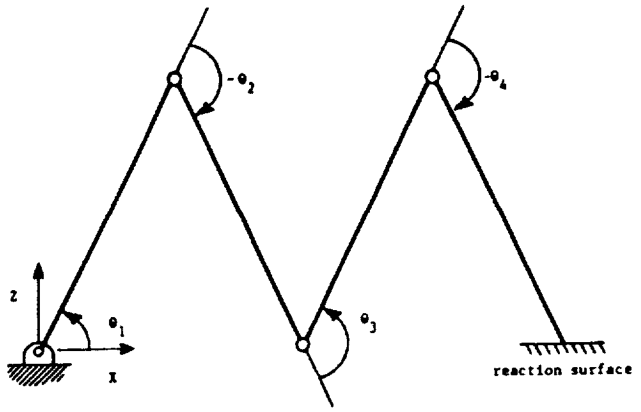


Figure 1. Four Link Robot in Horizontal Plane

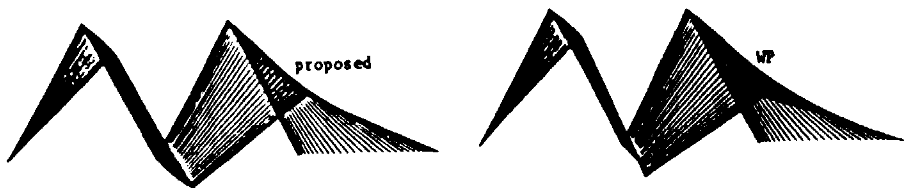


Figure 2 a. Robot Configurations in Simulation 1 (A_o = 0.5, w = 0.25)

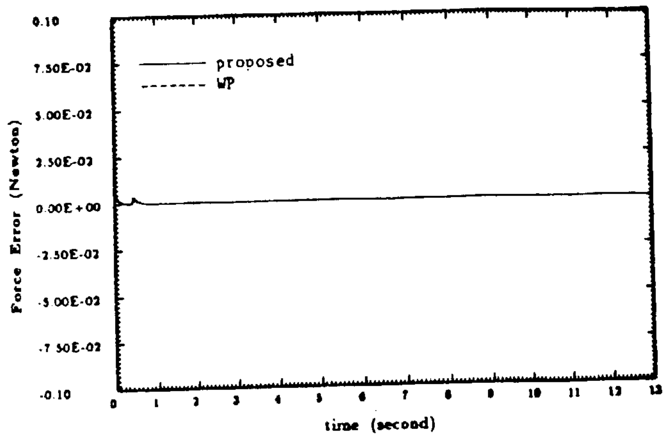


Figure 2 b. End-Effector/Environment Contact Force Error in Simulation 1 (A_o = 0.5, w = 0.25)

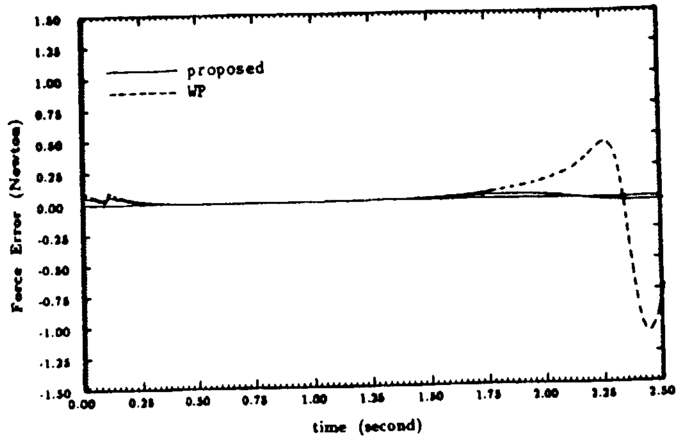


Figure 2d. End-Effector/Environment Contact Force Error in Simulation 1 (A_o = 0.8, w = 1.25)

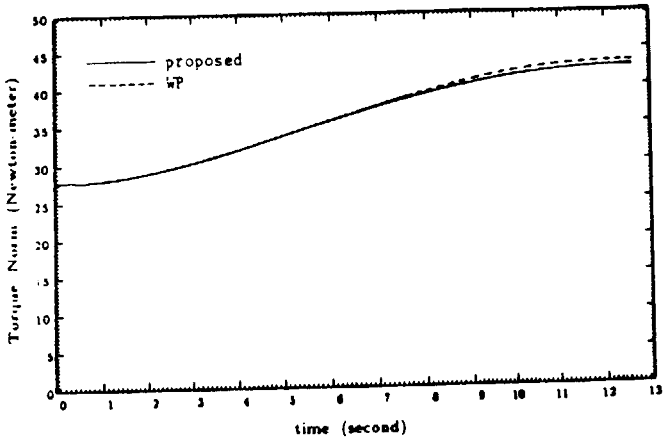


Figure 2 c. Norm of Joint Torque Vector in Simulation 1 (A_o = 0.5, w = 0.25)

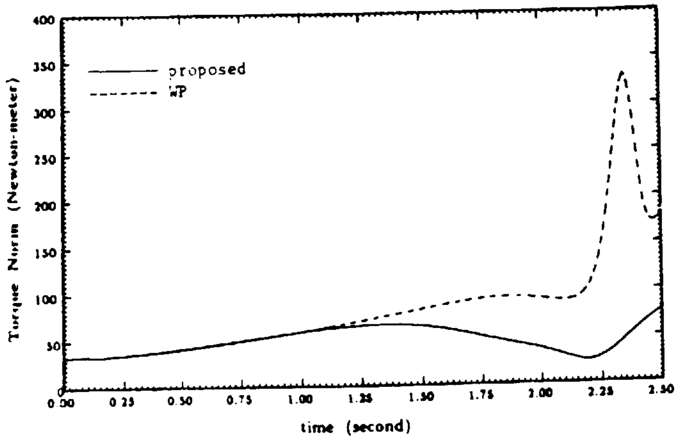


Figure 2e. Norm of Joint Torque Vector in Simulation 1 (A_o = 0.8, w = 1.25)

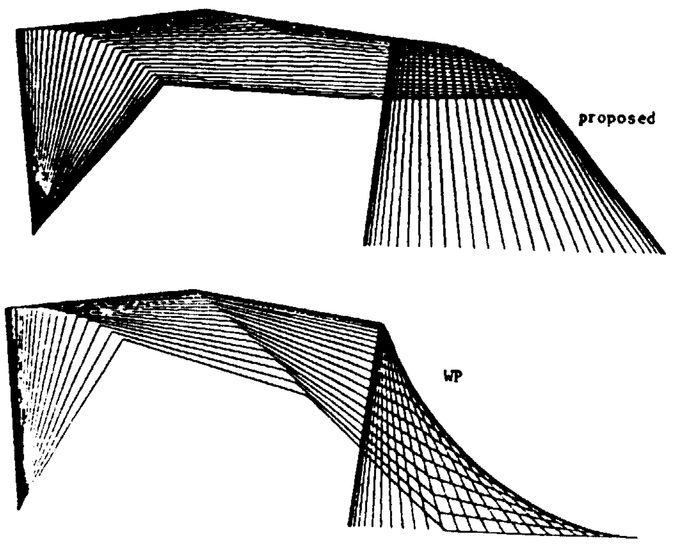


Figure 3a. Robot Configurations in Simulation 2

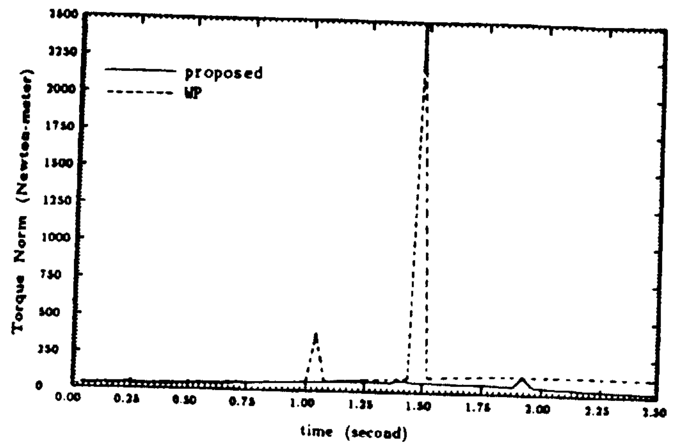


Figure 3c. Norm of Joint Torque Vector in Simulation 2

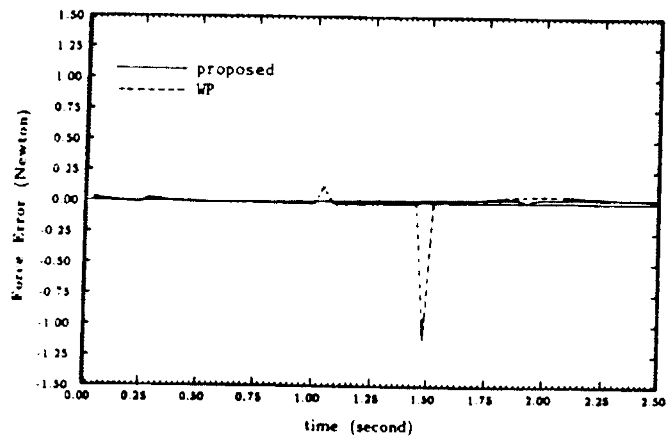


Figure 3b. End-Effector/Environment Contact Force Error in Simulation 2

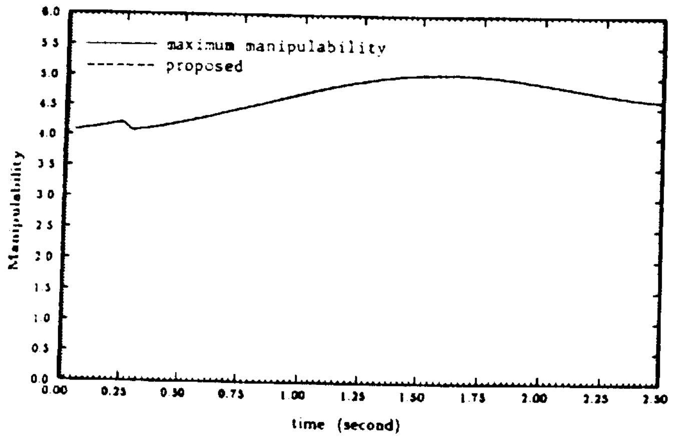


Figure 3d. Variation of Manipulability in Simulation 2

N90-29005

Kinematics, Controls, and Path Planning Results for a Redundant Manipulator

by Bruce Gretz[1] *and Scott Tilley*[2]
Ford Aerospace Corporation, Space Systems Division
3825 Fabian Way, Palo Alto, CA 94303

Abstract

The inverse kinematics solution, a modal position control algorithm, and path planning results for a 7 degree of freedom manipulator are presented. The redundant arm consists of two links with "shoulder" and "elbow" joints and a spherical wrist. The inverse kinematics problem for tip position is solved and the redundant joint is identified. It is also shown that a locus of tip positions exists in which there are kinematic limitations on self-motion. A computationally simple modal position control algorithm has been developed which guarantees a nearly constant closed-loop dynamic response throughout the workspace. If all closed-loop poles are assigned to the same location, the algorithm can be implemented with very little computation. To further reduce the required computation, the modal gains are updated only at discrete time intervals. Criteria are developed for the frequency of these updates. For commanding manipulator movements, a 5th-order spline which minimizes jerk provides a smooth tip-space path. Schemes for deriving a corresponding joint-space trajectory are discussed. Modifying the trajectory to avoid joint torque saturation when a tip payload is added is also considered. Simulation results are presented.

Introduction

Configuring and designing robotic systems for space applications involve many considerations not present in terrestrial systems. Safety and versatility are of prime importance. For example, safety concerns create a need for obstacle avoidance algorithms. Versatility demands may necessitate the manipulator's ability to perambulate between locations. Redundant manipulators meet these requirements because the additional degree(s) of freedom allow inclusion of obstacle avoidance algorithms and increase the maneuverability of the manipulator. The redundant joint configuration presently studied consists of two links with identical two degree-of-freedom "shoulder" and "elbow" joints and a spherical wrist, making a total of 7 degrees of freedom. This particular joint geometry has favorable characteristics with respect to singularity avoidance, obstacle avoidance, and simplicity. It is a candidate for use in several NASA applications on the Space Shuttle, Space Station, Polar Platform, and OMV.

Since this paper deals with quantitative results for a representative space-based manipulator, it is necessary to summarize the assumed system requirements. The fundamental task required is a pick-and-place motion involving a payload of mass up to 100 kg and tip forces of 100 N. Speed of operation is not deemed a high priority, so the manipulator has been designed to achieve tip velocities of 0.5 m/sec. The workspace should be roughly 4 m across, therefore the links are each 1.0 m long. As a result, the joint must be capable of exerting 200 Nm of torque to meet the 100 N tip force requirement. Each link has a mass of 30 kg, including the associated joint.

[1] R&D Engineer
[2] Engineering Specialist

Kinematics Analysis

Figure 1 shows the manipulator configuration. The arm consists of two links of length L_1 and L_2 connected by a two degree of freedom rotational joint (the "elbow"). The base link is attached to the ground by an identical joint (the "shoulder"). Each joint has one rotation axis parallel to the inboard link (roll) and one perpendicular to it (pitch). The shoulder roll axis is normal to the ground surface. The joint angles are denoted θ_1, θ_2, θ_3, and θ_4 and called shoulder roll, shoulder pitch, elbow roll, and elbow pitch, respectively. Joint angle limitations are not being considered.

The four degrees of freedom in the shoulder and elbow joints thus provide redundancy for positioning the manipulator tip A three dimensional wrist can then be used to orient the end effector. Assuming a spherical wrist, its kinematics are decoupled from those of the rest of the arm and are not treated in this analysis.

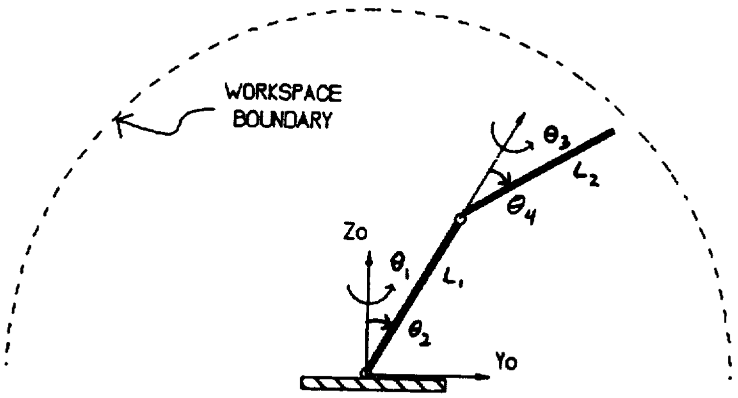


Figure 1: Joint Configuration of Redundant Manipulator

The forward kinematics of the arm are easily solved using a variety of methods. In the present analysis, homogeneous transformation matrices derived from the Denavit-Hartenberg parameters were multiplied together to produce the vector reaching from the base point to the tip [1]. The result is

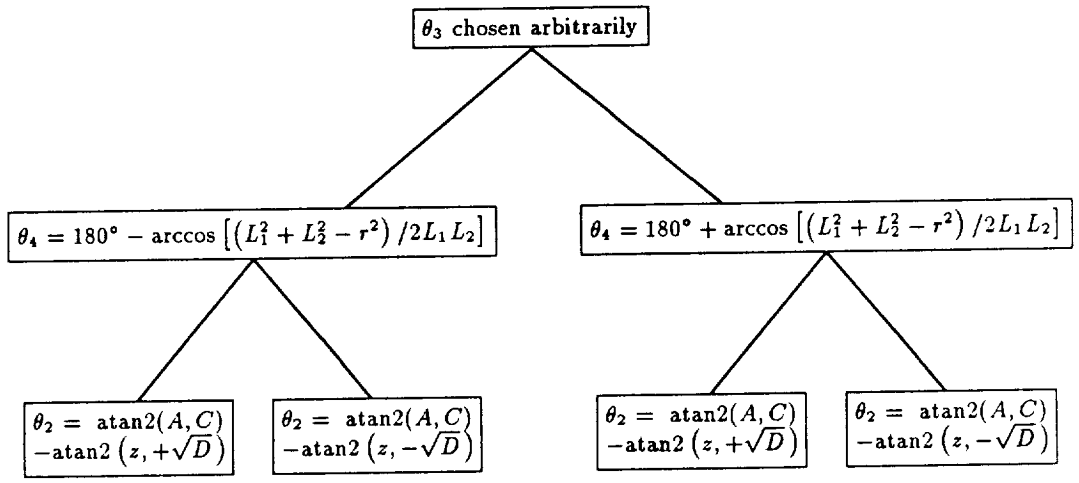
$$\begin{aligned} x &= L_1 c_1 s_2 + L_2 \left(c_1 s_2 c_4 - s_1 s_3 s_4 + c_1 c_2 c_3 s_4\right) \\ y &= L_1 s_1 s_2 + L_2 \left(s_1 s_2 c_4 + c_1 s_3 s_4 + s_1 c_2 c_3 s_4\right) \\ z &= L_1 c_2 + L_2 \left(c_2 c_4 - s_2 c_3 s_4\right). \end{aligned} \tag{1}$$

where c_1 denotes $\cos\theta_1$, etc. The reachable workspace is a sphere of radius $L_1 + L_2$ centered at the base.

Every redundant manipulator is capable of self-motion, that is, the tip can be fixed while the joint angles are varied. For the present manipulator, self-motion consists of "orbiting" the elbow joint in a circle. During orbiting all four joint angles must change. In particular, the elbow roll angle varies from 0° to 360°. It follows that for a given tip position, an inverse kinematics solution can be found for any elbow roll angle. The same cannot be said for the other three degrees of freedom, therefore the elbow roll angle is the redundant joint. (For some tip positions, there exists a kinematic limitation on the elbow roll angle. This will be addressed later.)

Specifying the tip position and elbow roll angle does not uniquely determine the other joint angles – there are still four possible solutions. These solutions determine one of two possible positions of the elbow joint and one of two possible orientations of link 1. For example, if the tip lies in the xy-plane (see Figure 1) then a point on the side of link 1 could "face" the z-axis or the xy-plane. Also, the elbow joint may be above or below the xy-plane.

Figure 2 shows an inverse kinematics "tree" containing the equations for these four solutions. The first step in obtaining a solution is arbitrarily choosing the elbow roll angle, θ_3. The elbow pitch angle , θ_4, is found next by examining the triangle whose sides are the two links and the vector from base to tip, $\vec{r}$. Since all three sides are known, the angle between the links is easily computed. Its supplement is θ_4. This solution has two values corresponding to the ambiguity in the sign of the inverse cosine. Physically, this corresponds to the elbow bending "up" or "down" and determines one of the two possible elbow joint positions. Once one of these two configurations is chosen, the appropriate branch of the tree is selected. The shoulder pitch angle, θ_2, is computed next. Its equation is found by manipulation of the forward kinematics equations. The sign ambiguity in this equation corresponds to link 1 facing "up" or "down". This choice of sign determines the final branch of the tree. The shoulder roll angle is now uniquely determined.


θ_3 chosen arbitrarily
$\theta_4 = 180° - \arccos\left[\left(L_1^2 + L_2^2 - r^2\right)/2L_1L_2\right]$
$\theta_4 = 180° + \arccos\left[\left(L_1^2 + L_2^2 - r^2\right)/2L_1L_2\right]$
$\theta_2 = \text{atan2}(A, C) - \text{atan2}\left(z, +\sqrt{D}\right)$
$\theta_2 = \text{atan2}(A, C) - \text{atan2}\left(z, -\sqrt{D}\right)$
$\theta_2 = \text{atan2}(A, C) - \text{atan2}\left(z, +\sqrt{D}\right)$
$\theta_2 = \text{atan2}(A, C) - \text{atan2}\left(z, -\sqrt{D}\right)$

For all solutions, $\boxed{\theta_1 = \text{atan2}[Bx + (As_2 + Cc_2)y, (As_2 + Cc_2)x - By]}$

$$A \equiv L_1 + L_2c_4 \qquad C \equiv L_2s_4c_3$$
$$B \equiv -L_2s_4s_3 \qquad D \equiv A^2 + C^2 - z^2$$

Figure 2: The Four Inverse Kinematics Solutions

It has been noted that for some tip positions, there is a kinematic limitation on self-motion, meaning that the elbow roll angle cannot take on an arbitrary value. Mathematically, this limitation can be derived from the equation for θ_2. If D, equal to $A^2 + C^2 - z^2$, is less than zero, then no solution exists. This occurs when A and C are both "small". A is the length of the arm projected onto the vector parallel to link 1, so A decreases as the arm is folded onto itself. C is proportional to $\cos\theta_3$, thus it decreases as θ_3 nears 90°. From this qualitative analysis two results may be concluded: 1) When the arm is relatively far extended the elbow roll angle can take on any value and thus complete orbiting is possible, and 2) When the arm is folded towards itself the elbow roll angle must be near 0°. Both of these conclusions can be restated rigorously. Assuming $L_1 = L_2 = L$, it can be shown that for tip positions lying outside of the volume defined by two spheres centered at $z = \pm L$ and having radius L, the elbow roll angle may take on any value. For tip positions lying inside of this volume, the elbow roll angle is constrained to

$$|\theta_3| < \arccos\left\{\sqrt{\frac{z^2 - (L_1 - \gamma L_2)^2}{L_2^2(1-\gamma^2)}}\right\}, \tag{2}$$

where $\gamma \equiv (L_1^2 + L_2^2 - r^2)/(2L_1L_2)$. This range of angles is centered around $\theta_3 = 0$. Figure 3 shows the regions of limited orbit capability.

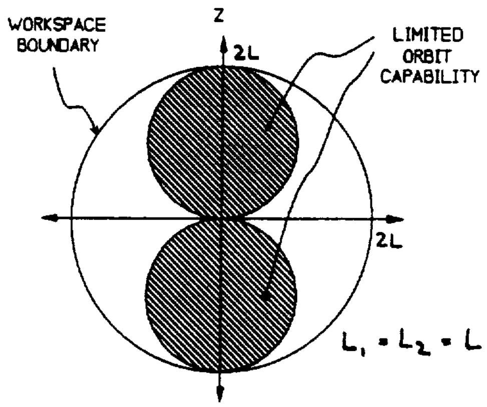


Figure 3: Regions of Limited Orbit Capability

For maximum maneuverability, Figure 3 indicates that it is desirable to keep the workspace near the xy-plane. An interesting parallel exists between this workspace location and the dexterous "workspace" of the human arm. The human arm is kinematically similar to the manipulator if we visualize its "z" axis extending horizontally out the sides of the shoulder. Our arms are most dexterous in front of us, which is near our xy-plane and corresponds to the manipulator area of complete orbit capability.

Controller Design and Analysis

The controller design for a space-based manipulator is primarily driven by requirements to maintain a specified closed-loop bandwidth with a minimum of computational complexity. The bandwidth is specified to accurately follow commanded trajectories. Disturbance rejection and modeling error impacts will be discussed later. Computed torque controllers, which use feedforward, will provide good dynamic response throughout the workspace; however, their computational complexity may preclude their use in space applications. The modal control algorithm presented in this paper is designed to maintain a nearly constant closed-loop dynamic response with a minimum of computation.

The equations of motion of any space manipulator take the form

$$\tau_c = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + F(\Theta, \dot{\Theta}) - \tau_d, \tag{3}$$

where τ_c is the joint control torque, Θ is a vector of joint angles, M is the mass matrix, and V is the nonlinear "velocity-squared" term of the dynamics, F is the friction terms, and τ_d is the joint disturbance torque arising from tip disturbance torques. For purposes of controller design, V and F can be viewed as a disturbance torques. Therefore, assuming the controller will have sufficient disturbance rejection and/or V and F are sufficiently small, the controller can be designed based on the approximate equations of motion given by

$$\tau_c = M(\Theta)\ddot{\Theta}. \tag{4}$$

If constant gain colocated joint control is applied to a manipulator the dynamic response varies widely throughout the workspace. Equation 4 shows that for slow motions this variation is primarily caused by the changes in the mass matrix as a function of Θ. (Physically, the apparent inertia at each joint changes with arm geometry.)

The modal control algorithm applies colocated joint torques using feedback gains which vary with configuration in order to ensure a nearly constant closed-loop bandwidth throughout the workspace. The feedback gains are computed from the mass matrix as follows. The mass matrix M is always real positive definite and thus may be transformed such that $S^T M S = D$, where D is diagonal and $S^T S = I$. The simplified equations of motion (equation 4) then become

$$S^T M S \ddot{\eta} = S^T \tau_c, \tag{5}$$

where $\eta \equiv S^T \Theta$. The elements of η are called the modal coordinates. Equation 5 may be rewritten as

$$D\ddot{\eta} = u, \tag{6}$$

where $u \equiv S^T \tau$ is the modal control torque. Placing the poles of this system using modal position and rate feedback is almost trivial because D is diagonal. Its diagonal elements are the modal inertias, denoted λ_i. The modal control torque thus takes the form

$$u_i = -(K_{p,i}\eta_i + K_{r,i}\dot{\eta}_i), \tag{7}$$

where $K_{p,i} = \lambda_i \omega_i^2$ and $K_{r,i} = 2\lambda_i \zeta_i \omega_i$ are the ith modal position and rate gains, respectively, which give the closed-loop poles associated with η_i a damping of ζ_i and a frequency of ω_i. The modal control torque is transformed back into joint space to give the joint control torque as

$$\tau_c = -Su = -S\left[K_p \quad K_r\right]\begin{Bmatrix}\eta \\ \dot{\eta}\end{Bmatrix} = -S\left[K_p \quad K_r\right]\begin{bmatrix}S^T & 0 \\ 0 & S^T\end{bmatrix}\begin{Bmatrix}\Theta \\ \dot{\Theta}\end{Bmatrix}, \tag{8}$$

where K_p and K_r are diagonal matrices containing the position and rate gains given in equation 7. The control torque may be rewritten as

$$\tau_c = -\left[SK_pS^T \quad SK_rS^T\right]\begin{Bmatrix}\Theta \\ \dot{\Theta}\end{Bmatrix}. \tag{9}$$

Since eigenvalues are preserved under a similarity transformation, the feedback scheme of equation 9 results in the same closed-loop poles that were assigned to the modal coordinates using equation 7. As a result, a constant dynamic response throughout the workspace is assured for sufficiently slow manipulator motions.

The choice of closed-loop frequency and damping is the result of hard requirements and engineering judgments. The requirements arise from desired tracking accuracy. This will be discussed later. The engineering judgements include considerations of disturbance rejection, positioning accuracy, tip force application, and noise sensitivity. Other factors which impact system stability and performance include structural flexibility, modeling errors, and time delays. Further, the control must be implemented on actuator/drive subsystems which contain their own dynamics [2]. The influence on all of these factors on choice of closed-loop pole location are topics of continuing research.

Reducing Control Computation

Implementing the algorithm described above requires diagonalization of the mass matrix and several matrix multiplications involving S. Much of this computation can be avoided by a simple restriction on the pole placement, namely, that each of the poles corresponding to the η_i be placed at the same location. In this case, the position and rate gain matrices become

$$\begin{aligned} K_p &= D\omega^2 \\ K_r &= 2D\zeta\omega, \end{aligned} \tag{10}$$

where ζ and ω are the damping and frequency, respectively, of that one pole location. As a result, equation 9 reduces to

$$\tau_c = -\left[M\omega^2 \quad 2M\zeta\omega\right]\begin{Bmatrix}\Theta \\ \dot{\Theta}\end{Bmatrix}. \tag{11}$$

The control gains can thus be computed simply by multiplying the mass matrix by a scalar. The restriction that all modal poles be placed at the same location is not unrealistic. For trajectory following, it is only necessary that their frequencies are sufficiently high and their damping is adequate.

Further computation reduction can be achieved by updating the control gains (the matrix of equation 9) less frequently than every microprocessor cycle. Thus the same control gains are used for several cycles even though the manipulator configuration is changing slightly. The gain computation can then be spread over several cycles with the gains being updated only after the computation is complete.

Analysis has been performed to determine how often these updates need to take place. The minimum gain update frequency depends on how fast the mass matrix is changing since the gains are computed from it. For the present manipulator, the mass matrix is most sensitive to the shoulder and elbow pitch angles. The shoulder pitch angle changes the apparent inertia about the shoulder roll joint because it moves the entire manipulator either closer or farther from that joint's axis. The elbow pitch angle folds the arm either in or out and thus changes the apparent inertia about both shoulder joints. As a result, the gain update frequency should be set according to expected pitch angle rates for a given manipulator motion. It has been found from simulation that the gains should be updated no less than once every 5° of either pitch angle rotation. Such rotations change the terms in the inertia matrix by less than 10%, provided the arm is not fully extended.

Tip-Space Stiffness and Contact Forces

Of interest in the design of a position controller are the forces of contact generated when the manipulator tip approaches a desired location and touches the environment. These forces are important in determining how well a manipulator performs a given task and whether there is a possibility of damaging the environment. For these reasons, it is important to quantify the contact forces which a given controller generates.

Contact forces are generated by a manipulator under position control when an obstacle prevents the tip from reaching its commanded position. For example, if the tip is commanded to a position behind a wall (due to position sensor inaccuracies or an error in modeling the environment) the tip will be stopped by the wall but continue to exert a force on it as the tip tries to reach the commanded position. The magnitude of the resulting contact force depends on how far the commanded position is behind the wall and the Cartesian "stiffness" of the control system. This stiffness can be expressed as a matrix K in the relation

$$F = -K\Delta x, \tag{12}$$

where F is the force acting on the wall and Δx the distance from the commanded position to the wall. Equation 12 also expresses the relation between a force exerted on the tip in free space and the resulting tip deflection.

Any manipulator under position control exhibits such a stiffness due to position feedback in the controller. In steady-state, the control law can be written $\tau_c = -G\Delta\Theta$, where G is the position gain matrix given by $G = SK_pS^T$ when using modal control (see equation 9). It is well-known that the Jacobian J relates tip deflections to joint deflections by $J\Delta\Theta = \Delta x$ and tip forces to joint torques by $J^TF = \tau$. Substituting these two relations into the steady-state control law above and rearranging yields

$$F = -(JSK_p^{-1}S^TJ^T)^{-1}\Delta x. \tag{13}$$

Thus the apparent stiffness matrix of the manipulator under modal control is $K = (JSK_p^{-1}S^TJ^T)^{-1}$. The properties of this matrix as a function of the controller gains and joint angles is a topic of continuing research.

Path Planning for a Redundant Manipulator

The simplest problem in path planning is computing a tip-space position, velocity, and acceleration trajectory that moves the end effector from one point to another. A 5th-order spline has been chosen for this purpose because it can give zero velocity and acceleration at the end points. The solution is

$$x(t) = x_i + (6\tau^5 - 15\tau^4 + 10\tau^3)(x_f - x_i), \tag{14}$$

where x_i and x_f are the initial and final positions and τ (defined as t/T) is normalized time with T the total maneuver time [3]. It can be shown that this spline also gives the minimum jerk for any polynomial trajectory. Hollars recommends that the controller have a bandwidth of at least $4/T$ Hz to adequately track this spline. The same spline is used for all three tip-space coordinates. As a result the trajectory is a straight line between the start and end points.

The next task in path planning is generating a joint-space trajectory corresponding to the desired tip-space trajectory. For redundant manipulators, there exists an infinite number of joint trajectories for each tip trajectory. In the present case there is one redundant degree of freedom, therefore one additional constraint must be added in order to produce a solution. This constraint could arise from considerations of singularity avoidance, obstacle avoidance, tip-space stiffness, etc.

The present manipulator has no internal singularities within the region of complete orbit capability. Therefore, an easy singularity avoidance scheme consists of limiting the workspace to this region. A constraint still needs to be chosen to solve the inverse kinematics. The constraint $\theta_3 = 0$ is one simple possibility. This leaves four possible solutions for the other joint angles (see Figure 2). A single one can be selected based on how the links are to be oriented during the motion (elbow "up" or "down", etc.). This choice could be driven by constraints on the position of the elbow itself arising from obstacle avoidance concerns.

Another possible constraint is minimizing joint velocities. This can be accomplished by resolved-rate control in which a desired tip velocity trajectory is transformed into a joint velocity trajectory. The Jacobian pseudo-inverse is used to find the instantaneous minimum joint velocity. The solution is

$$\dot{\Theta}(t) = J^{\dagger}\dot{X}(t), \tag{15}$$

where $\dot{X}(t)$ is the vector of tip-space coordinates and $J^{\dagger}$ is the Jacobian pseudo-inverse given by $J^{\dagger} = J^T(JJ^T)^{-1}$. This solution minimizes the 2-norm of the joint velocity vector at each point in the trajectory. Several modifications to this method have been proposed in the literature [4,5]. They generally attempt to optimize some other performance criterion or potential function.

One argument for using equation 15 is that it helps avoid singularities because joint velocities tend to increase near them. However, this method causes the tip to follow the desired trajectory exactly, therefore if the trajectory passes close to a singularity then the minimum joint velocity solution can be arbitrarily large. Wampler and Leifer [6] have proposed an interesting modification to this method which causes the tip to deviate from the desired trajectory when it approaches a singularity. In this way an upper bound on joint velocities can be maintained.

For the present manipulator, limiting the workspace to the region of complete orbit capability will avoid all internal singularities. If the manipulator is required to move out of this region then $\theta_3 = 0$ is the recommended constraint because it will avoid orbit angle limits. If the tip is always in this region then either $\theta_3 = 0$ or equation 15 gives acceptable results for simple pick-and-place operations. When constraints involving obstacle avoidance, elbow joint position, or tip stiffness arise, the redundancy can be used to address them.

Path Planning in the Presence of a Tip Payload

Since we are designing a manipulator to perform pick-and-place operations, path planning with a tip payload is of concern. Clearly, executing a trajectory with a tip payload will require larger control torques than tracking the same trajectory without a payload. A nominal trajectory duration for movements without a payload should be selected such that the peak joint torque commanded is a certain fraction of the maximum joint torque. This nominal duration should be varied with trajectory distance in order to keep the average tip velocity constant. This will ensure that the velocity-squared terms and the inertia term of the equations of motion maintain the same relative magnitude (see equation 3).

When a payload is added, the nominal trajectory must be modified in order to ensure the same peak joint torque command. Using simple results from the dynamics of accelerating a point mass, we can assume that the maximum control torque required to execute a trajectory is inversely proportional to the square of the maneuver time. That is,

$$\tau_{c,max} \propto \frac{1}{T^2}, \tag{16}$$

where $\tau_{c,max}$ is the maximum control torque and T is the trajectory duration. The first step in modifying the trajectory is running a simulation to determine the peak joint torque commanded when moving the payload through the nominal trajectory duration. Equation 16 can then be used to adjust the maneuver time accordingly. Also, the desired closed-loop pole frequency should be lowered so that it is no higher than that required for tracking. This will minimize the sensitivity of the controller to noise and unmodeled dynamics.

The mass matrix used to compute the control gains should include modeling of the payload. If it does not then the closed-loop poles will have a lower frequency and damping than that desired. As a result, the disturbance and noise rejection may be degraded. Including modeling of the payload in the mass matrix will ensure that the desired closed-loop poles are achieved. Since space-based manipulators will initially be used in highly structured environments, the time of attachment and mass properties of payloads should be readily available.

Simulation Results

This section presents simulations of tip trajectory following with and without a payload using the modal control algorithm and two redundancy management schemes. The starting and ending tip coordinates (in meters) in the x-y-z coordinate system of Figure 1 are $(-0.8, 1.0, 0.6)$ and $(0.6, 1.2, -0.8)$, respectively, giving a trajectory length of about 2 m. Note that the line connecting these points lies completely within the area of complete orbit capability. The control gains are updated every 0.25 sec and the payload is assumed to be a point mass of 100 kg located at the tip.

Figure 4 shows the response with no payload using the $\theta_3 = 0$ constraint. All closed-loop pole frequencies are set to 2.4 rad/sec, which is the minimum required for a 10 sec slew. In the first plot the actual and commanded tip motion are shown. Although the actual tip motion lags slightly behind the desired trajectory, it converges accurately to the desired end position at the end of the maneuver. Note that the discrete gain updating causes jumps in the commanded joint torques. Since joint dynamics are not modeled here, the commanded torque is equal to the applied torque. In actuality, the dynamics of the joint motor will smooth these jumps while not degrading the tracking accuracy. Modeling joint dynamics is currently being researched [2].

Figure 5 shows the same simulation except that the pseudo-inverse is used to generate the joint trajectory. Notice how θ_3 attains a final angle of about 35° in order to decrease the average velocity of the other three joints. Otherwise, the performance is the same as before. Figure 6 shows the simulation of Figure 4 except that a 100 kg payload has been added. Another simulation showed that the maximum control torque with this payload and a maneuver time of 10 sec is about 12 Nm. Therefore, using equation 16 and the fact that the previous simulations have maximum

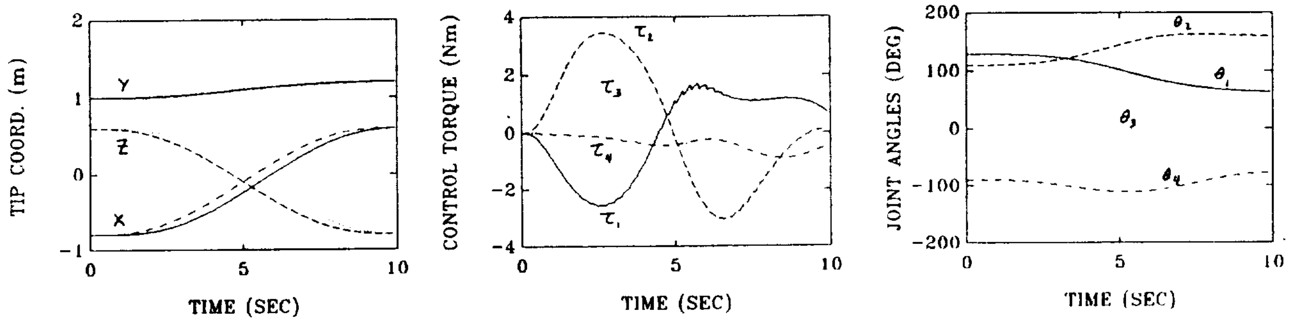


Figure 4: Simulated Maneuver with $\theta_3 = 0$ Constraint

control torques of 3 Nm, the maneuver time was lengthened by a factor of 2 ($= \sqrt{12/3}$) to bring the maximum control torque back to 3 Nm. In addition, the closed-loop poles were reduced to 1.2 rad/sec to match the increase in maneuver time. The tracking performance is as good as that with no payload.

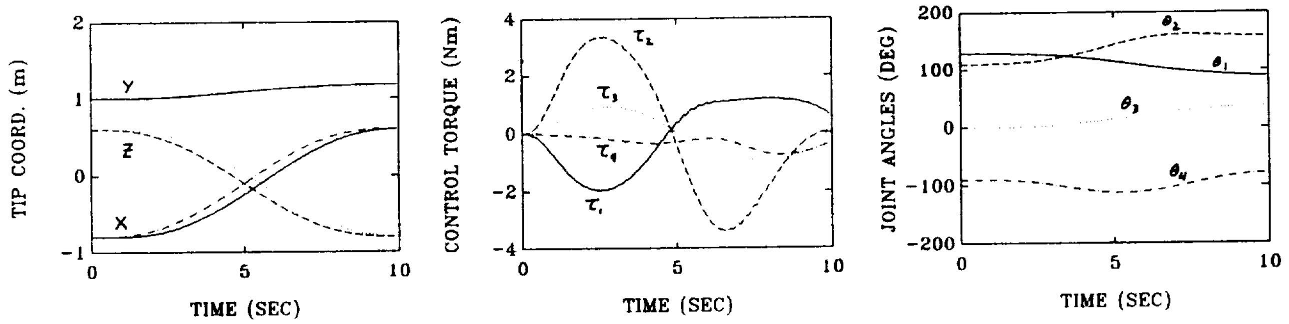


Figure 5: Simulated Maneuver with Minimum Joint Velocity Constraint

Conclusions

The inverse kinematics solution, a modal position control algorithm, and path planning results for a 7 degree of freedom manipulator have been presented. After arbitrarily choosing the elbow roll angle, the redundant degree of freedom, the inverse kinematics has four solutions. Each solution corresponds to a different orientation of the links in space. It is also shown that a locus of tip positions exists in which there are kinematic limitations on the orbit angle.

A computationally simple modal position control algorithm has been developed which guarantees a nearly constant closed-loop dynamic response throughout the workspace. The algorithm consists of diagonalizing the mass matrix into four modal inertias and computing feedback gains to control the modal coordinates. This controller is able to reject the disturbance arising from the unmodeled velocity-squared terms. If all closed-loop poles are assigned to the same location, the algorithm can be implemented with very little computation. To further reduce the required computation, the modal gains are at discrete time intervals. An update frequency of every 5° of either pitch angle

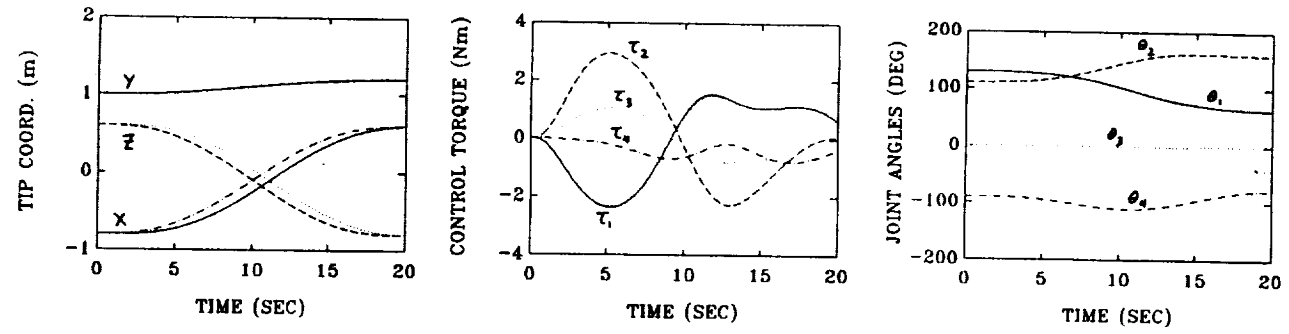


Figure 6: Simulated Maneuver with 100 kg Payload

motion significantly reduces computation without degrading performance.

For commanding manipulator movements, a 5th-order spline with zero velocity and acceleration at the end points provides a smooth tip-space path. The frequencies of the closed-loop poles should be at least $4/T$ Hz, where T is the trajectory duration, to maintain adequate tracking. The best singularity avoidance scheme is keeping the tip trajectory in the region of complete orbit capability. The orbit angle can then be used to address other constraints such as obstacle avoidance or tip-space stiffness. A method is presented for modifying the trajectory duration when a payload is added to maintain a constant joint control torque. The payload should be modeled in the mass matrix to allow accurate control over the closed-loop bandwidth.

References

[1] Craig, J.J., "Introduction to Robotics – Mechanics & Control", Addison-Wesley, 1986.

[2] Tilley, S.W., Emerick, K., Francis, C., and Hollars, M.G., "Preliminary Results on Noncolocated Torque Control of Space Robot Actuators", presented at the NASA Conference on Space Telerobotics, Pasadena, CA, January 1989.

[3] Hollars, M.G., Experiments in End-Point Control of Manipulators with Elastic Drives", Stanford University PhD Thesis, May 1988.

[4] Dubey, R. and Luh, J.Y.S., "Performance Measures and Their Improvement for Redundant Robots", presented at Winter Annual Meeting of ASME, Anaheim, CA, from *Robotics: Theory and Applications*, DSC vol. 4, pp. 143-151, December 1896.

[5] Walker, Ian D. and Marcus, Steven I., "Subtask Performance by Redundancy Resolution for Redundant Robot Manipulators", Communication in *IEEE Journal of Robotics and Automation*, vol. 4, no. 3, pp. 350-354, June 1988.

[6] Wampler II, C.W. and Leifer, L.J., "Applications of Damped Least-Squares Methods to Resolved-Rate and Resolved-Acceleration Control of Manipulators", *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 31-38, March 1988.

N90-29006

A Complete Analytical Solution for the Inverse Instantaneous Kinematics of a Spherical-Revolute-Spherical (7R) Redundant Manipulator

R.P. Podhorodeski, R.G. Fenton, A.A. Goldenberg
Robotics and Automation Laboratory
Dept. of Mech. Eng., University of Toronto
Toronto, CANADA, M5S 1A4

Abstract

Using a method based upon resolving joint velocities using reciprocal screw quantities, compact analytical expressions are generated for the inverse solution of the joint rates of a seven revolute (spherical-revolute-spherical) manipulator. The method uses a sequential decomposition of screw coordinates to identify reciprocal screw quantities used in the resolution of a particular joint rate solution, and also to identify a Jacobian null-space basis used for the direct solution of optimal joint rates. The results of the screw decomposition are used to study special configurations of the manipulator, generating expressions for the inverse velocity solution for all non-singular configurations of the manipulator, and identifying singular configurations and their characteristics.

This paper therefore serves two functions: a new general method for the solution of the inverse velocity problem is presented; and complete analytical expressions are derived for the resolution of the joint rates of a seven degree of freedom manipulator useful for telerobotic and industrial robotic application.

1. Introduction

The inverse velocity problem for a redundant manipulator is underdetermined. That is, an infinite number of joint rate solutions providing a required end effector velocity will exist. A means of resolving the "best" joint rate solution and computation efficiency are requirements of an inverse velocity solution method. To form a complete inverse instantaneous (velocity) kinematic solution for a specific manipulator, special configurations and their characteristics must be identified.

Several approaches for the resolution of "optimal" joint rates for redundant manipulators have been proposed. These techniques can be classified as local (e.g. see Hollerbach and Suh[1] and the references of [1], [2], [3] and [9]), global (e.g. see Kazerounian and Wang[2] and the references of [2], and [3]), kinematic function based (e.g. see Wampler and Baker [3]), and constraint based (e.g. see Baillieul[4]). Global, kinematic function, and constraint techniques (in a local sense), have the advantage of maintaining the same joint displacements during repetitive execution of a task. Local optimizations have the disadvantage of being nonrepetitive, and globally nonoptimal, but remain an important technique where insufficient information or computational time is available for global optimization.

Analytical derivation of expressions for the inverse velocity solution allow a computational efficiency difficult to achieve with numerical solution schemes. Works by Sugimoto[5] ("orthogonal basis" decomposition of screw coordinates), Hunt[6] (direct inversion of a screw coordinate matrix (Jacobian) using convenient frames of reference), and Stanisic et. al.[7] (canonical reference for three parameter motion) are recent examples of techniques for the derivation of analytical expressions for the inverse velocity solution of nonredundant manipulators.

In this work, an inverse velocity solution based on a decomposition of screw coordinates is presented (Sections 2,3 and 4), and is applied to the derivation of analytical results for a seven revolute (7R) manipulator (Section 5). The decomposition identifies reciprocal screw quantities (terminology reviewed in Section 2) used for a particular joint velocity solution, and a basis for the Jacobian null-space useful in joint rate optimization. Optimization for quadratic objective functions, yields direct solutions for the optimum (local) joint rates in terms of pseudo-inverses of a weighting of the null-space basis. These solutions require the inverse of matrices of reduced order, (e.g. a scalar quantity or a seven degree of freedom robot), in comparison to pseudo-inverses of the manipulator Jacobian.

The 7R manipulator analyzed features a spherical base, a revolute elbow, and a spherical wrist. This joint layout was proposed by Hollerbach[8] as an "optimal" seven degree of freedom layout, for which one of the objectives was the elimination of singularities caused by single joint displacement conditions. As such, the robot should be useful for telerobotic and industrial application where a degree of autonomous motion is required (i.e. preplanning for singularity avoidance is not possible). Analytical results are derived for the inverse velocity solution for all non-singular configurations. Singular configurations are examined and characterized in terms of the screw decomposition.

2. Resolving Joint Velocities Using Reciprocal Screws

A *screw* is a line in space having an associated linear *pitch.* [11] As such it represents five independent parameters (four for the line, one for the pitch). Associating an amplitude acting on the screw yields six independent parameters. Screw quantities are natural entities for describing spatial instantaneous motion (velocities) and forces and moments, (i.e. any velocity can be considered to be a rotational velocity about an axis and a translational velocity parallel to the same axis, and any system of forces and moments is equivalent to a force in a direction and a couple in a plane perpendicular to the direction).

A screw can be represented as a dual vector by its screw coordinates, $\{\$; \$_o\}^T$,

$$\$ = \{\$; \$_o\}^T = \{\mathbf{L}; \mathbf{L}_o + p_L \mathbf{L}\}^T \tag{1}$$

where $\mathbf{L}$ and $\mathbf{L}_o$ are respectively the direction of the line and its moment about a reference origin (Plucker line coordinates), and p_L is the pitch of the screw. A screw quantity is represented by the product of an amplitude and a screw,

$$\mathbf{S} = \alpha\{\$; \$_o\}^T = \{\mathbf{s}; \mathbf{s}_o\}^T \tag{2}$$

If $\mathbf{S}$ is the velocity of a rigid body, *(a twist about a screw)*, then α is referred to as the *twist amplitude*, $\mathbf{s}$ is the angular velocity vector of the body, and $\mathbf{s}_o$ is the translational velocity of a point on the rigid body (extended to be) coincident with the reference origin. If $\mathbf{S}$ represents a system of forces *(a wrench acting on a screw)*, $\mathbf{s}$ is the resultant vector of the forces acting on the body, and $\mathbf{s}_o$ is the resultant vector of the moments acting on the body plus the sum of the moments of all forces about the reference origin.

The reciprocal product of two screw quantities is the inner product,

$$\mathbf{S}_1 \,\chi\, \mathbf{S}_2 = \mathbf{s}_1 \cdot \mathbf{s}_{o_2} + \mathbf{s}_{o_1} \cdot \mathbf{s}_2 \tag{3}$$

The reciprocal product of a "twist" and a "wrench" quantifies a rate of work. Two screws are reciprocal when their reciprocal product is zero, e.g. a body having a motion described by a twist, $\mathbf{S}_i$, subjected to a force system described by a wrench on a screw reciprocal to, $\mathbf{S}_i$ performs no work. A set of r linearly independent screws forms an r-system. Reciprocal to an r-system is a $(6-r)$-system of screws [12].

If a rigid body is acted upon by twist amplitudes about a chain of n screws the resulting velocity, $\mathbf{M}$, is

$$\alpha_1\$_1 + \alpha_2\$_2 + \cdots + \alpha_n\$_n = \mathbf{M} \tag{4}$$

In **robotics application** the joint axes are the screws, $\$_i$, $i=1,n$, (and the screw coordinates can be shown to be equivalent to the columns of the manipulator Jacobian with respect to the frame of reference, i.e., $[J] = [\$_1 \cdots \$_n]$). The joint rates, $\dot{q}_i$, $i=1,n$, are the twist amplitudes of equation (4), and $\mathbf{M}$ is the end effector velocity. In the inverse velocity problem we are concerned with finding $\dot{q}_i$, $i=1,n$ such as to provide a required $\mathbf{M}$.

The joint rates can be resolved using reciprocal screw quantities. That is, if a wrench on a screw, $\mathbf{B}$ is known such that $\mathbf{B} \,\chi\, \$_i = 0$, $i \neq n$, and $\mathbf{B} \,\chi\, \$_n \neq 0$, the nth twist amplitude (joint rate) can be resolved by taking reciprocal products of both sides of equation (4) with $\mathbf{B}$.

$$(\alpha_1\$_1 + \alpha_2\$_2 + \cdots + \alpha_n\$_n) \,\chi\, \mathbf{B} = \alpha_n\$_n \,\chi\, \mathbf{B} = \mathbf{M} \,\chi\, \mathbf{B} \tag{5}$$

and therefore $\qquad \alpha_n = \dot{q}_n = (\mathbf{M} \,\chi\, \mathbf{B})/(\$_n \,\chi\, \mathbf{B})$

Equation (5) represents a virtual work expression, i.e. the rate of work done by the end effector moving at the rate, $\mathbf{M}$ when subjected to the wrench, $\mathbf{B}$, must be equal to the rate of work generated by the joint velocity, $\dot{q}_n$ about $\$_n$ when subjected to the same wrench, since $\mathbf{B} \,\chi\, \$_i = 0, i \neq n$.

3. An Inverse Velocity Solution Based on a Decomposition of Screw Coordinates

After a joint velocity, e.g. $\dot{q}_n$ of equation (5), is resolved, its contribution to the end effector velocity can be removed, e.g. $\mathbf{M}_{eff} = \mathbf{M} - \dot{q}_n\$_n$. Resolution of the next joint velocity, e.g. $\dot{q}_{n-1}$, requires only a screw quantity reciprocal to the remaining screws, e.g. $\$_1 \cdots \$_{n-2}$. These reciprocal screw quantities can be found using the sequential decomposition presented in this section.

Sequentially the jth screw of the Jacobian is decomposed into twist amplitudes, α_{ij}, about the joint screws, $\$_i$, $i<j$, and the complement of a wrench, $\mathbf{B}_j$, having a null reciprocal product (NRP) with the joint screws, $\$_i$, $i<j$. That is,

$$\$_j = \mathbf{B}_j^* + \sum_{i=1}^{j-1} \alpha_{ij}\$_i \tag{6}$$

where $\mathbf{B}_j \chi \$_i = 0$, $i<j$, and $\mathbf{B}^* = \{\mathbf{b}_{oj}; \mathbf{b}_j\}^T$ is defined as the wrench complement. By sequentially taking reciprocal products with $\mathbf{B}_i$, and noting $\mathbf{B}_j^* \chi \mathbf{B}_i = \mathbf{B}_j \chi \mathbf{B}_i^* = \mathbf{B}_j \chi (\$_i - \alpha_{i-1i}\$_{i-1} - \cdots - \alpha_{1i}\$_1) = 0$ for $i<j$, the required twist amplitudes, α_{ij}, are determined to be

$$\alpha_{ij} = (\$_{ij} \chi \mathbf{B}_i)/(\$_i \chi \mathbf{B}_i), \quad \text{where } \$_{ij} = \$_j - \sum_{k=i+1}^{j-1} \alpha_{kj}\$_k \tag{7}$$

Notice this is a Gram-Schimdt type decomposition [13] where the inner product is the reciprocal product, and dual vectors are being decomposed.

If $\$_j$ is linearly dependent on the previous j-1 linearly independent screws then a set of unique c_i exists such that $c_1\$_1 + c_2\$_2 + \cdots + c_{j-1}\$_{j-1} + \$_j = 0$. In this case the decomposition returns a null $\mathbf{B}_j^*$ value, and the values of α_{ij}, $i<j$, of equation (7) correspond to the negatives of the values of c_i, $i<j$. These values of c_i together with a value of one (1) associated with $\$_j$ form a vector for the null-space basis of [J] ($\equiv [\$_1 \cdots \$_n]$). For the decomposition of the remaining screws, $\$_j$ and its associated null $\mathbf{B}_j$ are not considered.

Let us assume that the first r screws of [J] are linearly independent, where r is the rank of [J], and the remaining $n-r$ screws are linearly dependent on the first r screws, (this is achieved by removing the linearly dependent screws as they are found in the decomposition sequence). The complements of the NRP wrenches may be expressed as

$$[\mathbf{B}^*] = [\mathbf{J}][\mathbf{d}] \tag{8}$$

where $[\mathbf{B}^*] = [\mathbf{B}_1^* \, \mathbf{B}_2^* \cdots \mathbf{B}_r^* \, \mathbf{0} \cdots \mathbf{0}]$, and $d_{ij} = 1$ if $i=j$, or $-\alpha_{ij}$ if $i<j$ and $i \le r$, or 0 otherwise

The last $n-r$ columns of [d] form a basis for the null-space of [J]. The ordering of the screws when doing the sequential decomposition is arbitrary, but must be maintained throughout the complete solution. The subscripts associated with the screw quantities in the decomposition can be considered to refer to integers of a set, $\$_{ord}$, corresponding to the order of decomposition.

A particular joint velocity solution can be formed by decomposing **M** into joint rates about the "linearly independent" screws of [J], i.e. $\mathbf{M} = \sum_{j=1}^{r} \dot{q}_{j_{part}} \$_j$, where

$$\dot{q}_{j_{part}} = (\mathbf{M}_j \chi \mathbf{B}_j)/(\$_j \chi \mathbf{B}_j), \quad j=r, 1, -1, \quad \text{with} \quad \mathbf{M}_j = \mathbf{M} - \sum_{m=j+1}^{r} \dot{q}_{m_{part}} \$_m \tag{9}$$

A general joint velocity solution can be expressed as

$$\{\dot{q}\}_{n \times 1} = \{\dot{q}\}_{part_{n \times 1}} + [a]_{n \times (n-r)} \{\lambda\}_{(n-r) \times 1} \tag{10}$$

where $[a]$ is a null-space for the joint screw coordinates (Jacobian). The particular joint velocity solution of equation (9) corresponds to a solution with $\dot{q}_{j_{part}} = 0$ for $j>r$. If this particular solution is used and $[a]$ is formed from the last $n-r$ columns of [d], recalling that [d] has unit values on the diagonal and is upper triangular, the $\{\lambda\}$ of equation (10) are seen to correspond to the values of $\dot{q}_j$, $j>r$. These rates shall be referred to as the redundant joint rates.

Optimizing the joint rate solution involves finding the optimal basis multipliers, $\{\lambda\}_{opt}$, (equivalent to the optimal "redundant" joint rates). Substitution into equation (10) then yields the optimal joint rates. For example consider a weighted sum square of the joint velocities, i.e. $f_{obj} = ([\mathbf{W}](\{\dot{q}\}_{part} + [\mathbf{a}]\{\lambda\}))^2$, where [W] is a weighting matrix. Differentiating f_{obj} with respect to $\{\lambda\}$ and equating to zero gives

$$\{\lambda\}_{opt} = -([\mathbf{a}]^T[\mathbf{W}]^T[\mathbf{W}][\mathbf{a}])^{-1}[\mathbf{a}]^T[\mathbf{W}]^T[\mathbf{W}]\{\dot{q}\}_{part} \tag{11}$$

Details on using the null-space basis for the optimization of joint rates for obstacle avoidance, joint displacement centering, joint torque minimization, and iterative least squares displacement closure can be found in [10] In each case joint rates are optimized for quadratic objective functions, resulting in direct solutions for the optimal redundant joint rates in terms of a left pseudo-inverse[13] of a weighting of the null-space basis, similar to that of equation (11).

Forming this pseudo-inverse requires the inversion of a $(n-r) \times (n-r)$ matrix, (e.g. for a seven degree of freedom robot with a Jacobian of full rank, this is a scalar quantity).

The results of the screw decomposition characterize the redundancy of the manipulator, indicate the rank of the Jacobian, and allow the solution of the inverse velocity problem. The method is suitable for numerical application, for which the computational costs are discussed in [9]. The method is also useful for the derivation of explicit expressions for the Jacobian null-space basis and null reciprocal product wrenches. These expressions allow analytical solution of the inverse velocity problem, and are useful for the identification of special configurations of the analyzed manipulator, as is demonstrated with an example in Section 5.

4. Multi-arms having Common Redundant Degrees of Freedom

Consider a system comprised of m manipulators (arms) sharing n_c common joint degrees of freedom. The Jacobian of the ith manipulator is composed of the joint screw coordinates individual to the particular manipulator, $[\$]_i$, and the screw coordinates of the common degrees of freedom, $[\$]_c$, i.e., $[\mathbf{J}]_{i_{6\times n_i+n_c}} = [[\$]_{i_{6\times n_i}}\ [\$]_{c_{6\times n_c}}]$. The screws $[\$]_i$ are assumed to span the task requirements of the ith manipulator, rendering the $[\$]_c$ joint degrees of freedom "redundant". The individual manipulator joint axes may also have redundancy, (i.e. $n_i > r_i$).

The screw coordinates of $[\mathbf{J}]_i$ can be decomposed yielding $[\mathbf{B}]_i$ and the null-space basis $[\mathbf{a}]_{i_{(n_i+n_c)\times(n_i-r_i+n_c)}}$. The null-space bases for each arm can be combined, concatenating the columns for the common degrees of freedom, to yield a null-space basis for the Jacobian of the entire manipulator system. That is,

$$[\mathbf{J}][\mathbf{a}] = \begin{bmatrix} [\$]_{1_{6\times n_1}} & \cdot & 0 & [\$]_{c_{6\times n_c}} \\ & \cdot & & \vdots \\ 0 & \cdot & [\$]_{m_{6\times n_m}} & [\$]_{c_{6\times n_c}} \end{bmatrix}_{6m\times(\sum n_i + n_c)} [\mathbf{a}]_{(\sum n_i+n_c)\times(\sum(n_i-r_i)+n_c)} = [\mathbf{0}] \qquad (12)$$

Similarly, particular solutions can be found for each arm and concatenated. The general joint solution solution becomes

$$\begin{Bmatrix} \{\dot q\}_1 \\ \cdot \\ \{\dot q\}_m \\ \{\dot q\}_c \end{Bmatrix}_{(\sum n_i+n_c)\times 1} = \begin{Bmatrix} \{\dot q\}_{1_{part}} \\ \cdot \\ \{\dot q\}_{m_{part}} \\ \{\dot q\}_{c_{part}} \end{Bmatrix}_{(\sum n_i+n_c)\times 1} + [\mathbf{a}]_{(\sum n_i+n_c)\times(\sum(n_i-r_i)+n_c)} \{\lambda\}_{(\sum(n_i-r_i)+n_c)\times 1} \qquad (13)$$

Note that a null-space basis is not a function of frame of reference. That is, a convenient frame of reference can be utilized to form each component of the assembled total system null-space basis. For multi-arm examples the reader is referred to [9] and [10].

5. Analytical Expressions for the Inverse Velocity Solution of a 7R manipulator

Overview

The decomposition of screw coordinates presented in the previous section is used in deriving expressions for [a] and [B] for the 7R manipulator illustrated in Figure 1. The manipulator features a spherical group of joints at the base and at the wrist. Hollerbach[8] suggested this joint layout as being the "optimal" for a 7R manipulator, for which one of the objectives of optimality was the elimination of singularities.

Based on the results of the screw decomposition special configurations of the manipulator are identified. These configurations correspond to cases when groups of joint axes become linearly dependent and yet the manipulator retains full motion ability, and to cases of joint dependency leading to motion ability degeneracy (singular configurations). Screw decompositions using two frames of reference are performed to form compact analytical expressions for use in the inverse kinematic solution for all non-singular configurations of the manipulator. Singular configurations are examined and characterized within the context of the screw decomposition.

A screw decomposition

Solution of the inverse kinematic problem can be performed with respect to any frame of reference. A frame of reference was chosen as: $z_{ref\,1}$ aligned in the direction of $\$_5$; $y_{ref\,1}$ in the opposite direction to that of $\$_4$; with the origin of the reference frame located at the intersection of the three wrist axes, see Figure 1. This reference frame was chosen to exploit the decoupling provided by the spherical wrist, and to minimize the complexity of the Jacobian terms. With respect to this reference frame the screw coordinates of the joint axes (columns of the Jacobian) are [†]

$$
\begin{aligned}
\$_1^{ref\,1} &= \{S_2C_3C_4+C_2S_4,\ -S_2S_3,\ -S_2C_3S_4+C_2C_4;\ -S_2S_3(C_4g+h),\ -S_2C_3g-(S_2C_3C_4+C_2S_4)h,\ S_2S_3S_4g\}^T \\
\$_2^{ref\,1} &= \{-S_3C_4,\ -C_3,\ S_3S_4;\ -C_3(C_4g+h),\ S_3(g+C_4h),\ C_3S_4g\}^T \\
\$_3^{ref\,1} &= \{S_4,\ 0,\ C_4; 0,\ -hS_4,\ 0\}^T \\
\$_4^{ref\,1} &= \{0,\ -1,\ 0;\ -h,\ 0,\ 0\}^T \\
\$_5^{ref\,1} &= \{0,\ 0,\ 1; 0,\ 0,\ 0\}^T \\
\$_6^{ref\,1} &= \{S_5,\ -C_5,\ 0; 0,\ 0,,\ 0\}^T \\
\$_7^{ref\,1} &= \{-C_5S_6,\ -S_5S_6,\ C_6; 0,\ 0,\ 0\}^T
\end{aligned}
\qquad (14)
$$

Decomposition of the screw set in the order, $\$ord = \{5, 6, 7, 4, 3, 2, 1\}$, yields the null reciprocal wrenches,

$$
\begin{aligned}
\mathbf{B}_5^{ref\,1} &= \{0,\ 0,\ 0;\ 0,\ 0,\ 1\}^T \\
\mathbf{B}_6^{ref\,1} &= \{0,\ 0,\ 0;\ S_5,\ -C_5,\ 0\}^T \\
\mathbf{B}_7^{ref\,1} &= \{0,\ 0,\ 0;\ -C_5S_6,\ -S_5S_6,\ 0\}^T \\
\mathbf{B}_4^{ref\,1} &= \{-h,\ 0,\ 0;\ 0,\ 0,\ 0\}^T \\
\mathbf{B}_3^{ref\,1} &= \{0,\ -S_4h,\ 0;\ 0,\ 0,\ 0\}^T \\
\mathbf{B}_2^{ref\,1} &= \{0,\ 0,\ C_3S_4g;\ 0,\ 0,\ 0\}^T \\
\mathbf{B}_1^{ref\,1} &= \{0,\ 0,\ 0;\ 0,\ 0,\ 0\}^T
\end{aligned}
\qquad (15)
$$

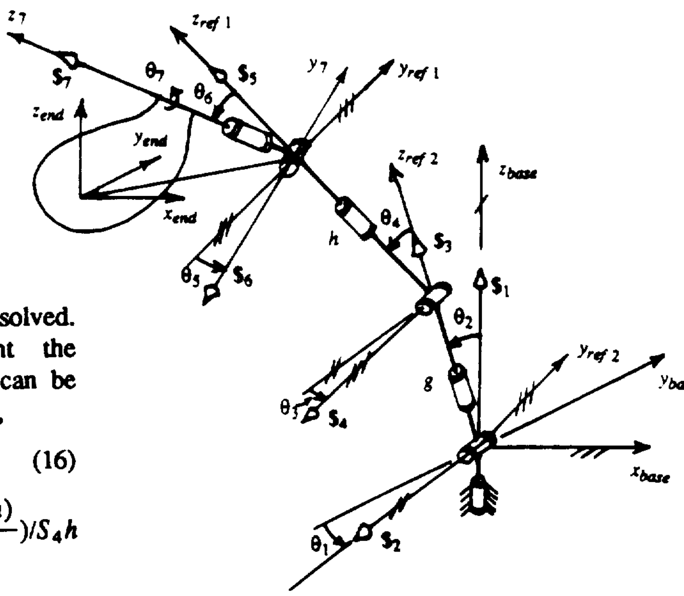


Figure 1 - 7 R Spherical-Revolute-Spherical Manipulator

A particular joint rate solution can now be resolved. Let $\mathbf{M}^{ref\,1} = \{\omega_x, \omega_y, \omega_z, v_x, v_y, v_z\}^T$ represent the required task space motion. This screw quantity can be explicitly decomposed onto the joint screws yielding,

$$\dot{q}_2 = \mathbf{M} \chi \mathbf{B}_2^{ref\,1} / \$_2^{ref\,1} \chi \mathbf{B}_2^{ref\,1} = v_z/(C_3S_4g) \qquad (16)$$

$$\dot{q}_3 = \frac{(\mathbf{M} - \dot{q}_2\$_2^{ref\,1}) \chi \mathbf{B}_3^{ref\,1}}{\$_3^{ref\,1} \chi \mathbf{B}_3^{ref\,1}} = -(v_y - \frac{v_zS_3(g+C_4h)}{C_3S_4g})/S_4h$$

$\dot{etc.}$

where $\dot{q}_i \equiv \dot{\theta}_i$

Alternatively efficient customized code (ignoring zero (0) operations and one (1) multiplications) can be produced at this point for the particular joint rate solution. The operations required for such a solution are; $\dot{q}_2$: $1\times$ and $0+$, $\dot{q}_3$: $6\times$ and $5+$, $\dot{q}_4$: $3\times$ and $2+$, $\dot{q}_7$: $3\times$ and $2+$, $\dot{q}_6$: $5\times$ and $4+$, $\dot{q}_5$: $0\times$ and $0+$, for a total of $18\times$ and $13+$. No computational costs are involved in finding the reciprocal wrenches once the Jacobian screw coordinates (equation (14)) are known.

The first joint axis for this order of decomposition corresponds to the redundant screw. Decomposing the screw coordinates of this joint yields the null-space basis,

$$
[\mathbf{a}] = d_{j7} = \begin{bmatrix} 1 \\ -S_2S_3/C_3 \\ (-S_2g-S_2C_4h-C_2C_3S_4h)/(C_3S_4h) \\ 0 \\ (S_2C_4S_6g+S_2S_4C_5C_6g+S_2S_6h)/(C_3S_4S_6h) \\ S_2S_5g/(C_3h) \\ -S_2C_5g/(C_3S_6h) \end{bmatrix} \qquad (17)
$$

[†] Details are included as Appendix 1, $S_i \equiv \mathrm{Sin}(\theta_i)$, $C_i \equiv \mathrm{Cos}(\theta_i)$

The expressions in the null-space basis indicate that $\$_4$ has no component in the null (is linearly independent), and therefore in general (special configurations excepted) joints $\$_1$, $\$_2$, $\$_3$, $\$_5$, $\$_6$ and $\$_7$ as a group have one degree of redundancy. This basis can be utilized in the optimization of the joint rates (e.g. equation (11)).

Special configuration identification

Conditions which cause a normally non-zero wrench, $\mathbf{B}_i$, to become null correspond to special configurations of a manipulator. These special configurations may correspond to a linear dependency within a "redundant group" of joints causing the joint initially chosen to be last (e.g. joint 1 in the above decomposition) to become linearly independent and hence unsuitable as the "redundant" joint. In this case reordering the screws with one of the linearly dependent joints as the redundant joint will yield a complete set of $[\mathbf{B}]$.

The special configuration may also correspond to further joint linear dependency, (i.e., an increase in the dimension of the null-space of $[\mathbf{J}]$). In this case, reordering of the decomposition will find $r<6$ non-zero wrenches, and there will exist a set of linearly independent wrenches, $\mathbf{W}_i$, $i=1,6-r$, reciprocal to the screws of $[\mathbf{J}]$, where r is the rank of $[\mathbf{J}]$. The manipulator will not be able to instantaneously produce motions having non-zero rates of work subject to $\mathbf{W}_i$. This corresponds to a loss of a degree(s) of instantaneous end effector motion capability, and is commonly referred to as a *singularity*. In [9] the authors present a scheme for instantaneously planning "optimal" alternative motion specifications satisfying the required reciprocity with $\mathbf{W}_i$, for manipulators at or near singular configurations.

The above decomposition demonstrated that for the 7R manipulator, typically any one of $\$_1$, $\$_2$, $\$_3$, $\$_5$, $\$_6$ and $\$_7$ could be chosen to represent the redundancy (the "redundant joint") of the manipulator. Furthermore, since six typically non-zero $\mathbf{B}$ values were found, the Jacobian was seen to normally be of full rank.

Examination of the wrenches of equation (15) reveal null $\mathbf{B}$ values occur if $C_3 = 0$, $S_4 = 0$ or $S_6 = 0$. If $C_3 = 0$, then $\$_2$, $\$_3$, $\$_5$, $\$_6$ and $\$_7$ become linearly dependent causing $\$_1$ to be unsuitable choice for redundant joint. Similarly if $S_6 = 0$ then $\$_5$ and $\$_7$ become linearly dependent, again rendering $\$_1$ as an unsuitable choice for redundant joint. Reordering of the decomposition (performed below) in both of these cases will find six non-zero $\mathbf{B}$ values, indicating that these configurations do not correspond to singularities. Reordering the decomposition for $S_4 = 0$, finds only five non-zero wrenches indicating a singular configuration. This case and multiple joint displacement conditions leading to loss of task space freedom are considered later.

A second screw decomposition

A decomposition order having $\$_5$ or $\$_7$ as the final joint axis screw coordinates to be decomposed would be suitable for either $C_3 = 0$ or $S_6 = 0$. It is convenient to reference the screws with respect to a frame located at the base spherical group of joints for such a decomposition. Consider the reference frame $ref\,2$ illustrated in Figure 1, where $z_{ref\,2}$ is in the direction of $\$_3$, and $y_{ref\,2}$ is in the opposite direction to that of $\$_4$. The joint screw coordinates with respect to this frame of reference are,

$$\begin{aligned}
\$_1^{ref\,2} &= \{S_2C_3,\ -S_2S_3,\ C_2;\ 0,\ 0,\ 0\}^T\\
\$_2^{ref\,2} &= \{-S_3,\ -C_3,\ 0;\ 0,\ 0,\ 0\}^T\\
\$_3^{ref\,2} &= \{0,\ 0,\ 1;\ 0,\ 0,\ 0\}^T\\
\$_4^{ref\,2} &= \{0,\ -1,\ 0;\ g,\ 0,\ 0\}^T\\
\$_5^{ref\,2} &= \{-S_4,\ 0,\ C_4;\ 0,\ -S_4g,\ 0\}^T\\
\$_6^{ref\,2} &= \{C_4S_5,\ -C_5,\ S_4S_5;\ C_5(g+C_4h),\ S_5(C_4g+h),\ S_4C_5h\}^T\\
\$_7^{ref\,2} &= \{-C_4C_5S_6-S_4C_6,\ -S_5S_6,\ -S_4C_5S_6+C_4C_6;\ S_5S_6(g+C_4h),\ -g(C_4C_5S_6+S_4C_6)-C_5S_6h,\ S_4S_5S_6h\}^T
\end{aligned} \tag{18}$$

Decomposition of the screw set in the order, $\$ord = \{3, 2, 1, 4, 5, 6, 7\}$, yields the null reciprocal wrenches,

$$\begin{aligned}
&\mathbf{B}_3^{ref\,2} = \{0,0,0;0,0,1\}^T,\quad \mathbf{B}_2^{ref\,2} = \{0,0,0;-S_3,-C_3,0\}^T,\quad \mathbf{B}_1^{ref\,2} = \{0,0,0;S_2C_3,-S_2S_3,0\}^T\\
&\mathbf{B}_4^{ref\,2} = \{g,0,0;0,0,0\}^T,\quad \mathbf{B}_5^{ref\,2} = \{0,-S_4g,0;0,0,0\}^T,\quad \mathbf{B}_6^{ref\,2} = \{0,0,S_4C_5h;0,0,0\}^T\\
&\mathbf{B}_7^{ref\,2} = \{0,0,0;0,0,0\}^T
\end{aligned} \tag{19}$$

For this order of decomposition $\$_7$ is the redundant joint, and the null-space basis yielded by the decomposition is the result of equation (17) multiplied by $-C_3S_6h/(S_2C_5g)$.

Examination of the NRP wrenches of equation (19) reveals that null **B** values occur if $S_2 = 0$, $S_4 = 0$ or $C_5 = 0$. For the conditions $S_2 = 0$ or $C_5 = 0$ the results of the first decomposition (equations (14)-(17)) are suitable for use. Hence the above two decompositions of screw coordinates provide expressions for the inverse velocity solution for all non-singular configurations of the manipulator.

Singular configurations

The only single joint displacement condition which causes a loss of motion degree of freedom is $S_4 = 0$, corresponding to a straight arm configuration. In this case decomposition of the screw coordinates, regardless of the order chosen, will yield only five reciprocal wrenches. That is, the rank of the Jacobian is five, the dimension of its null-space is two, and there is a screw, **W**, (a 1-system), reciprocal to all of the joint screws. The manipulator instantaneously cannot produce a motion having a non-zero reciprocal product with **W**. A decomposition is performed below for $S_4 = 0$, generating analytical expressions for [**B**] and $[a]$, and **W** is identified.

Further examination of the wrenches of equations (15) and (19), and the joint screw coordinates of equations (14) and (18), indicate that motion degeneracies (singularities) are also present for multiple joint displacement conditions (e.g.: $S_2 = 0$ and $C_3 = 0$; $S_6 = 0$ and $C_5 = 0$; and $S_2 = 0$ and $S_6 = 0$). Decompositions for these cases are also performed below. The four cases are illustrated in Figure 2.

$\rightarrow S_4 = 0$

Using *ref* 1 as the reference the screw coordinates for $S_4 = 0$ reduce to:

$$\begin{aligned}
\$_1^{ref\,1} &= \{S_2C_3,\ -S_2S_3,\ C_2;\ -S_2S_3(g+h),\ -S_2C_3(g+h),\ 0\}^T \\
\$_2^{ref\,1} &= \{-S_3,\ -C_3,\ 0;\ -C_3(g+h),\ S_3(g+h),\ 0\}^T \\
\$_3^{ref\,1} &= \{0,\ 0,\ 1;\ 0,\ 0,\ 0\}^T \\
&\$_4^{ref\,1},\ \$_5^{ref\,1},\ \$_6^{ref\,1},\ \$_7^{ref\,1} \quad \text{as in Equation (14)}
\end{aligned} \tag{20}$$

Decomposing the screw coordinates in the order $\$ord = \{5, 6, 7, 4, 2, 1, 3\}$, (shifting 3 to the end when a null $\mathbf{B}_3^{ref\,1}$ is found), yields:

$$\begin{aligned}
&\mathbf{B}_5^{ref\,1},\ \mathbf{B}_6^{ref\,1},\ \mathbf{B}_7^{ref\,1},\ \mathbf{B}_4^{ref\,1} \quad \text{as in Equation (15)} \\
&\mathbf{B}_2^{ref\,1} = \{0,\ S_3(g+h),\ 0;\ 0,\ 0,\ 0\}^T \\
&\mathbf{B}_1^{ref\,1} = \{0, 0, 0; 0, 0, 0\}^T, \quad \mathbf{B}_3^{ref\,1} = \{0, 0, 0; 0, 0, 0\}^T
\end{aligned} \tag{21}$$

The Jacobian null-space basis generated by the decomposition of $\$_1^{ref\,1}$ and $\$_3^{ref\,1}$ is:

$$[a] = \begin{bmatrix} 1 & S_2C_3/S_3 & 0 & 0 & -(C_2S_3S_6-S_2S_5C_6)/S_3S_6 & -S_2C_5/S_3 & -S_2S_5/S_3S_6 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}^T \tag{22}$$

A screw reciprocal to $\$_1 \cdots \$_7$ is $\mathbf{W}^{ref\,1} = \{0, 0, 1; 0, 0, 0\}^T$ indicating that a point on the end effector coinciding with the origin of *ref* 1, can have no translational velocity in the $z_{ref\,1}$ direction (the arm direction).

$\rightarrow S_2 = 0$ and $C_3 = 0$

Again using *ref* 1 as the reference the screw coordinates for $S_2 = 0$ and $C_3 = 0$ reduce to:

$$\begin{aligned}
\$_1^{ref\,1} &= \{S_4,\ 0,\ C_4;\ 0,\ -S_4h,\ 0\}^T \\
\$_2^{ref\,1} &= \{-C_4,\ 0,\ S_4;\ 0,\ g+C_4h,\ 0\}^T \\
&\$_3^{ref\,1},\ \$_4^{ref\,1},\ \$_5^{ref\,1},\ \$_6^{ref\,1},\ \$_7^{ref\,1} \quad \text{as in Equation (14)}
\end{aligned} \tag{23}$$

Decomposing the screw coordinates in the order $\$ord = \{5, 6, 7, 4, 3, 2, 1\}$, yields:

$$\begin{aligned}
&\mathbf{B}_5^{ref\,1},\ \mathbf{B}_6^{ref\,1},\ \mathbf{B}_7^{ref\,1},\ \mathbf{B}_4^{ref\,1},\ \mathbf{B}_3^{ref\,1} \quad \text{as in Equation (15)} \\
&\mathbf{B}_2^{ref\,1} = \{0, 0, 0; 0, 0, 0\}^T, \quad \mathbf{B}_1^{ref\,1} = \{0, 0, 0; 0, 0, 0\}^T
\end{aligned} \tag{24}$$

The Jacobian null-space basis generated by the decomposition of $\$_1^{ref\,1}$ and $\$_2^{ref\,1}$ is:

$$[a] = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & (g+C_4h)/S_4h & 0 & -(S_4C_5C_6g+C_4S_6g+S_6h)/S_4S_6h & -S_5g/h & C_5g/S_6h \end{bmatrix}^T \tag{25}$$

A screw reciprocal to $\$_1 \cdots \$_7$ is $\mathbf{W}^{ref\,1} = \{0, 0, 1; 0, 0, 0\}^T$.

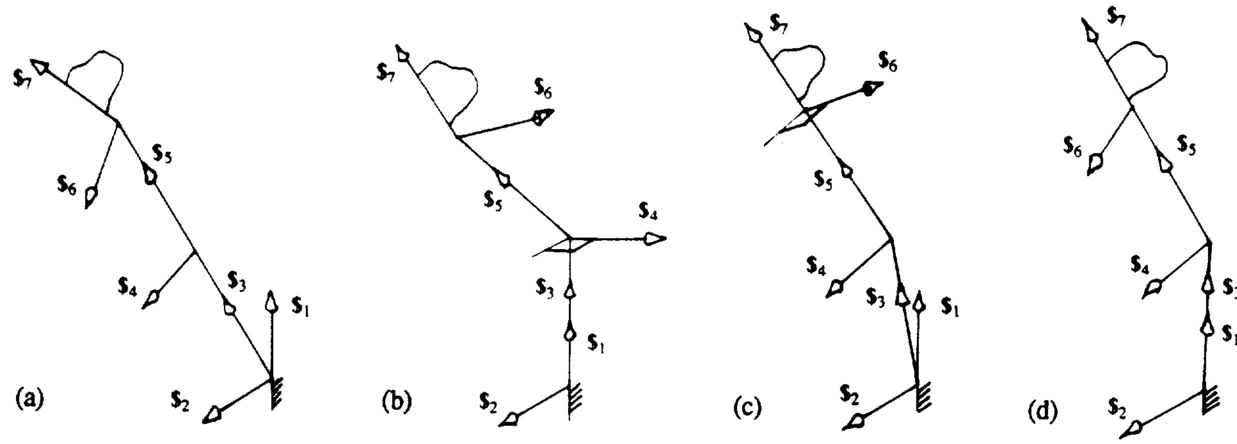


Figure 2 - Singular Configurations, (a) $S_4 = 0$, (b) $S_2 = 0$ and $C_3 = 0$, (c) $S_6 = 0$ and $C_5 = 0$, (d) $S_2 = 0$ and $S_6 = 0$

$\rightarrow S_6 = 0$ and $C_5 = 0$

Using *ref* 2 as the reference the screw coordinates for $S_6 = 0$ and $C_5 = 0$ reduce to:

$$\begin{aligned} &\$_1^{ref\,2}, \$_2^{ref\,2}, \$_3^{ref\,2}, \$_4^{ref\,2}, \$_5^{ref\,2} \quad \text{as in Equation (18)} \\ &\$_6^{ref\,2} = \{C_4, 0, S_4; 0, C_4g + h, 0\}^T \\ &\$_7^{ref\,2} = \{-S_4, 0, C_4; 0, -S_4g, 0\}^T \end{aligned} \tag{26}$$

Decomposing the screw coordinates in the order *$ord* = {3, 2, 1, 4, 5, 6, 7}, yields:

$$\begin{aligned} &\mathbf{B}_3^{ref\,2}, \mathbf{B}_2^{ref\,2}, \mathbf{B}_1^{ref\,2}, \mathbf{B}_4^{ref\,2}, \mathbf{B}_5^{ref\,2} \quad \text{as in Equation (19)} \\ &\mathbf{B}_6^{ref\,2} = \{0, 0, 0; 0, 0, 0\}^T, \quad \mathbf{B}_7^{ref\,2} = \{0, 0, 0; 0, 0, 0\}^T \end{aligned} \tag{27}$$

The Jacobian null-space basis generated by the decomposition of $\$_6^{ref\,2}$ and $\$_7^{ref\,2}$ is:

$$[a] = \begin{bmatrix} C_3h/S_2g & -S_3h/g & -(S_2S_5g+S_2C_4S_5h+C_2C_3S_4h)/S_2S_4g & 0 & (C_4S_5g + h)/S_4g & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}^T \tag{28}$$

A screw reciprocal to $\$_1 \cdots \$_7$ is $\mathbf{W}^{ref\,2} = \{0, 0, 1; 0, 0, 0\}^T$.

$\rightarrow S_2 = 0$ and $S_6 = 0$

Using *ref* 1 as the reference the screw coordinates for $S_6 = 0$ and $C_5 = 0$ reduce to:

$$\begin{aligned} &\$_1^{ref\,1} = \{S_4, 0, C_4; 0, -S_4h, 0\}^T \\ &\$_2^{ref\,1}, \$_3^{ref\,1}, \$_4^{ref\,1}, \$_5^{ref\,1}, \$_6^{ref\,1} \quad \text{as in Equation (14)} \\ &\$_7^{ref\,1} = \{0, 0, 1; 0, 0, 0\}^T \end{aligned} \tag{29}$$

Decomposing the screw coordinates in the order *$ord* = {5, 6, 4, 3, 2, 1, 7}, (where 7 is shifted to the end due to a null $\mathbf{B}_7^{ref\,1}$ being found), yields:

$$\begin{aligned} &\mathbf{B}_5^{ref\,1} = \{0, 0, 0; 0, 0, 1\}^T, \quad \mathbf{B}_6^{ref\,1} = \{0, 0, 0; S_5, -C_5, 0\}^T, \quad \mathbf{B}_4^{ref\,1} = \{-h, 0, 0; -C_5S_5, -S_5^2, 0\}^T \\ &\mathbf{B}_3^{ref\,1} = (1/(h^2 + S_5^2))\{-hS_4C_5S_5, -hS_4(h^2 + S_5^2), 0; S_4C_5^2h^2, S_4C_5S_5h^2, 0\}^T \\ &\mathbf{B}_2^{ref\,1} = \{-C_3C_4g-C_3h+\alpha_{42}h, S_3g+S_3C_4h+\alpha_{32}hS_4, C_3S_4g; -S_3C_4-\alpha_{32}S_4-\alpha_{62}S_5, -C_3+\alpha_{42}+\alpha_{62}C_5, 0\}^T \end{aligned} \tag{30}$$

$\mathbf{B}_1^{ref\,1} = \{0, 0, 0; 0, 0, 0\}^T, \quad \mathbf{B}_7^{ref\,1} = \{0, 0, 0; 0, 0, 0\}^T$

where α_{32}, α_{42}, and α_{62} are given in Appendix 2

The Jacobian null-space basis generated by the decomposition of $\$_7^{ref\,1}$ and $\$_1^{ref\,1}$ is:

$$[a] = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}^T \tag{31}$$

A screw quantity (not normalized) reciprocal to $\$_1 \cdots \$_7$ is $\mathbf{W}^{ref\,1} = \{ -S_5, C_5, -C_4S_5/S_4 - S_3C_5/C_3S_4; C_5h, S_5h, 0\}^T$.

Further Conditions of Degeneracy

Each of the above conditions correspond to a single loss of task space motion freedom. Further multiple joint displacement conditions leading to the loss of more than one motion degree of freedom can be observed by examination of the reciprocal wrenches of equations (21), (24), (27), and (30). These multiple conditions include: $S_4 = 0$, $S_3 = 0$, and $S_2 = 0$; and $S_4 = 0$, $S_5 = 0$, and $S_6 = 0$; both leading to a loss of two degrees of task space freedom. The multiple condition $S_2 = 0$, $S_3 = 0$, $S_4 = 0$, $S_5 = 0$, and $S_6 = 0$ results in a loss of three task space motion degrees of freedom. The manipulator can never lose more than three degrees of freedom.

6. Conclusions

The general method for the inverse solution of manipulator joint velocities based on a decomposition of screw coordinates presented in this work, has the advantage of inherently identifying a basis for the null-space of the Jacobian. The null-space basis has been shown to be useful for the resolution of locally optimum joint velocities, generating direct solutions for the optimal joint rates in terms of a pseudo-inverse of a weighting of the basis, for quadratic joint rate objective functions. The inverse velocity solution method has been demonstrated to be suitable for the derivation of analytical expressions for manipulators by application to a specific example.

Efficient resolution of the joint velocities for the spherical-revolute-spherical (7R) manipulator is possible using the analytical expressions derived in this work. The identification of the special configurations of the manipulator, and the characterization of singular configurations, make this a complete solution.

References

[1] Hollerbach, J.M., Suh, K.C., "Redundancy Resolution of Manipulators through Torque Optimization", *IEEE J. of Rob. & Auto., Vol. RA-3, No. 4, August 1987, pp. 308-316.*

[2] Kazerounian, K., Wang, Z., "Global versus Local Optimization in Redundancy Resolution of Robotic Manipulators", *Int. J. of Rob. Reas.*, Vol. 7, No. 5, October 1988, pp. 3-12.

[3] Baker, D.R., Wampler II, C.W., "On the Inverse Kinematics of Redundant Manipulators", *Int. J. of Rob. Res.* Vol. 7, No. 2, April 1988, pp. 3-21.

[4] Baillieul, J., "A Constraint Oriented Approach to Inverse Problems for Kinematically Redundant Manipulators, *IEEE Int. Conf. on Rob. and Aut.*, 1987, pp. 1827-1833.

[5] Sugimoto, K., "Determination of Joint Velocities of Robots Using Screws", *J. of Mech., Trans., Auto. in Des.*, Vol. 106, June 1984, pp. 222-227.

[6] Hunt, K., "Robot Kinematics - A Compact Analytical Inverse Solution for Velocities", *J. of Trans., Mech., Auto. in Des.*, Vol. 109, March 1987, pp. 42-49.

[7] Stanisic, M., M., Pennock, G. R., Krousgrill, C. M., "Inverse Velocity and Acceleration Solutions of Serial Robot Arm Subassemblies Using the Canonical Coordinate System, *Int. J. of Rob. Res.* Vol. 7, No. 1, Feb. 1988, pp. 29-41.

[8] Hollerbach, J.M., "Optimum Kinematic Design for a Seven Degree of Freedom Manipulator", *2nd Int. Symp. Rob. Res.*, 1985, pp. 215-222.

[9] Podhorodeski, R.P., Goldenberg, A.A., Fenton, R.G., "A Screw Decomposition Based Method for the Inverse Solution of the Instantaneous Kinematics of Manipulators", *Int. Meet. "Advances in Robot Kinematics"*, Ljubljana, 1988, pp. 103-112.

[10] Podhorodeski, R.P., Goldenberg, A.A., Fenton, R.G., "Resolving Redundant Manipulator Joint Rates and Identifying Special Arm Configurations using Jacobian Null-Space Bases", *submitted to the IEEE J. of Rob. and Aut.*

[11] Ball, R.S., *Theory of Screws: A Study in the Dynamics of a Rigid Body*, Hodges, Foster, and Co., 1876.

[12] Hunt, K., *Kinematic Geometry of Mechanisms*, Oxford Press, 1978.

[13] Strang, G., *Linear Algebra and Its Applications*, Academic Press, New York, 1976.

[14] Paul, R.P., *Robot Manipulators: Mathematics, Programming, and Control*, The MIT Press, Massachusetts, 1981.

Appendix 1 - Generation of the Screw Coordinate Models

In terms of Denevit and Hartenberg parameters[14] the 7R manipulator links can be described as tabulated in Table 1. The following rotation matrices can be found,

$$[\mathbf{R}]_1 = \begin{bmatrix} C_1 & 0 & S_1 \\ S_1 & 0 & -C_1 \\ 0 & 1 & 0 \end{bmatrix}, \quad [\mathbf{R}]_2 = \begin{bmatrix} C_2 & 0 & -S_2 \\ S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix}, \quad [\mathbf{R}]_3 = \begin{bmatrix} C_3 & 0 & S_3 \\ S_3 & 0 & -C_3 \\ 0 & 1 & 0 \end{bmatrix},$$

$$[\mathbf{R}]_4 = \begin{bmatrix} C_4 & 0 & -S_4 \\ S_4 & 0 & C_4 \\ 0 & -1 & 0 \end{bmatrix}, \quad [\mathbf{R}]_5 = \begin{bmatrix} C_5 & 0 & S_5 \\ S_5 & 0 & -C_5 \\ 0 & 1 & 0 \end{bmatrix}, \quad [\mathbf{R}]_6 = \begin{bmatrix} C_6 & 0 & -S_6 \\ S_6 & 0 & C_6 \\ 0 & -1 & 0 \end{bmatrix},$$

$$[\mathbf{R}]_7 = \begin{bmatrix} C_7 & -S_7 & 0 \\ S_7 & C_7 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Link	Variable	Twist	a	d
1	θ_1	90.	0.	0.
2	θ_2	-90.	0.	0.
3	θ_3	90.	0.	g
4	θ_4	-90.	0.	0.
5	θ_5	90.	0.	h
6	θ_6	-90.	0.	0.
7	θ_7	0.	0.	0.

Table 1 - D & H Parameters

The screw coordinates of the joint axes are expressed with respect to the reference frames, *ref* 1 and *ref* 2, in terms of the rotation matrices in Tables 2 and 3 respectively. The operator $z([\mathbf{A}])$ is defined as $z([\mathbf{A}]) = [\mathbf{A}]\{0, 0, 1\}^T$.

A screw quantity, $\mathbf{S}^s$, known in frame s, can be transformed and expressed in frame, f, by

$$\mathbf{S}^f = \{[\mathbf{R}]_{fs}\mathbf{s}^s; [\tilde{p}^f]_{fs}[\mathbf{R}]_{fs}\mathbf{s}^s + [\mathbf{R}]_{fs}\mathbf{s}_o^s\} \equiv [\mathbf{T}]_{fs}\mathbf{S}^s \quad \text{where } [\mathbf{T}]_{fs} = \begin{bmatrix} [\mathbf{R}]_{fs} & [0] \\ [\tilde{p}^f]_{fs}[\mathbf{R}]_{fs} & [\mathbf{R}]_{fs} \end{bmatrix}$$

with $[\mathbf{R}]_{fs}$ the 3×3 rotation matrix describing the orientation of the frame s with respect to the frame f, $[\tilde{p}^f]_{fs}$ a 3×3 skew symmetric (cross product matrix) of the location of the s reference origin with respect to the f origin expressed in the f reference coordinates, and [0] a 3×3 null matrix. Rotation matrices and displacement vectors required for transformation of an end effector velocity screw known with respect to an inertially oriented, end effector tip located frame, to the reference frames are

$$[\mathbf{R}]_{ref\,1\to end} = ([\mathbf{R}]_1[\mathbf{R}]_2[\mathbf{R}]_3[\mathbf{R}]_4)^T, \quad \{p^{ref\,1}_{ref\,1\to end}\} = [\mathbf{R}]_5[\mathbf{R}]_6[\mathbf{R}]_7\{x\}^{end^T}$$

$$[\mathbf{R}]_{ref\,2\to end} = ([\mathbf{R}]_1[\mathbf{R}]_2[\mathbf{R}(\theta_3)]_z)^T,$$

$$\{p^{ref\,2}_{ref\,2\to end}\} = \{0, 0, g\}^T + [\mathbf{R}(90)]_x[\mathbf{R}]_4\{0, 0, h\}^T + [\mathbf{R}(90)][\mathbf{R}]_4[\mathbf{R}]_5[\mathbf{R}]_6[\mathbf{R}]_7\{x\}^{end^T}$$

where $\{x\}^{end^T}$ is the tip location with respect to an end effector oriented wrist located frame.

i	$\$_i^{ref\,1}$	
	$\{\$_i^{ref\,1}\}^T$	$\{\$_o^{ref\,1}{}_i\}^T$
5	$\{0, 0, 1\}$	$\{0, 0, 0\}$
6	$z([\mathbf{R}]_5)$	$\{0, 0, 0\}$
7	$z([\mathbf{R}]_5[\mathbf{R}]_6)$	$\{0, 0, 0\}$
4	$z([\mathbf{R}]_4^T)$	$\{x^{ref\,1}_{elbow}\} \times \{\$_4^{ref\,1}\}$
3	$z([\mathbf{R}]_4^T[\mathbf{R}]_3^T)$	$\{x^{ref\,1}_{base}\} \times \{\$_3^{ref\,1}\}$
2	$z([\mathbf{R}]_4^T[\mathbf{R}]_3^T[\mathbf{R}]_2^T)$	$\{x^{ref\,1}_{base}\} \times \{\$_2^{ref\,1}\}$
1	$z([\mathbf{R}]_4^T[\mathbf{R}]_3^T[\mathbf{R}]_2^T[\mathbf{R}]_1^T)$	$\{x^{ref\,1}_{base}\} \times \{\$_1^{ref\,1}\}$

where $\{x^{ref\,1}_{elbow}\}=\{0, 0, -h\}^T$
and $\{x^{ref\,1}_{base}\}=\{x^{ref\,1}_{elbow}\}+[\mathbf{R}]_4^T[\mathbf{R}]_3^T\{0, 0, -g\}^T$

Table 2 - 7R Joint Screw Coordinates (*ref* 1 reference)

i	$\$_i^{ref\,2}$	
	$\{\$_i^{ref\,2}\}^T$	$\{\$_o^{ref\,2}{}_i\}^T$
3	$\{0, 0, 1\}$	$\{0, 0, 0\}$
2	$z([\mathbf{R}(\theta_3)]_z^T[\mathbf{R}]_2^T)$	$\{0, 0, 0\}$
1	$z([\mathbf{R}(\theta_3)]_z^T[\mathbf{R}]_2^T[\mathbf{R}]_1^T)$	$\{0, 0, 0\}$
4	$z([\mathbf{R}(90)]_x)$	$\{x^{ref\,2}_{elbow}\} \times \{\$_4^{ref\,2}\}$
5	$z([\mathbf{R}(90)]_x[\mathbf{R}]_4)$	$\{x^{ref\,2}_{wrist}\} \times \{\$_6^{ref\,2}\}$
6	$z([\mathbf{R}(90)]_x[\mathbf{R}]_4[\mathbf{R}]_5)$	$\{x^{ref\,2}_{wrist}\} \times \{\$_7^{ref\,2}\}$
7	$z([\mathbf{R}(90)]_x[\mathbf{R}]_4[\mathbf{R}]_5[\mathbf{R}]_6)$	$\{x^{ref\,2}_{wrist}\} \times \{\$_7^{ref\,2}\}$

where $\{x^{ref\,2}_{elbow}\}=\{0, 0, g\}^T$
and $\{x^{ref\,2}_{base}\}=\{x^{ref\,2}_{elbow}\}+[\mathbf{R}(90)]_x[\mathbf{R}]_4\{0, 0, h\}^T$

Table 3 - 7R Joint Screw Coordinates (*ref* 2 reference)

Appendix 2 - α_{32}, α_{42}, α_{62} for $S_2 = 0, S_6 = 0$

$$\alpha_{32} = \frac{-hS_3C_4S_4(1 + h^2) - g(S_3S_4S_5^2 - C_3C_4S_4C_5S_5 + h^2S_3S_4)}{h(S_4^2 + h^2)}$$

$$\alpha_{42} = \frac{C_3C_4gh + C_3h^2 + S_3C_4C_5S_5 + \alpha_{32}S_4C_5S_5 + C_3S_5^2}{S_5^2 + h^2}, \quad \alpha_{62} = C_3C_5 - S_3C_4S_5 - \alpha_{32}S_4S_5 - \alpha_{42}C_5$$

MAN-MACHINE SYSTEMS

N90-29007

ADJUSTABLE IMPEDANCE, FORCE FEEDBACK AND COMMAND LANGUAGE AIDS FOR TELEROBOTICS
(Parts 1-4 of an 8-Part MIT Progress Report)

Thomas B. Sheridan, G.Jagganath Raju, Forrest T. Buzan, Wael Yared and Jong Park

Man-Machine Systems Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

This paper summarizes four separate projects recently completed or in progress at the MIT Man-Machine Systems Laboratory. Four others are described in a companion paper in Volume 3.

1. OPERATOR ADJUSTABLE IMPEDANCE IN MASTER-SLAVE TELEOPERATION - G. Jagganath Raju

Abstract. A 2-port impedance network model of a single degree of freedom remote manipulation system in which a human operator at the master port interacts with a task object at the slave port in a remote location is presented. The design of the network involves the selection of feedback gains for the servomechanisms that transmit motion and force information from one port of the 2-port to the other in both directions. The methodology proposed here allows for this selection to be based on both stability requirements and specifications of desired port impedances, given models of the task and the human operator. The resulting design guidelines guarantee stability for any passive task object at the slave port and any passive human impedance at the master port.

Introduction. In remote manipulation tasks that use master-slave manipulators, the ability to successfully execute the task, and the performance level of the human operator are dependent on the impedance characteristics of the task object, the master-slave system, and the operator's arm. If the master- slave system were designed such that its impedance characteristics could be adjusted by the human operator while doing the task, it may be possible to improve the performance level of the operator significantly.

When humans manipulate objects in their environment, two senses that are extensively used are vision and the "muscle senses" that mediate kinesthesis and proprioception. The assumption made here is that the objective of the manipulation task is to identify and/or alter the location of an object in the environment. The term "telepresence" reflects the concept of transporting the human operator not in body but in sensation to the remote location. Though the "skin senses" may be blocked by a telemanipulator mechanism and/or the telecommunication channel, the "muscle senses" and vision may be replaced with high fidelity transmission channels of vision and force/displacement. In reality, owing to limitations imposed by the environment (distances, transmission medium) and technology (sensor resolution, transmission bandwidth, time delays) the transmission signals are degraded and have to be enhanced or compensated for in some way to be of real value.

A force feedback channel can provide the operator with values of the force levels in the interaction with the object, displayed on a screen, or better still convert these measurements back to a force level through a servo-actuator to redirect a sense of "feel" to the operator. The force transmission channel in the forward direction transmits the forces that the human operator would have imposed on the task had she been able to manipulate the object directly.

The approach adopted here is to model the MSM as a 2-port network with the operator-master arm interface designated as the master port and the task-slave arm as the slave port. The dynamics of the force transmission channels and the responses of the MSM at the master and slave ports can then be characterized by a set of "network functions". The sub-systems that comprise a model of a remote manipulation system are depicted in Fig. 1.

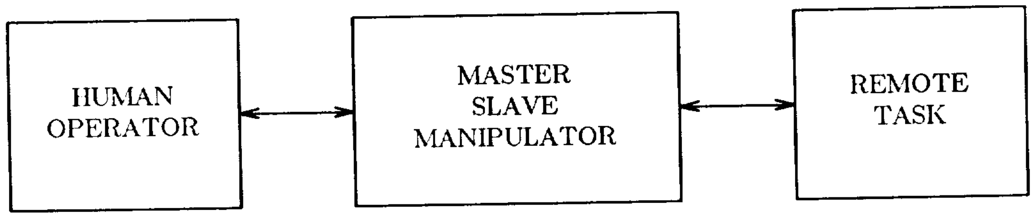


Figure 1. Remote manipulation.

PRECEDING PAGE BLANK NOT FILMED

The three sub-systems are: the human operator, who manipulates the master device of the MSM in a manner that results in the slave device of the MSM acting on the task to achieve the desired goal; a MSM which transmits forces and motion between the human operator and the remote task; the task object in the remote environment that is being manipulated by the slave device of the MSM.

What constitutes an "ideal" telemanipulator? Intuitively a reasonable response would be: An ideal telemanipulator is one that provides complete transparency of the interface. In other words the operator feels as if the task object were being handled directly. For force-feedback systems Handlykken [1] suggests that this can be represented by an infinitely stiff and weightless mechanical connection between the end-effector of the master arm and the slave arm.

Yokokhoji and Yoshikawa [2] argue that the ideal response of a remote manipulator system is one in which the position and force responses of the master arm are systematically equal to the responses of position and force when the operator manipulates the object directly. But from a human factors point of view, Vertut [3] suggests that the operator may sometimes get tired of holding a constant weight, and reports implementing a system with continuous variation of the force feedback ratio to reduce fatigue and improve precision. Indeed it is not all that clear that the highest level of force-feedback is universally beneficial in executing all tasks.

Hill [4] reports that in the classic peg-in-hole insertion task, the insertion phase shows little difference with or without force feedback. Bejczy and Handlykken [5] report from experimental studies that there seems to be an optimal combination of the force-feedback gain (from slave to master) and the feedforward gain (from master to slave) and that this combination may be task dependent.

It appears that as yet there is no consensus regarding an universal idealization of a remote manipulator system. Indeed to some extent the hypothesis in this work is partly motivated by the non-existence of such a universal standard, since this brings out the necessity of designing an adjustable system. An additional implication of this statement may be that the "ideal" telemanipulator is an "adjustable" one.

At the input or master port, the MSM interacts with the operator; at the output or slave port it interacts with the task object in the remote environment, as illustrated in Fig. 2.

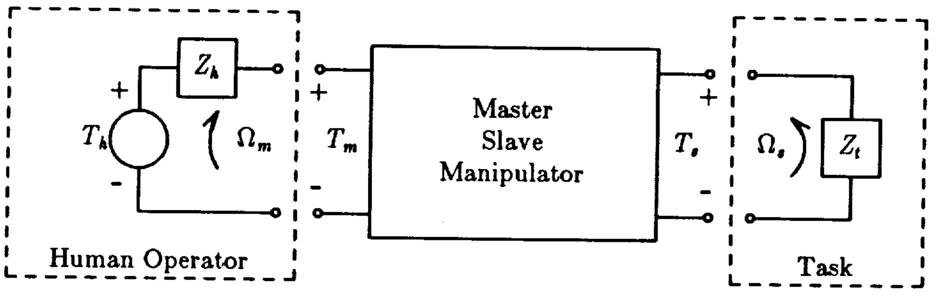


Figure 2. Electrical network model of master-slave manipulator.

Electrical network model of master-slave manipulator. At each port (master and slave) the co-energy variables of interest are the effort variables (torques T_m and T_s) and the flow variables (velocities Ω_m and Ω_s). Either of the co-energy variables at each port may be chosen to be the independent variable, and the value of the dependent variable is then determined by the parameters that characterize the MSM. If the flow variables are considered to be the inputs, the MSM can be represented by an impedance matrix [Z(s)] such that

$$\underline{T}(s) = [Z(s)]\,\underline{\Omega}(s) \qquad \underline{T}(s) = \begin{Bmatrix} T_m(s) \\ T_s(s) \end{Bmatrix} \qquad \underline{\Omega}(s) = \begin{Bmatrix} \Omega_m(s) \\ \Omega_s(s) \end{Bmatrix} \tag{0.1}$$

or

$$\begin{Bmatrix} T_m(s) \\ T_s(s) \end{Bmatrix} = \begin{bmatrix} Z_{11}(s) & Z_{12}(s) \\ Z_{21}(s) & Z_{22}(s) \end{bmatrix} \begin{Bmatrix} \Omega_m(s) \\ \Omega_s(s) \end{Bmatrix} \tag{0.2}$$

Two other representations of the MSM are obtained if one of the inputs at one port is an effort variable and the input at the other port is a flow variable, or vice-versa. These are commonly referred to as hybrid models of the 2-port. Since each of the alternative representations governs the same physical system, the elements of each matrix can be calculated in terms of the elements of one of the other representations. Two special cases of the 2-port that are relevant to our context are the *bilateral* 2-port and the *reciprocal* 2-port.

When both the off-diagonal elements of the 2-port matrix are non-zero, signal flow takes place in both directions (state changes at one port cause changes at the other) and the network is called a *bilateral* 2-port. If these two off-diagonal elements are equal the 2-port is termed *reciprocal.*

The design goals are:(1) The 2-port MSM has to be stable for any passive termination $Z_h(s)$ at the master port, and any passive termination $Z_t(s)$ at the slave port; (2) the port impedances, $Z_m(s)$ at the master port and $Z_s(s)$ at the slave port, have to match desired values specified by the designer. By combining the conditions for stability and the conditions to realize desired port stiffnesses, the constraints on selection of feedback gains for the servo-mechanisms, [K], are derived and used in the design process. A general approach for a n-port network is to require that the immitance matrix for the n-port network be positive real. This is equivalent to constraining the n-port described by the matrix [Z(s) or Y(s)] to be passive, and hence inherently stable.

The criteria that have to be satisfied by the impedance matrix [6] are: (1)$Z_{ij}(s)$ is real; (2) $Z_{ij}(s)$ is analytic in the RHS; (3) Poles of Z_{ij} $(j\omega)$ are simple, and the hermitian residue matrix at each of these poles is positive-semidefinite; (4)The determinant of the hermitian matrix of the impedance function is non-negative for all Ω.

The design was realized in hardware [7] by a one degree-of-freedom Master-Slave Manipulator. Using this apparatus, an experiment was conducted to check if the capability to adjust the impedance parameters of the MSM was gainful. From the results there are some interesting observations that can be made in a qualitative sense. The level of force-feedback that the operator feels is determined by the specification of the master port impedance. Force-feedback is useful in interacting with remote tasks up to a certain level, beyond which a performance index based on speed and success of task execution did not improve significantly. If other criteria such as operator fatigue, comfort and arm strength (a low force-feedback level would allow a weak operator to compress a stiff spring) are factored into the performance index, there is no apparent reason to believe that the highest level of force-feedback would be the best.

The experiment clearly established that the selection of the slave port impedance is dependent on the task characteristics. Hence a reasonable conclusion that can be reached is that the capability to adjust the impedances of the MSM is advantageous in executing tasks with widely differing characteristics or tasks that are made up of sub-tasks that have differing characteristics. Before contact with the task object the slave port impedance should below, and the master port impedance high, so that contact can be sensed but not result in the imposition of excessive force on the task object. Upon contact, the slave port impedance should be increased so that the task can be executed, and the master port impedance reduced to provide a comfortable but adequate level of force-feedback to the operator. The functional dependence of the impedances selected on the characteristics of the operator and the task is an area that can be explored further with the aid of design tools that have been developed in this work.

References

(1) Handlykken, M. and Turner, T., Control system analysis and synthesis for a six DOF universal, *Proc. IEEE Conf. Decision and Control*, 1980.
(2) Yokokohji, Y. and Yoshikawa, T., *Analyis of Maneuverability and Stability for Master-Slave System*, Automation Research Laboratory, Kyto University, Uji, Kyoto, Japan.
(3) Vertut, J., Fournier, R., Espiau, B. and Andre, G., Advances in a computer-aided bilateral system, *ANS Topical Meeting on Robotics and Remote Handling*, Gatlinburg, TN, 1984.
(4) Hill, J. W., *Study of Modeling and Evaluation of Remote Manipulation Tasks with Force Feedback*, SRI Project 7696 Final Report, SRI International, Menlo Park, CA, 94025, March 1979.
(5) Bejczy, A. K. and Handlykken, M., Experimental Results with a Six Degree-of-Freedom Force Reflecting Hand Controller, *Proceedings of the 17th Annual Conference on Manual Control*, UCLA, Los Angeles, CA, June 1981.
(6) Ghausi, M.S., *Principles and Design of Linear Active Circuits*. McGraw-Hill, Inc. 1965.
[7] Raju, G.J., *Operator Adjustable Impedance in Bilateral Remote Manipulation*, PhD Thesis, MIT, Cambridge, MA, Sept. 1988.

2. TELEOPERATORS WITH LARGE TIME DELAYS: A PREDICTIVE AID WITH FORCE REFLECTION - Forrest T. Buzan

Abstract. Delays in teleoperator control loops make "real time" control difficult. Operators tend to adopt a move-and-wait strategy (Ferrell, 1965, 1966). Noyes and Sheridan found that a "predictor display" which showed the kinematic response of the slave to the master commands helped the operator work faster and more continuously. This work addresses the extension of the predictor concept to include force feedback and dynamic modelling of the manipulator and the environment.

Predictive Operator Aid. If the manipulator commands are sent through a model of the manipulator system without the delay, the response of the model should predict the delayed output which we will receive from the actual manipulator. We then have the choice of using the real, delayed output or the predicted (model's) output; or we can use both in some manner.

Figure 3 shows the general system diagram for the open loop predictor (below) running in parallel with the actual manipulator (above). The delay is shown in the feedback. It can be placed in either or split between them with no loss of generality. Also, keep in mind that the forward loop gain, 'K', as well as the preceding summing junction "exist" within the human operator.

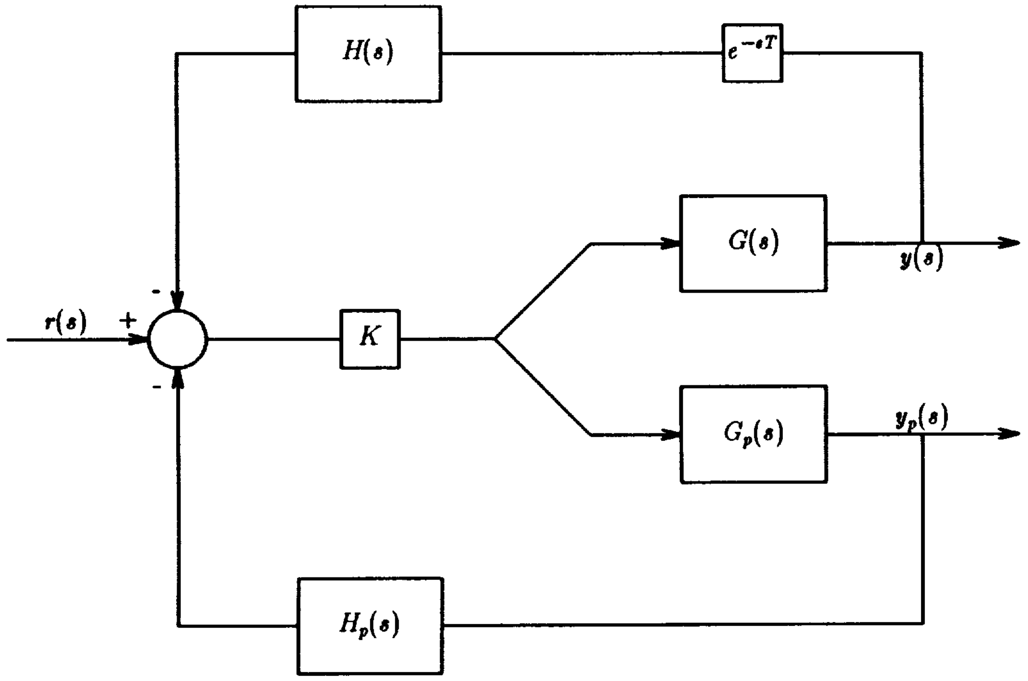


Figure 3. General system diagram for open loop predictor.

If we use only the real feedback with no attenuating filtering (H(s)=1, Hp(s)=0), we observe the standard time delayed feedback system with its well known stability problems. However, using only the predicted feedback (H(s)= 0, Hp(s)=1) we no longer have to worry about the delay induced instability, but we lose accuracy and disturbance robustness.

The predictor display has the effect of giving the operator visual feedback from both the real manipulator and the predictor. The operator then subconsciously combines them to get his "best estimate". Since the normal means of receiving force feedback is strongly coupled to the position control input, operators seem to have more trouble using delayed force feedback than delayed video. Delayed force feedback tends to look like a disturbance. We want to find the best combination of actual and/or predicted force feedback for the operator. The predictor display serves as a model.

In the most directly analogous form we provide both the delayed feedback and the predicted feedback to the operator. The predicted feedback is applied directly to the backdrive input joystick. The delayed feedback is applied to a passive joystick; this separates it more from the loop, letting the operator use it as desired. This presentation form is called "dual" force feedback.

In "complementary" force feedback we explicitly combine the two force feedbacks using complimentary low and high pass filters. The logic for this comes from the predictor display usage: for real time motion tracking the operator uses the predictor's output which seems to causally follow the commands; however, once the transient behavior has settled and the time delay has passed the operator checks the real feedback for accuracy. The low passed real force feedback provides better steady state error, but using the predicted feedback for the higher frequencies lessens the stability problem.

Experiments. Testing of operator performance using force feedback is being done using a single degree-of-freedom simulation with two backdriveable motors: one for input and direct force feedback, the other for passive, indirect force feedback. Tests compare delayed (direct and indirect), predicted, dual and complementary force feedback, and compare them to the baseline of no force feedback. Test subjects perform a series of timed trials involving two tasks: grappling of a floating mass and insertion of the mass into a slot.

References

(1) Ferrell, W.R., Remote Manipulation with Transmission Delay, *IEEE Transactions On Human Factors in Electronics*, No. 1, September 1965.
(2) Ferrell, W.R., Delayed Force Feedback, *Human Factors*, October 1966, pp. 449-455.
(3) Noyes, M. and Sheridan, T.B., A Novel Predictor for Telemanipulation through a Time Delay, *Proceedings of the 1984 Annual Conference on Manual Control*, Moffett Field, CA: NASA Ames Research Center.

3. MODELING AND COMMUNICATING OF INTENTION TO A COMPUTER - Wael Yared

Abstract. A system has been constucted to infer intent from the operator's commands and the teleoperation context, and generalize this information to interpret future commands

Issues in the design of robot command languages. This deals with a central issue confronting designers of human-computer interfaces: the modeling and communication of the human user's intent. This presupposes, of course, that the more immediate issues of defining a syntax and semantics for the interaction have been adequately dealt with. We are stressing here the distinction between the pragmatics of the interaction, on one hand, and its syntactic/semantic aspects on the other. The human user's intent relates sequences of domain actions ("plans") to changes in some model of the task environment. Its representation clearly constitutes, therefore, an issue in pragmatics.

The key words here are "change" and "environment". Interfaces such as robot command languages have only had, so far, atrophied means for the symbolic representation of change due to some agent's actions. Without such tools it is impossible to implement procedures that can reason about the relevance of an act in a plan, or that can benefit from the constraints and opportunities the environment imposes on the agent. These capabilities are, in turn, the basic building blocks for representing and exploiting intentions. A human-computer interaction that lacks these notions is artificial- not in the sense of requiring the human to learn a formal syntax, but in the much more aggravating one of lacking the purposive dimension of any human language.

The issue then is not merely one of user convenience, but of the efficiency and robustness of human-computer communication. An efficient process of task instruction is one in which few or no steps need to be repeated, modified or added when invoking that same task in a different context. A robust instruction is one that permits recovery from error when unexpected contingencies arise in the environment. To illustrate the ideas developed in the course of the present research, we use a simple command language for a 2-D cartesian manipulator as a case study. The scenario for the interaction is that of a human expert typing to the computer a description of a task in the current environment; the human is assumed to beknowledgeable and fully cooperative.

Intention recognition. The purpose of the recognition algorithm is not to check the user's beliefs and intentions against some general blueprint for plans, and then decide whether the user's plan is valid or not; that would amount to no more than a simple terminological convention. Rather, upon enumeration by the user of the actions that comprise the plan, the plan is assumed to be valid, and the system identifies in it the intended acts. Plans are primarily characterized by the intentions they comprise, and this characterization later helps in generalizing plans. The following subsections correspond to the two main layers of the algorithm: the intention recognition algorithm proper, and the plan generalization stage.

In a typical interaction, the user types to the system a sequence of primitive actions that accomplish some task in the simulated environment (e.g. retrieving a block from a warehouse row, switching blocks, etc.). As the instructions are typed, the system parses them and "interprets" them by properly directing pointers to the corresponding executable code. At that stage the task could, in principle, be carried out by the system in the environment without any further user intervention. This is where the intention recognition routine enters the game, however. Instead of giving the user the option of having the task executed, the system proceeds to reason about the plan just defined to elicit the intentions behind it.

What the system does, in effect, is to simulate for itself the effects of that sequence of actions incrementally. To construct this chain of simulated effects, the system has to build a snapshot of its state vector and of the world model after applying each action, and update its private copy of both. Since the chain of actions has already been provided by the user, it is a simple matter to test, at each step in the plan, whether the following action is already executable or not. The system thus identifies the intended acts from the enabling acts in the plan, and stores this information for later use. Upon completion of that stage, the plan is appended to a dynamic plan library. The following points must be emphasized:

(a) The cost of this exercise is negligible: the sequence of instructions is to be entered by the user anyway, and the procedural code for the primitive actions already exists- whether intention recognition is done, or whether straightforward "dumb" plan execution is desired. Any robot command language has to include at least these, and nothing else is required to perform intention recognition. The computational cost of testing the necessary conditions for enablement is proportional to the size of the plan and thus poses no problem. Testing for the sufficient conditions could, in theory, be more problematic; I haven't looked at a worst-case scenario yet.

(b) It is not assumed that each act enumerated by the user is an intended act; rather, they are all intentional acts. The distinction is not futile, since only intended acts are the ones the user is committed to both in terms of reasoning (planning) and in guiding his or her behavior according to the intended act's success or failure.

(c) In contrast to work in planning, no action interrelationships are posited from the start. The system is given a linear, unstructured chain of acts and proceeds to infer nondeductively its corresponding structure.

(d) The intention recognition process is invisible to the user; the intention recognition routine doesn't, in any conventional sense, belong to the user-interface properly speaking, nor to the execution monitor or top-level control of a planner. Although no psychological relevance is claimed in any part of the present work, it is useful to take note of this in trying to dispel the conventional "clean" separation between "smart" planner and "dumb" top-level control.

Plan re-interpretation. A plan for a physical manipulation task is re-interpreted if that task is invoked by the user in some different context than the one in which it was originally defined. When this case arises, the plan is reevaluated and eventually modified; if modified, it is appended again to the plan library. The plan re-intepretation procedure comprises the following steps:

(a) The system locates in the plan library the most recent precedent of the plan definition, together with the latter's corresponding model of the world environment at the time of the original definition. If the plan definition includes parameters, the system performs the trivial task of substituting the current parameters for the old ones in the original plan definition. At this point the plan is re-compiled, and the system proceeds to the relevance check.

(b) In a first phase of the relevance check, the system performs a backward pass on all the subgoals of the modified plan in order to determine which of them are still relevant. A subgoal is no longer relevant if the current state of the world and the robot are identical to the ones it is there to achieve. The state of the world and of the robot a subgoal achieves is calculated by aggregating the effects of all the previous subgoals with those of the intended act of the current subgoal. Since it appears reasonable to assume that when a subgoal is irrelevant, then all its preceding siblings become irrelevant, the backward pass is stopped at the first occurrence of an irrelevant subgoal. Pointers are then redirected to ensure that the new plan only comprises the relevant subgoals.

(c) In a second phase of the relevance check, the system checks the relevance of the enabling acts for each (now relevant) subgoal. Within each subgoal, a backward pass evaluates the relevance of the enabling acts with the same mechanism that was used in the intention recognition routine. In other words, the executability of each enabling act is checked in turn (in the context of the current subgoal); when an enabling act is found executable, its preceding siblings are all no longer relevant and are discarded from the plan.

Implementation. The ideas presented in the previous sections have been implemented in an experimental robot interface called GRICE (Generalization through Recognition of Intention and Chains of Enablement). GRICE simulates a 2-D cartesian manipulator in a task environment that includes a warehouse and several blocks. The user configures the environment (places the blocks and the robot) with the mouse, or retrieves a pre-configured one from a data file. A menu then prompts the user to define a new task, retrieve a previously defined task, generalize on a previously defined task, or execute a current task.

GRICE is implemented in the C language on a Silicon Graphics IRIS 2400 system. The reasoning routines, the task execution routines, and all the data structures (about 50% of the 3000 lines of code) are totally machine-independent and domain-indepedent. Some of the domain-dependent routines (the repertoire of robot primitives, but not the command interpreter) and the graphic simulation draw heavily on the IRIS graphics library and graphics-dedicated hardware.

Data structures for plans (both source and binary versions), world model and robot state vector are all implemented as standard doubly-linked lists, which eases their traversal in both directions. An object-oriented programming style allows the reasoning and task-execution routines to be independent of domain of application; in C, this is implemented using pointers to functions.

4. COMMAND LANGUAGE FOR OBSTACLE AVOIDANCE - Jong Park

Abstract. The purpose of this research is to design a command language system that is robust, easy to learn, and has more natural man-machine communication. A general telerobot problem selected as an important command language context is finding a collision-free path for a robot.

Background and needs. A complete task-level robot command language is long overdue, where goals for the position of objects are to be specified, rather than the motions of the robot needed to achieve these goals. Available techniques, such as the configuration-space approach of Lozano-Perez, are promising, but finding a solution takes too long to be used on-line.

In order to overcome this difficulty, and in keeping with the notions of "supervisory control", it is suggested that a human operator should play a more active role by making global decisions, letting the computer deal with the details. In the context of collision avoidance the human operator should decide how to maneuver to avoid collision, then let the robot find if his commands will work. Such a command language should be reliable, with no misinterpretation of commands, should be easy and natural for the operator to use, and should be easy to learn.

Command language. In the command language being designed, the operator types natural-language-like commands to a keyboard (a symbolic input) and simultaneously may use a joystick to indicate geometric direction or magnitude (an analogic input). He may also move a cursor to point (to an object or attributes of an object such as surfaces, edges or vertices on a video or computer-graphic screen--easier than calling their names) in coordination with a symbolic action command. Interpretation by the computer in this case will depend upon the viewpoint of the operator.

Both syntactics of the command language (e.g., the mathematical equations or logical rules constraining what is admissible) and the semantics (specification of motions and locations of objects and their attributes relative to the environment) are important. In Figure 4, for example, it may be more natural to say "A approach B until x away " than "A go y along normal to B", since the constraint "x" is likely to be more relevant to what matters to the operator than the actual distance travelled. Specifying "Go 10 inches parallel to and 1 inch away from (this) edge in (this) direction" (edge and direction indicated by translating and orienting cursor) is likely to be easier than saying "Go (87,35,67) to (93, 64, 24)" after determining the precise starting and ending coordinates. Usually the operator needs to know and specify some variables with reasonable precision, though other variables can assume a wide range. If the computer has other context information it can even make sense of a command like "Go 3 above (this) surface".

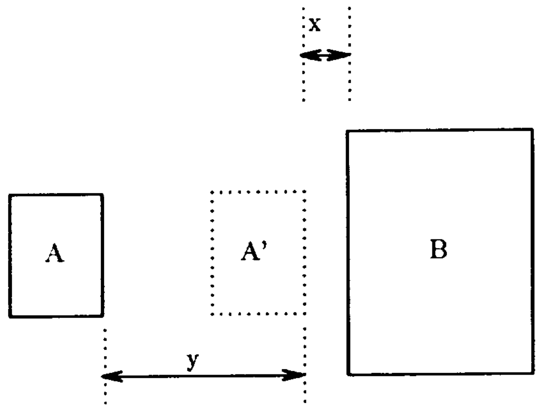


Figure 4. A approaches B.

Hardware and software. The system hardware consists of three input devices (keyboard, mouse, 6 degree-of-freedom joystick) and graphics workstation. The software consists of six modules as shown in Figure 5: natural language interface (NLI), indication interpreter (II), general interpreter (GI), guidance handler (GH), graphic simulator (GS), and executor (EX). NLI is based upon the already developed "augmented transition network" capable of interpreting limited vocabulary and grammer. Typed natural-language-like commands are interpreted by NLI. Voice commands can be added later. II works with NLI to interpret the meaning of the operator's mouse-cursor commands. GI takes input from NLI and II and determines robot motions accordingly. GH handles inputs from the 6 DOF joystick and also moves the robot. GS checks if commanded motions are free of collision, and if so EX is called to execute the motions.

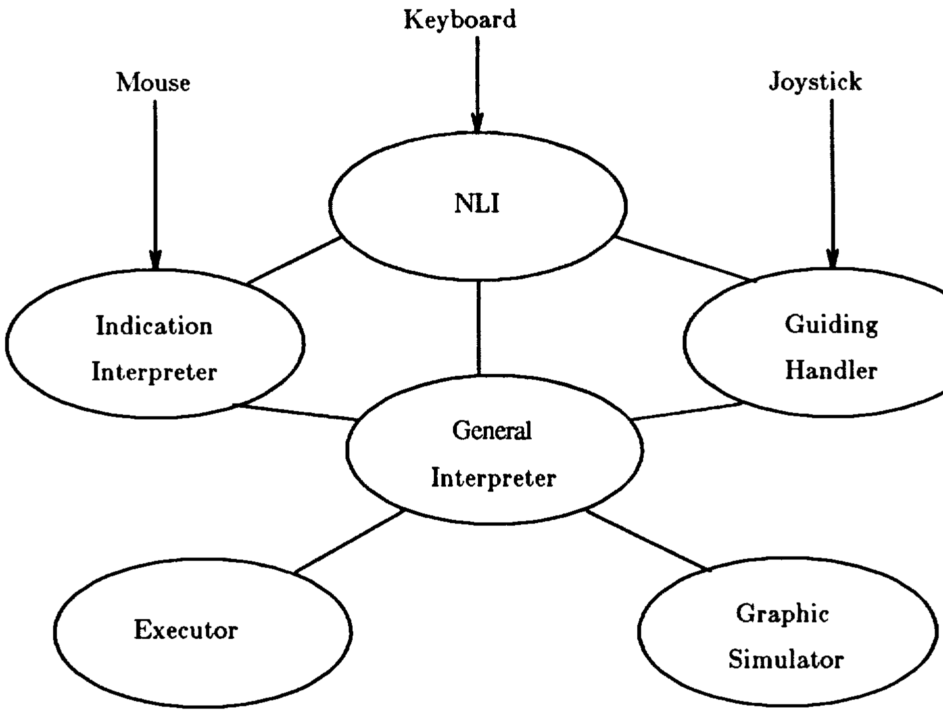


Figure 5. System configuration.

Experiments. Experimental subjects are being asked to perform a telerobotic obstacle avoidance task to evaluate the effectiveness of the command language.

ACKNOWLEDGMENT. Projects 1, 2 and 4 were sponsored by Jet Propulsion Laboratory, Project 3 by NASA Ames Research Center. Their support is gratefully acknowledged.

N90-29008

VARIABLE FORCE AND VISUAL FEEDBACK EFFECTS ON TELEOPERATOR MAN/MACHINE PERFORMANCE

Michael J. Massimino and Thomas B. Sheridan

Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

An experimental study was conducted to determine the effects of various forms of visual and force feedback on human performance for several telemanipulation tasks. Experiments were conducted with varying frame rates and subtended visual angles, with and without force feedback.

1. Introduction

In Section two of this paper we describe our research objectives for conducting our experiments that focused on human/machine interaction for space teleoperation.

Section three contains descriptions of the experimental equipment. Our experimental design is discussed in Section four. Six test subjects used a master/slave manipulator during two experimental sessions. In one session the subjects performed the tasks with direct vision, with and without force feedback, and with the manipulator at three different distances from the task board yielding three subtended visual angles: 3.28, 1.64, and 1.09 degrees respectively. During the other session, the tasks were performed using a video monitor for visual feedback, with and without force feedback, and with three different frame rates (3, 5, and 30 frames/second) for the video transmission. The tasks were three peg-in-hole type tasks corresponding to 4, 5, and 6 bits/task according to Fitts' Index of Difficulty.

Experimental results for the video viewing and direct viewing environments are presented in Section five. The experimental data were analyzed through an analysis of variance. The video viewing results showed that frame rate, force feedback, task difficulty, and the interaction of frame rate and force feedback made significant differences in task times. For the direct viewing environment, subtended visual angle, force feedback, task difficulty, and the interaction of subtended visual angle and force feedback made significant differences in task times. Also in Section five are the results of comparing performance between the video and direct viewing environments. Comparable visual feedback to the human operator was provided by (a) the 1.64 subtended visual angle for direct viewing, and (b) thirty frames/second frame rate for video viewing. This allowed for an analysis between the direct and video viewing environments. While force feedback and task difficulty made significant differences in task times, the view itself (video vs. direct) did not.

Conclusions and suggestions are made in Section six based on the research results to help facilitate improved teleoperator performance for space operations.

Additional information on this experimental study can be found in [1].

2. Research Objectives

Performing a task directly with hands and eyes, unimpeded by physical distance, hardware, or artificial communication, has been observed experimentally to be quicker, easier, and usually more accurate than performing a task remotely using video and a remote manipulator device [2]. However, not all space tasks are well suited to be done manually by astronauts in extra-vehicular activity (EVA) due to the nature of the task, the risks and nature of the space environment, or the astronaut being constrained by his/her gloves or pressure suit. Thus there are safety, cost, and efficiency considerations that can make EVA procedures undesirable. Many researchers and engineers, faced with the necessity of remote viewing and handling, have sought to understand how human/machine interaction for the control of space teleoperators can be improved [3-5]. Nevertheless, knowing when to have manipulation tasks done by astronauts in EVA, remote manipulators controlled by humans, or autonomous robots is an important current issue.

The human operator's performance in telemanipulation can surely be improved by providing the operator with appropriate and adequate feedback. Teleoperation will not eliminate or drastically alter the need for humans in space, but it will alter the roles that humans fulfill in space and should improve human productivity there. The use of remote manipulators should also free up valuable crew time to be spent on other space operations and experiments.

The elements of a remote viewing and manipulation system include a video camera, a telecommunication channel, a video display, human eyes, human arms and hands, a master arm, a slave or robot arm, and the task operation itself (such as putting a peg into a hole). These elements have not only desirable but also undesirable properties. They tend to add undesirable forces, displacements, time, illumination, and contrast to the remote environment that would not be there in the manual situation. They can be perceived as "filters" that prevent information from reaching the human operator, and thus retard performance [2]. In the experiments presented in this paper the effects of various degrees of some of these "filters" were investigated including:

* Force feedback and its effects on motor capabilities.
* Video frame rate, and its effects at different values on performance.
* Subtended visual angle and its effects on task performance for manipulation with direct viewing.
* Task difficulty and its effects on the cognitive and motor capabilities of human operators.
* The use of a video monitor versus the use of direct vision for remote manipulation.
* The interactions of two or more of the above variables with each other and the corresponding effects on performance.

These "filters" and their effects need to be identified and quantified to provide information to facilitate efficient teleoperation in space. Experimental findings, quantified and analyzed through statistical methods, should prove helpful to researchers and policy makers alike. The results could be implemented over a variety of space applications including:

* space shuttle remote manipulator system [6]
* control of an orbital maneuvering vehicle with a robotic front end [7]
* telerobotic servicer on the space station
* a number of planetary and lunar missions such as a Mars sample return mission or lunar exploratory operations [8-9].

The major goal of the experiments presented here was to provide information to help technologists better understand what the capabilities of humans are when interfacing with space telemanipulators under various sensory feedback conditions.

3. Experimental Equipment

E2 Manipulator System

The E2 master-slave manipulator had the capability of operating with direct electronic coupling control both bilaterally with force feedback, and unilaterally without force feedback coming from the slave back to the master arm. Our E2 was right handed, as were all of the test subjects used in the experiments. Both master and slave arms had seven degrees of freedom including end effector gripping, were geometrically similar, and were kinematically isomorphic to the operator's arm and hand. The manipulator degrees of freedom are shown in figure 1. The E2 had seven degrees of freedom: three (x, y, and z direction) for arm translation, one for arm rotation (azimuth), one for gripper elevation, one for gripper twist, and one for the grasping motion of the gripper jaw. Tracking time delays were considered negligible during the experiments due to the manipulator's quick and accurate response to the operator's input motions [10].

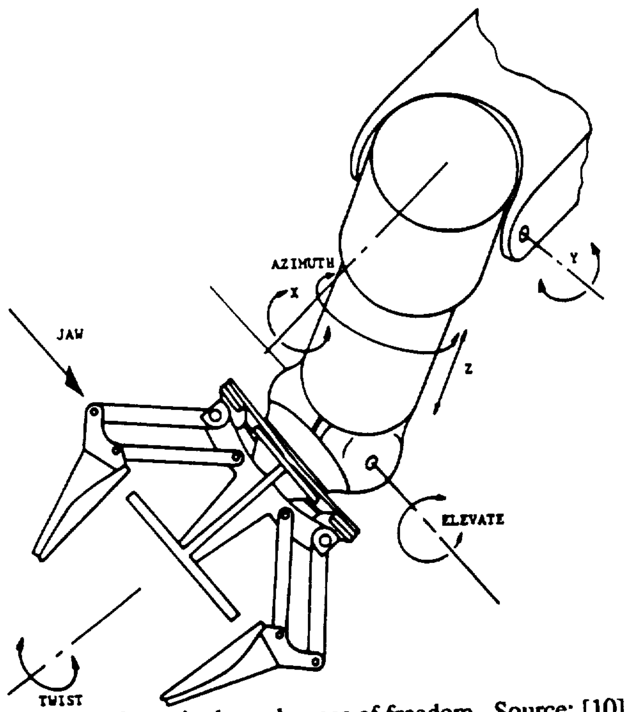


Figure 1 - E2 manipulator degrees of freedom. Source: [10]

Video System

Central to altering the video environment was the capability of varying the frame rate. This was accomplished through the use of the AT & T Truevision Advanced Raster Graphics Adapter (TARGA 16) board and a computer program. The resolution selected for the experiments was 512 X 256 pixels in order to have non-interlaced video input or output. TARGA captures images in real time:1/60th of a second per field or 1/30th per frame. Once a frame rate was selected, the TARGA board would capture a frame at the rate necessary to provide the requested frame rate. A color monitor provided the visual feedback to the test subjects.

Task Board

The task board consisted of four slots. Each slot was made with two side boards and a center board that formed a back. Figure 2 displays the task board dimensions. The two end slots were 3.75 inches wide and the middle slots were 3.25 and 3 inches wide respectively. The block used in the experiments was 2.75 inches wide, providing task tolerances from left to right of one inch, one-half inch, one-quarter inch, and one-inch. The redundancy of the one inch tolerance on the outer slots allowed the alternating of right and left motion. The centers of each of the four slots were eight inches apart. Mounted on the lower portion of each center block were limit switches that controlled the clock to record the task times.

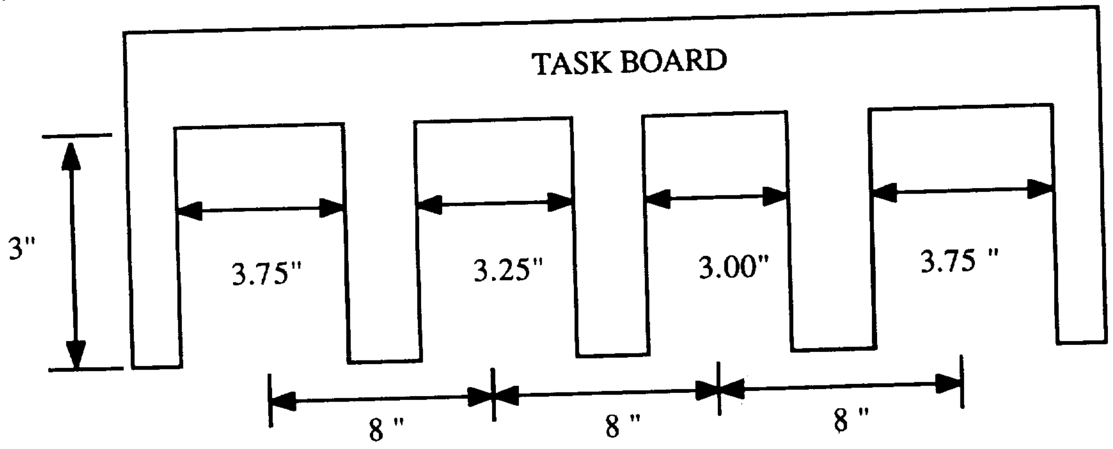


Figure 2 - Task Board Layout

4. Experimental Design

Tasks

The tasks consisted of moving the block of wood on the task board with the manipulator arm. Thus the distance moved (eight inches) versus the tolerance of fit (1,.5,.25 inches) provided multiples of 2 for easy use of Fitts' law [11] which applies the formula: Index of Difficulty (Id) in bits per task equaled $\log_2(2A/B)$, where A equaled the distance moved and B equaled the tolerance of fit. Fitts' law produced indices of difficulty of 4, 5, and 6 bits per task respectively.

Video Viewing Experiments

Since communication channels for teleoperation in space are often constrained by limited bandwidth for information transmission, decreasing the amount of information that needs to be transmitted can save time and money. Decreasing frame rate is one way to decrease the amount of information that is transmitted, and this section of the experiment was designed to measure the effects of varying frame rate on operator performance. The goal was to gain information on acceptable frame rates for controlling a remote teleoperator under different force feedback and task difficulty conditions.

The direct view of the task board was cut-off from the subject and the video monitor was used with three different frame rates: three frames per second (fps), five fps, and thirty fps, with and without force feedback, for a total of six experimental conditions. Thirty frames per second provided satisfactory image fusion, i.e. output that appeared as steady motion. Five fps and three fps were selected based on previous experiments and preliminary experimentation. Ranadive [12] found that a threshold frame rate existed at three fps beyond which task performance was virtually impossible. He also discovered that frame rates below 5.6 fps considerably degraded performance and increased variability. Preliminary experimentation confirmed these trends. After trying many frame rates, 30, 5, and 3 fps were chosen since they appeared to represent breaks in the performance curve.

Direct Viewing Experiments

The tasks were also performed with the subjects using direct vision for visual feedback, with and without force feedback. The master manipulator was placed at three different viewing distances from the task board: four, eight, and twelve feet. Each distance from the task board in the direct viewing experiments had an associated subtended visual angle. The formula: subtended visual angle = (57.3) (60) L/D degrees, gives a measure for subtended visual angle where L = the size of the object measured perpendicular to the line of sight, and D = the distance from the front of the eye to the object [13]. In this case L = 2.75 inches and D varied between 4 feet (48 inches), 8 feet (96 inches), and 12 feet (144 inches), yielding subtended visual angles of 3.28 degrees, 1.64 degrees, and 1.09 degrees respectively. This allowed analysis of the effects of varying subtended visual angles on task times.

Direct Viewing at 8 feet and Video Viewing at 30 fps

The eight foot distance yielded the same subtended visual angle as the experiments that were performed with the video monitor (1.64 degrees). This allowed for a comparison of the 1.64 degrees direct viewing data with the 30 fps video data since 30 fps appeared to the eye as a steady motion. Thus 1.64 degrees direct data and 30 fps video data each had subtended visual angles and frame rates that appeared equal to the human operator. Some of the major differences were stereo vision with direct viewing, the ability to move one's head to change the viewing line of sight with direct viewing, and the different environments surrounding the task board for each view. It was the effects on performance of the stereo vision and other differences in the views that were of interest in this section of the experimental design.

Experimental Procedures

Six MIT graduate students were used as test subjects. Each subject attended a one and a half to two hour training session a few days prior to performing the final experimental runs. The subjects were made familiar with the experimental design and procedures and became acquainted with the E2 manipulator system. They performed each task condition and repeated the tasks until they said they were familiar and felt comfortable with the manipulator and the manipulation environment. Then they performed the tasks for time just as they would in the actual experiments. When their performance times met minimum training levels and the learning effects subsided, the subjects were then trained on the next experimental condition.

Subjects then underwent two separate experimental sessions: one using the video monitor and the other for performing the tasks with direct vision. A balanced latin square was used so that each experimental condition preceded and followed every other condition an equal number of times to counterbalance the effects of fatigue and learning on the experimental results.

The video portion consisted of three frame rates, two forms of force feedback, and three different tolerances which corresponded to three different indices of difficulty for a 3x2x3 design. The direct vision experiments included three visual angles, and the previously stated feedback and tolerance/difficulty parameters for a 3x2x3 design. Each task was performed five times going to the right and five times going to the left for 5x2 = 10 tasks per condition. Six subjects were used so the video experiments had a total of 3x2x3x5x2x6 = 1080 number of data points, and the direct vision experiments also yielded a total of 3x2x3x5x2x6 = 1080 number of data points.

5. Experimental Results

The experimental data was analyzed through an analysis of variance (ANOVA) with a 95 percent confidence level as described in [14-15], and Newman-Keuls post-hoc testing as outlined in [16]. The results of the statistical analysis are illustrated in tables 1 and 2 and figures 3 to 8.

Table 1 - ANOVA results for variables and interactions with statistical significance during video viewing performance

SOURCE	Degrees of Freedom	F-VALUE	P-VALUE
Frame rate	2, 10	73.1	0.0001
Force feedback	1, 5	167	0.0001
Task difficulty	2, 10	15.3	0.0009
Interaction of frame rate & force feedback	2, 10	8.99	0.0058

Table 2 - ANOVA results for variables and interactions with statistical significance during direct viewing performance

SOURCE	Degrees of Freedom	F-VALUE	P-VALUE
Subtended visual angle	2, 10	57.8	0.0001
Force feedback	1, 5	49.5	0.0009
Task difficulty	2, 10	27.3	0.0001
Interaction of subt. vis. ang. & force feedbk	2, 10	6.91	0.013

Effects of Force Feedback

As displayed in tables 1 and 2, force feedback made a significant difference in performance. For video performance, force feedback was significantly better than no force feedback. With force feedback the average task time for all frame rates combined was 2.98 seconds and the absence of force feedback produced a combined average task time of 5.29 seconds.

Even when viewing was direct (no video), force feedback made a significant improvement in the mean task times. The presence of force feedback yielded a total mean task time of 1.71 seconds and without force feedback the total mean task time was 2.80 seconds.

Effects of Task Difficulty

For the video viewing experiments, task difficulty made a significant difference as shown in table 1. The Newman-Keuls analysis determined that only the quarter inch tolerance or the most difficult task (Id=6 bits) had significantly different mean task times from the other two tasks. The one inch tolerance task with Id=4 and the half inch tolerance task with Id=5 were not significantly different from each other. For Id=4 the mean task time was 3.55 seconds, for Id=5 the mean task time was 3.95 seconds, and for Id=6 mean task time was 4.90 seconds.

These relationships are more clearly represented in figure 3. The mean task time values are plotted for the three different indices of difficulty. The error bars represent the standard error for each mean. A linear regression yielded y=0.758+ 0.675x. An exponential fit was also done, yielding a relationship of time (y) to index of difficulty (x) of y=1.83*10^(0.07x). The pattern of the data points and the results of the Newman-Keuls post-hoc test suggest that the exponential plot described the behavior or trends of the data better than the linear plot. These were different from the linear results that Fitts [11] obtained but were similar to the results of Hill [17-18].

Although these results are based on only three data points, each data point is the mean of three hundred and sixty data observations. Further, the plots are only meant to indicate general trends in performance. Therefore it is with some confidence that these results are presented.

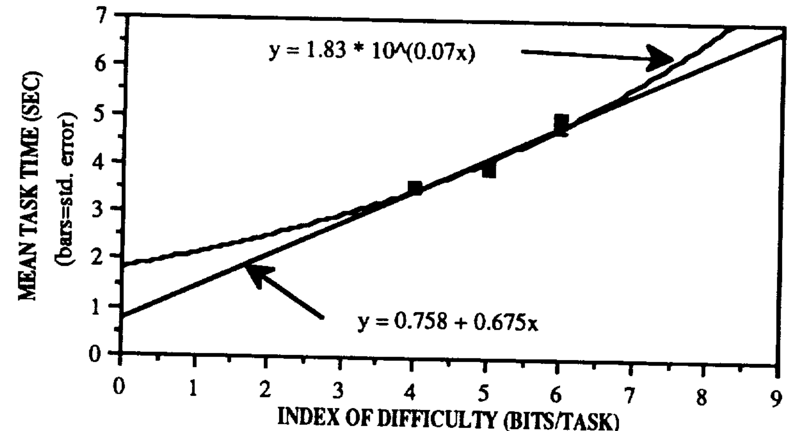


Figure 3 - Task Difficulty Effects on Video Viewing Performance

Table 2 displays that task difficulty also made a significant difference in performance for the direct viewing experiments. Post-hoc testing revealed that the easier tasks with Id's equal to 4 and 5 did not produce significantly different means. However, the most difficult task (Id=6) was found to yield task means that were significantly different from the other two. The mean task time for Id=4 was 1.89 seconds, for Id=5 it was 2.15 seconds, and for Id=6 it was 2.74 seconds. Figure 4 graphically displays the effects of task difficulty on direct viewing performance. Both a linear fit, y=0.138+0.424x, and an exponential fit, y=0.882*10^(0.081x), were performed. As was found with the video viewing results, the increases in mean task time displayed more of an exponential tendency than a linear one.

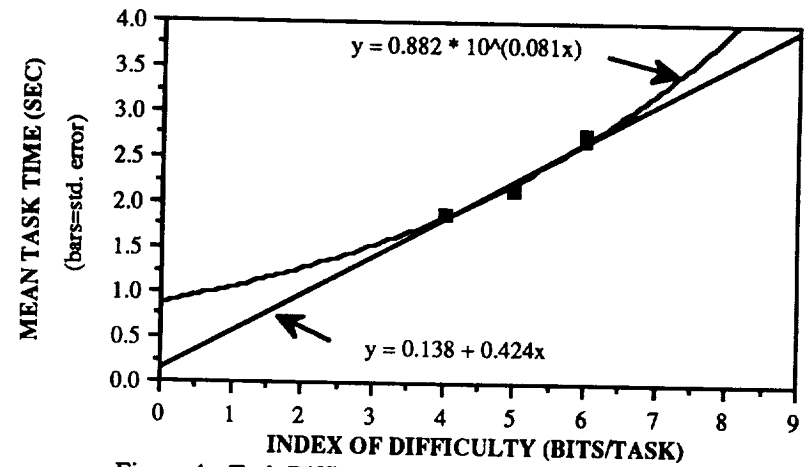


Figure 4 - Task Difficulty Effects on Direct Viewing Performance

Effects of Frame Rate

Frame rate made a significant difference in performance (table 1). The Newman-Keuls post-hoc test showed that all three frame rates were significantly different. Mean task time with a frame rate of three frames per second (fps) was 5.36 seconds, for five fps it was 4.48 seconds, and for 30 fps it was 2.56 seconds. These results are shown graphically in figure 5.

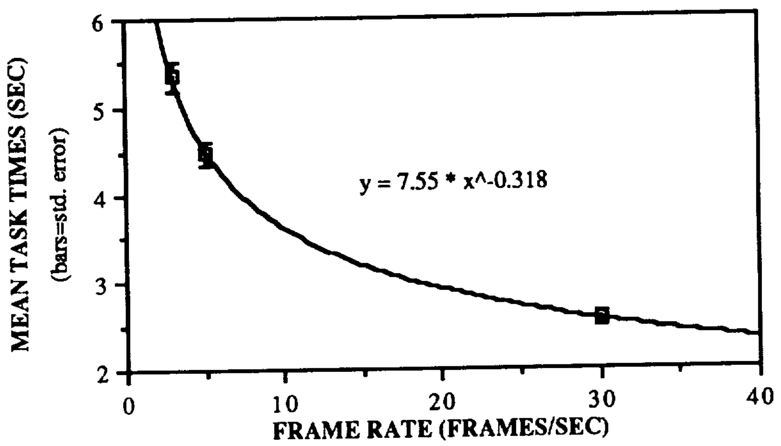


Figure 5 - Frame Rate Effects on Video Viewing Performance

Interaction Effects of Frame Rate with Force Feedback

The interaction of frame rate and force feedback was also noticed to make a significant difference in performance times (table 1). Post-hoc testing results showed that three frames per second (fps) with force feedback, five fps with force feedback, and thirty fps without force feedback were found to produce mean task times that were not significantly different from each other. Three fps without force feedback and five fps without force feedback were not significantly different from each other. Thirty fps with force feedback was significantly different from all other conditions. These results are graphed in figure 6.

While at each frame rate, force feedback made a significant improvement in performance times, force feedback yielded a larger performance improvement at lower frame rates than at higher frame rates. Even at 3 fps, force feedback provided such a large improvement in performance that the mean task time was not significantly different from 30 fps without force feedback.

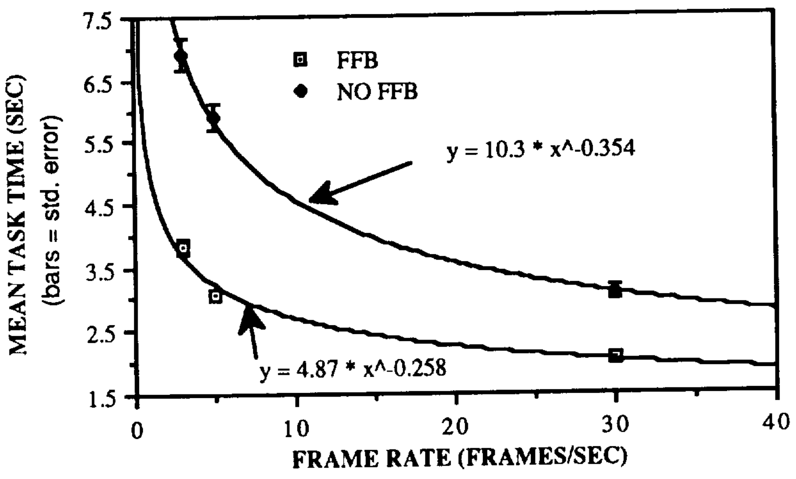


Figure 6 - Interaction of Frame Rate and Force Feedback

Effects of Subtended Visual Angle

Subtended visual angle did cause significant differences in mean task times (table 2). The Newman-Keuls post-hoc analysis determined that the three subtended visual angles produced task time means that were significantly different from each other. The mean task time for a subtended visual angle of 3.28 degrees was 1.92 seconds, for 1.64 degrees the mean task time was 2.27 seconds, for 1.09 degrees the mean task time was 2.59 seconds. These results are displayed in figure 7.

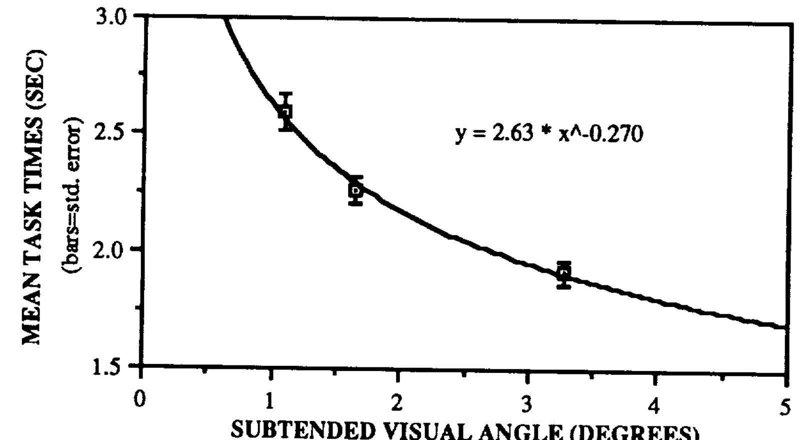


Figure 7 - Subtended Visual Angle Effects on Direct Viewing Performance

Interaction of Subtended Visual Angle with Force Feedback

The interaction of subtended visual angle with force feedback was found to make a significant difference in performance times (table 2). The Newman-Keuls post-hoc tests revealed that at all three subtended visual angles, force feedback versus no force feedback made a significant difference.

While operating with force feedback, decreasing the subtended visual angle did not significantly increase task times. Further, force feedback was able to improve performance at the smallest subtended visual angle by a margin large enough to make the task times not significantly different from those observed for the largest subtended visual angle without force feedback. Performing tasks without force feedback, and at a 3.28 degree subtended visual angle was not significantly different from that at 1.64 degrees without force feedback. However, performance at 1.09 degrees without force feedback was significantly different from that for the other two subtended visual angles without force feedback. These results are displayed in figure 8.

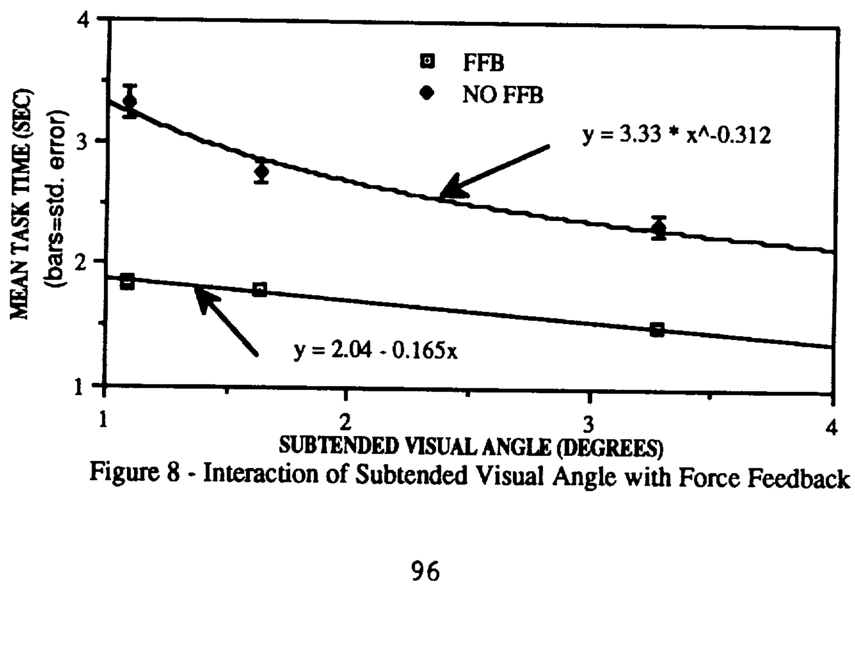


Figure 8 - Interaction of Subtended Visual Angle with Force Feedback

Effects of View (Video Viewing at 30 fps vs. Direct Viewing at 1.64 degrees)

When frame rate and subtended visual angle were similar, video and direct viewing mean task times were not found to be significantly different. This indicated that for these experiments whether the view was direct or video by itself did not make a difference and that the primary visual variables affecting performance were subtended visual angle and frame rate.

6. Conclusions and Recommendations

When using a remote manipulator system without force feedback, like the space shuttle remote manipulator system (RMS), and with direct vision, a relatively large subtended visual angle should be provided to the operator if possible. The experimental results suggested that in direct viewing telemanupulation, an adequate subtended visual angle could compensate for performance degradations that were due to the operator being at large distances from the task board. A larger subtended visual angle could be yielded by providing a larger target to increase the size of the image on the operator's retina thus increasing the subtended visual angle. Additionally, since force feedback would not be present, one would also expect that performance will be degraded more at smaller angles than it would be if force feedback were present.

The experiments indicated that frame rate and subtended visual angle were the visual feedback variables that affected performance significantly in the experiments, not whether direct or video viewing was implemented. Some previous studies (performed without a transmission time delay) concluded that position and force feedback are reconstructible to a large degree in teleoperation, but recreating the visual image as feedback to the operator was more challenging due to the loss of information (such as stereoscopic vision) when viewing a television monitor [19]. Our results suggested that there was not as a significant decrease in performance due to the loss of information when going from direct viewing to video viewing as might be expected. This leads one to conclude that if a manipulator without force feedback were being used (such as the RMS), and direct vision yielded a small subtended visual angle, it would be wise to use video transmission to provide a larger subtended visual angle. Although the video monitor may not provide stereoscopic vision, performance would probably be improved with the larger subtended visual angle. Thus if a choice is given between direct viewing with a small subtended visual angle against video viewing with a high frame rate and larger subtended visual angle, video viewing could be the wiser choice.

However, there may be other explanations for view not having a significant effect on mean task time. For example, the effects of stereo vision could be greater at shorter distances than at longer distances. Therefore it is possible that the eight foot distance was too great to utilize the full advantages of stereo vision. This is a topic for future research. Nevertheless the results found in these experiments suggest that the view itself did not have a significant effect on mean task times.

If teleoperation were to be controlled from a ground control station or a space station workstation with a video monitor (such as with the control of an orbital maneuvering vehicle or flight telerobotic servicer), operating at very low frame rates below 3 fps should be avoided unless large performance degradations are acceptable. The reduction in mean task times was found to occur at a faster rate when going from 5 fps to 3 fps than when going from 30 fps to 5 fps. This suggests that frame rates between 5 fps and 30 fps may produce performance results that would more likely meet acceptable performance criteria than frame rates below 5 fps. Thus if there is limited bandwidth available for frame rate and depending on the task, reducing frame rate from 30 fps may be acceptable until a cutoff frame rate is reached beyond which performance would be below the accepted level. It may be possible to reduce frame rate to a larger degree without harming performance by a great margin if force feedback is present. If force feedback is not present, it would probably be important to have the video transmission at a high frame rate.

Force feedback was found to make up for many of the performance degradations due to decreased feedback in the visual feedback channel. Force feedback significantly improved performance at all frame rates, and was particularly helpful at the lower frame rates. In direct viewing performance, force feedback was found to have a stabilizing effect. When subtended visual angle was decreased with force feedback, there was not a significant increase in task times. Additionally, force feedback yielded a larger improvement in performance time over the no force feedback case as subtended visual angle was decreased. Therefore whenever visual feedback conditions are extremely poor and cannot be improved, the use of force feedback could very well improve performance times to acceptable levels.

The results also suggest that force sensing was probably more important than vision for the insertion of the block into slot. Some scenarios may dictate that force feedback is impossible or undesirable, such as when used with a transmission time delay. But if force feedback is available and the task and environment do not prohibit its use, force feedback should be utilized.

As tasks become increasingly difficult, designers should not assume linear increases in task times. Task times can increase at increasing rates. This also suggests that beyond certain difficulty measures, performance time can increase beyond acceptable ranges.

For any manipulation task the effects of the different feedback variables will be unique, making broad generalizations ill advised. However the conclusions presented here indicate that task difficulty, force feedback, and the visual feedback parameters of frame rate and distance (or subtended visual angle) all can have significant effects on human performance for telemanipulation.

References

[1] Michael J. Massimino. "Effects of Force and Visual Feedback on Space Teleoperation; with Policy Implications." MIT Masters Thesis, Department of Mechanical Engineering, Man-Machine Systems Laboratory, May, 1988.

[2] Thomas B. Sheridan, "Why Bare Hands Beat Telemanipulators: An Approach to Understanding Why," MIT Man/Machine Systems Laboratory, December, 1986.

[3] Antal K. Bejczy and Kevin Corker. "Automation in Teleoperation from a Man-Machine Interface Viewpoint." AIAA/NASA Space Systems Conference, Costa Mesa, CA. June 5-7, 1984.

[4] Antal K. Bejczy. "Distribution of Man-Machine Controls in Space Teleoperation." SAE Technical Paper Series, 821496. Aerospace Congress & Exposition, Anaheim,CA,October 25-28, 1982, pp.15-27.

[5] Antal K. Bejczy. "Human Factors Issues in Space Teleoperation." 2nd Seminar on Human Factors Technology for Next Generation Transportation Vehicles, I.C.T.S., Amalfi, Italy. June 16-20, 1986.

[6] Aaron Cohen. "Automation and Robotics: Key to Productivity" 36th Congress of the International Astronautical Federation. Stockholm, Sweden, October 7-12, 1985.

[7] Kumar Krishen. "Vision Technology/Algorithms for Space Robotics Applications." SOAR '87 - First Annual Workshop on Space Operations Automation and Robotics. Houston, Texas, August 5-7, 1987.

[8] Stanley Deutsch. "Remotely Manned Systems - Augmenting Man's Capabilities in Space Operations." In Remotely Manned Systems: Exploration and Operation in Space. Ed. Ewald Heer. Pasadena: California Institute of Technology, 1973, pp. 3-5.

[9] Heer, Ewald. "Remotely Manned Systems for Operation and Exploration in Space." In Symposium on Theory and Practice of Robots and Manipulators. 1st, Udine, Italy, September 5-8, 1973, Udine, Italy: International Centre for Mechanical Sciences, 1973, pp. 150-167.

[10] James Hampton Black. "Factorial Study of Remote Manipulation with Transmission Time Delay." MIT Masters Thesis, December, 1970.

[11] Paul M. Fitts. "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement." Journal of Experimental Psychology, Vol. 47, No. 6, June, 1954, pp. 381-391.

[12] Vivek Ranadive. "Video Resolution, Framerate, and Grayscale Tradeoffs under Limited Bandwidth for Undersea Teleoperation." MIT Masters Thesis, Man-Machine Systems Laboratory, Department of Mechanical Engineering, September, 1979.

[13] Robert W. Bailey. Human Performance Engineering: A Guide For System Designers. Englewood Cliffs: Prentice-Hall, Inc., 1982.

[14] Alphonse Chapanis. Research Techniques in Human Engineering. Baltimore: The Johns Hopkins Press, 1959.

[15] William Mendenhall, Richard L. Scheaffer, and Dennis D. Wackerly. Mathematical Statistics with Applications. Boston: Duxbury Press, 1981.

[16] Francis J. Wall. Statistical Data Analysis Handbook. New York: McGraw-Hill, 1986.

[17] J. W. Hill. "Two Measures of Performance in a Peg-in-Hole Manipulation Task with Force Feedback." MIT Proceedings, 13th Annual Conference on Manual Control. June 15-17, 1977, pp. 301-309.

[18] J.W. Hill. "Study of Modeling and Evaluation of Remote Manipulation Tasks with Force Feedback." Final Report. JPL Contract 95-5170, March, 1979.

[19] Jean Vertut and Philippe Coiffet. Robot Technology, Volume 3A, Teleoperation and Robotics: Evolution and Development. Englewood Cliffs: Prentice-Hall, 1984.

N90-29009

Teleoperator Comfort and Psychometric Stability: Criteria for Limiting Master-Controller Forces of Operation and Feedback During Telemanipulation

Steven F. Wiker
Department of Industrial Engineering &
Wisconsin Center for Space Automation and Robotics

Elaine Hershkowitz and John Zik
Wisconsin Center for Space Automation and Robotics

University of Wisconsin-Madison

Abstract

In this paper we address the following question: How much force should we require operators to exert, or experience, when operating a telemanipulator master-controller for sustained periods without encountering significant fatigue and discomfort, and without loss of stability in psychometric perception of force. The need to minimize exertion demands to avoid fatigue is diametrically opposed by the need to present a wide a range of force stimuli to enhance perception of applied or reflected forces. For 104 minutes subjects repetitiously performed a series of 15 s isometric pinch grasps; controlled at 5, 15, and 25 percent of their maximum voluntary strength. Cyclic pinch grasps were separated by rest intervals of 7.5 and 15 s. Upon completion of every 10 minute period, subjects interrupted grasping activities to gage the intensity of fatigue and discomfort in the hand and forearm using a cross-modal matching technique. A series of psychometric tests were then conducted to determine accuracy and stability in the subject's perception of force experienced. Results showed that onset of sensations of discomfort and fatigue were dependent upon the magnitude of grasp force, work:rest ratio, and progression of task. Declines in force magnitude estimation slopes, indicating a reduction in force perception sensitivity, occurred with increased grasp force when work:rest ratios were greater than 1.0. Specific recommendations for avoiding discomfort and shifts in force perception, by limiting pinch grasp force required for master-controller operation and range of force reflection or work:rest ratios, are provided.

1. Introduction

Numerous questions have arisen in the course of design and construction of telemanipulator master-controllers and end-effectors; particularly, concerning the nature and magnitude of manipulative stimuli which should be presented to the controlling hand. Recent studies have found that limiting muscle exertions to less than 20 percent of maximum voluntary contraction (MVC) levels is not an adequate design guideline for avoidance of localized muscle fatigue and discomfort for all muscle groups [1]. Moreover, population estimates of manual strength capability are very limited. These problems, when combined with insidious development of over-estimations of forces encountered during states of localized muscle fatigue [2], argue for minimization of forces experienced in either controller operation or reflected by the end-effector. However, diametrically opposing such a strategy is the fact that, depending upon the level of force required to simply operate the master-controller, severely restricting the dynamic range of force reflection could result in only very limited, and potentially useless, force information.

This design paradox led us to conduct a study to address the following questions. First, what are acceptable levels of force of operation, or force reflection, in terms of operator tolerance and stability of force perception? Second, do changes in force perception occur and, if so, do they precede alerting signs of discomfort and fatigue? Finally, if changes in force perception are found, are they due to an insidious loss of contractility in response to localized muscle fatigue, perceptual masking of proprioceptive stimuli produced by fatigue and discomfort symptomatology, or both?

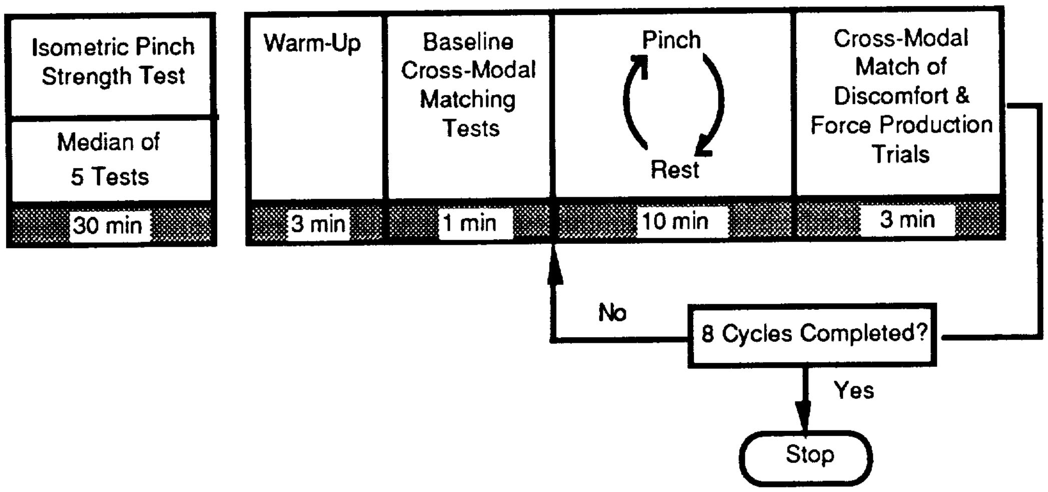
2. Methods and Materials

Subjects. Four male (25.0 ± 7.3 years; maximum pinch force 131.5 ± 13.2 N) and two female (34.5 ± 3.5 years; maximum pinch force 103.0 ± 24.0 N) served as subjects. All subjects reported and appeared to be in good health with no history of musculoskeletal disease. Participation in the experiment was on an informed consent and paid basis.

Apparatus and Methods. Onset and severity of fatigue and discomfort in the hand and forearm, loss of pinch grasp force capability due to fatigue, and shifts in force perception were examined throughout a 104 minute period of cyclic grasping. Subjects performed pulp-pinch grasps with the thumb and index finger of the dominant-hand; the remainder of the digits formed a power-grip. Grasps were initiated and held for 15 s, and were followed by either a 7.5 or 15 s rest period. Subjects rested by maintaining the same hand posture in a relaxed state. Cyclic exertions were continued for 10 minutes. Upon completion of the ten-minute period, subjects performed a cross-modal matching procedure to estimate the severity of fatigue and discomfort in the working hand and forearm. Following this estimate, subjects then performed a force magnitude estimation task which was followed by rest for the remainder of the 3 minute period. The cycle of repetitious exertions followed by a 3 minute interval for measuring discomfort, measuring force perception capability, and then rest, was repeated for 8 consecutive trials.

Each 104 minute test session was completed under a different level of grasp force (i.e. 5, 15, and 25 % of an individual's grasp strength) and work:rest ratio (i.e. 15s grasp: 15 s rest, or 15s grasp: 7.5s rest). The six days of testing required for an individual test subject was completed within a two to three-week period. See Figure 1 for a graphical summary of the experimental design employed.

DAY	WORK:REST CYCLE	PINCH FORCE (%MVC)	TRIAL
1		5	1,2, ... ,8
2	15s:15s	15	1,2, ... ,8
3		25	1,2, ... ,8
4		5	1,2, ... ,8
5	15s:7.5s	15	1,2, ... ,8
6		25	1,2, ... ,8



Isometric Pinch Strength Test
Median of 5 Tests
30 min
Warm-Up
3 min
Baseline Cross-Modal Matching Tests
1 min
Pinch
Rest
10 min
Cross-Modal Match of Discomfort & Force Production Trials
3 min
No
8 Cycles Completed?
Yes
Stop

Figure 1. Experimental design and procedural paradigm.

Figure 2 describes the apparatus used to measure and to control pinch grasp forces. Subjects were seated in front of a computer display with the distal phalanx of the thumb and index finger seated against horizontal tabs mounted atop of two vertical metal struts. An adjustable platform was used to raise or lower the subject's hand to insure that all subjects could comfortably seat their fingers against the struts using the same hand posture. The struts, attached to an immovable base, possessed balanced strain gages which passed stress-induced voltages through an amplifier to an analog-to-digital converter, and subsequently to a microcomputer for recording and statistical analyses.

Two cursors, whose vertical positions on a computer display were independently controlled by normal forces applied to the struts, were presented on a computer screen and used to feedback grasp force magnitudes to the subject. Subjects were instructed to jointly move the cursors to, and to remain on, a visual target for a 15s period. Start and finish of the exertions were timed, and initiation and cessation of the grasps was visually and aurally signalled, by computer. Though distances of cursor movements to the target were fixed, forces required to acquire the target were set to 5, 15, or 25 percent of the individual's measured grasp strength capability. Real-time recording of grasp force was monitored and used to confirm compliance with the experimental paradigm.

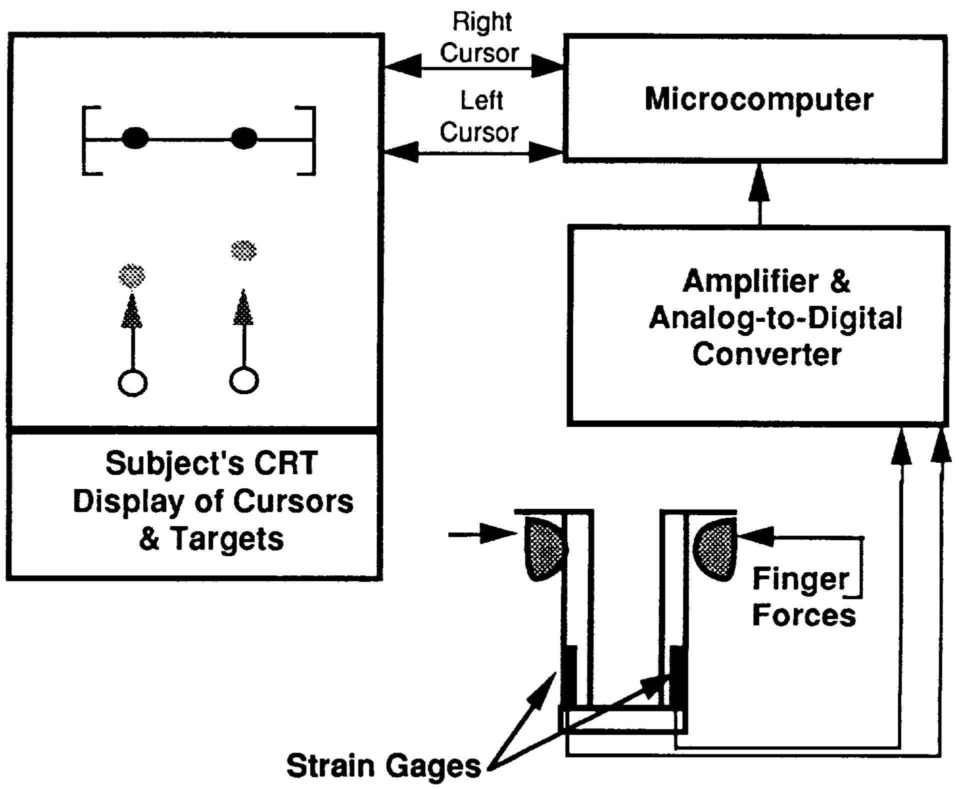

Right Cursor
Left Cursor
Microcomputer
Amplifier & Analog-to-Digital Converter
Subject's CRT Display of Cursors & Targets
Finger Forces
Strain Gages

Figure 2. Experimental apparatus used.

Maximum pinch grasp strength, or maximum voluntary contractions (MVCs), were measured using the apparatus described in Figure 2. Subjects performed a series of 5 maximum effort grasps for a period of 5 s. Average normal forces produced by the thumb and index finger during the last three seconds of the exertion were used as estimates. The median of 5 exertions, separated by 3 minute intervals for recovery, served as the estimate of the subject's maximum grasp strength. The median value was used to set force magnitudes in experimental trials.

Subjects estimated the magnitude of fatigue and discomfort in the hand and forearm using a cross-modal matching method described in detail elsewhere [1]. Subjects adjusted the length of a visual analog scale, a line, which was anchored between no signs and symptoms, and maximum tolerable discomfort. Thus, if the individual judged that the severity of symptoms were equivalent to 50% of their tolerable range, then they adjusted the length of the line displayed on the computer screen to 50% of its maximum length.

A similar approach was used to gage the ability of a subject to judge grasp forces. Following the procedures outlined by Lodge [3], subjects were presented a series of visual lines of different length and were instructed to match by exerting a grasp force of equal intensity. Thus, presentation of a visual line equal to 50% of its possible length was to be matched by applying a grasp force equal to 50% of the individual's perceived strength limit. Following a series of 5 trials, log-transforms of 2 s averages of stable force estimates (i.e. subjects reported when they believed that they had reached a stable exertion equivalent to the given line length) were plotted against log-transforms of line-lengths (i.e. 5, 20, 35, 50, and 65% of maximum line length), and a least-squares regression analysis was performed to estimate the slope of the plotted line. Forces produced during the 5 estimates were also recorded for subsequent analysis.

3. Results

Analysis of day-to-day baseline force magnitude estimation performance revealed no significant changes in average forces used to perform grasps, nor in the slopes of the force magnitude estimation test (p>.10) across test days. See Figure 3.

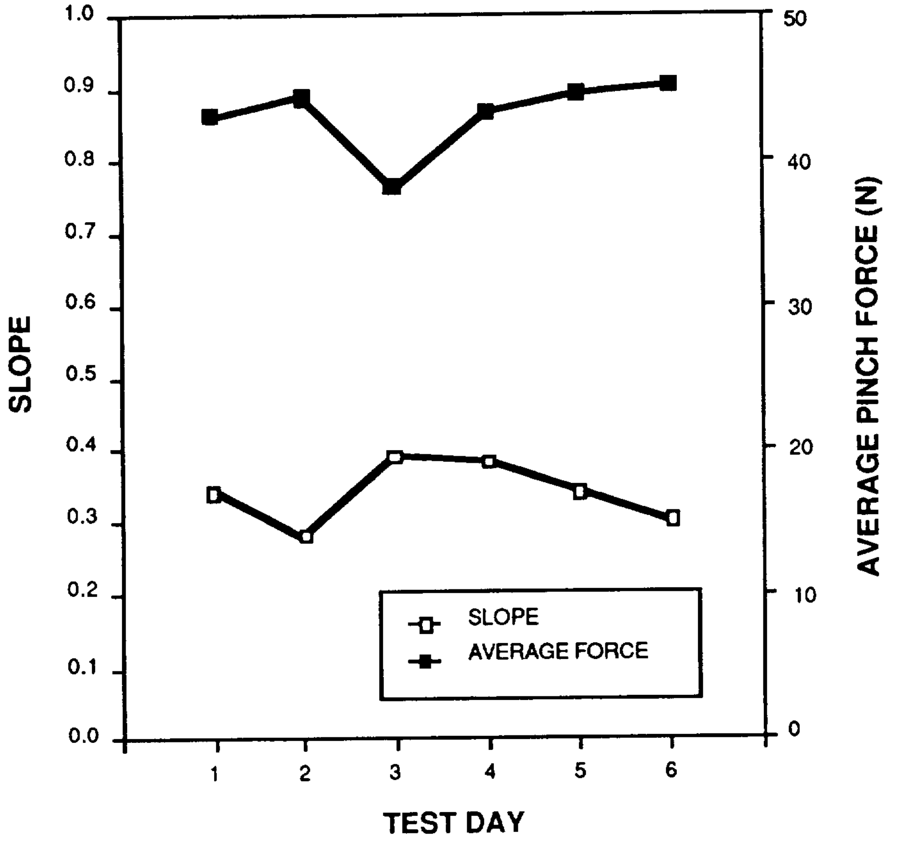


Figure 3. Force magnitude estimate slopes and average grasp forces produced during baseline tests.

Matching intensity of discomfort using a visual analog scale showed significant increases in discomfort occurred in response to increased pinch grasp force levels (F=181.2, p<.05), work:rest ratios (F=13.4, p<.05), and with progression of time (F=25.1, p<.05). Impact of work:rest ratio was significant only when subjects were exerting force levels of 25 percent of MVC (F=12.6, p<.05). All remaining treatment interactions were not statistically significant (p>.10). See Figure 4.

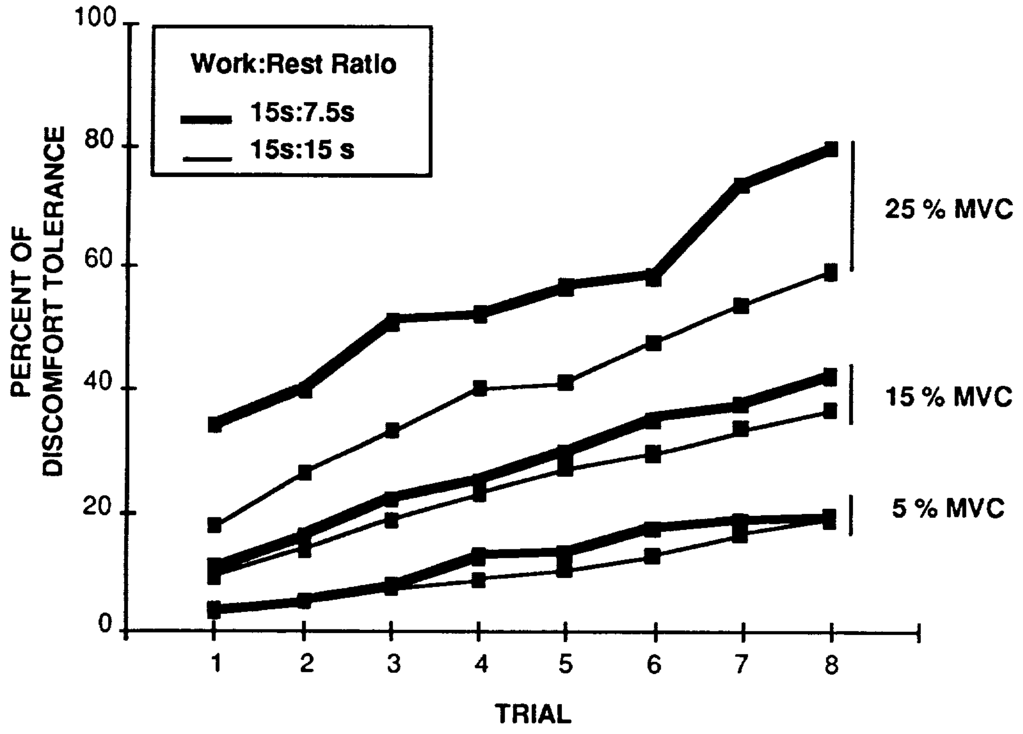


Figure 4. Percent of discomfort tolerance plotted as a function of pinch grasp force, work:rest ratio, and progression of test session.

Slopes of force magnitude estimation functions declined following test sessions in which pinch grasp forces exceeded 5 percent of MVC (F=4.4, p<.05), or when work:rest ratios were increased from 1 to 2 (F=9.9, p<.05). As shown in Figure 5, declines in slopes in response to increasing pinch force occurred only when work:rest ratios exceeded 1.0 (F=3.4, p<.05). All remaining main and interaction effects were neither materially or statistically significant (p>.10).

Unlike psychometric slopes, average pinch forces produced during magnitude estimation trials remained unchanged with only one exception (p>.10). Average pinch force increased slightly when work:rest ratios were increased from 1 to 2 (F=7.4, p<.05). See Figure 6.

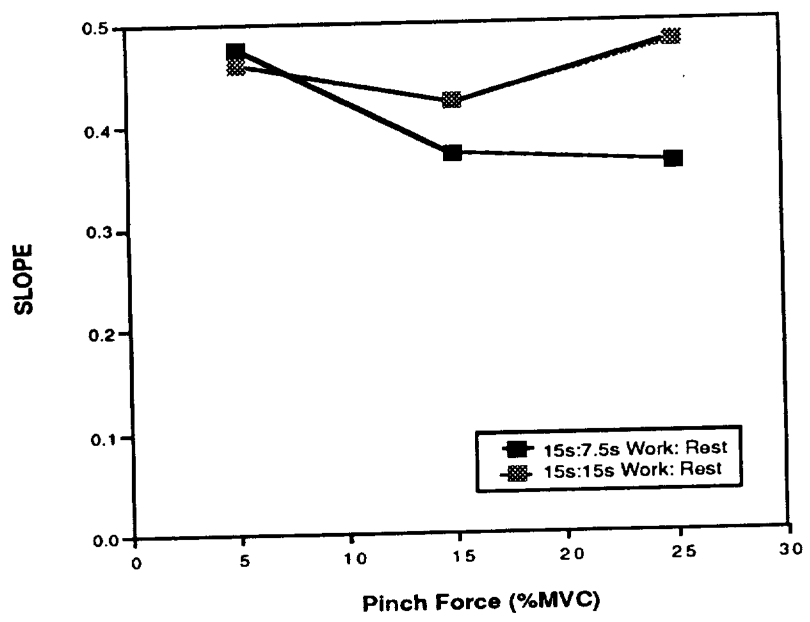


Figure 5. Slope of force magnitude estimation function plotted against pinch grasp force and work:rest ratio.

Correlation analyses showed direct relationships between magnitude of discomfort and magnitude of pinch force (r = 0.80, p<.05) and task duration (r = 0.50, p<.05). Slopes of force magnitude estimates declined when intensity of discomfort (r = -0.40, p<.05) and work:rest ratios (r = -0.39, p<.05) increased. Declines in slopes were also accompanied by increased average pinch force (r = -0.48, p<.05). All remaining correlations were not statistically significant (p>.10).

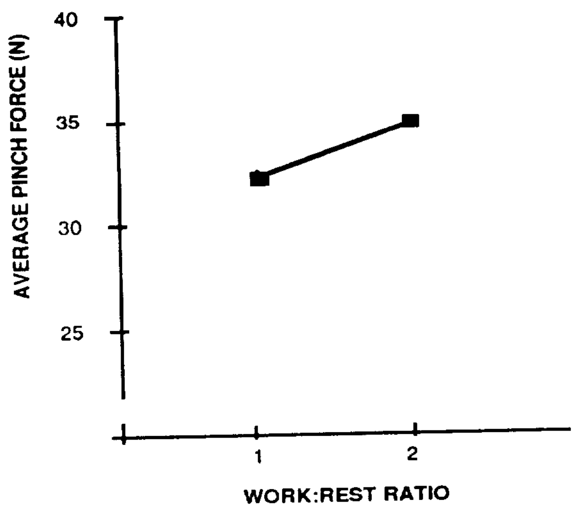


Figure 6. Average pinch grasp force during force estimation trials plotted across work:rest ratio.

4. Discussion and Conclusions

Subjects experienced some degree of fatigue or discomfort symptomatology even at relatively low levels of exertion (i.e. 5 % of MVC). At higher levels of grasp force (i.e. 25 % MVC) significant levels of discomfort were encountered in as little as 10 minutes. Discomfort, regardless of initial exertion level, continued to build with progression of the task in a constant manner. Work:rest ratios, or length of time provided to recover from the immediate consequences of the exertion, produced little or no effect until exertions exceeded 15 % of MVC. It is noteworthy that subjects rarely complained of discomfort or fatigue in musculature located in the forearm (i.e. the principal flexors of the digits). The thumb and index fingers, and tissues directly underlying finger contact on the smooth flat strut's surface, were the chief loci of discomfort. The direct mechanical stress could be tolerated at 15 % MVC with moderate reports of discomfort after 104 minutes. However, at 25 % of MVC a few subjects were near their tolerance limit, and would probably have been unable to complete a two-hour task. In earlier pilot experiments, some of our subjects were unable to complete the 104 minute protocol when grasp forces equalled 25 % of MVC.

Significant negative shifts in slopes of psychometric functions were found immediately when exertions equalled or exceeded 15 % of MVC and work:rest ratios were increased to 2.0. If subjects were provided sufficient rest between exertions (e.g. 15 s) then psychometric functions remained stable; regardless of exertion magnitude or level of persistent discomfort. There was no evidence to support the conclusion that shifts in psychometric functions were strictly a result of insidious loss of muscle contractility. Slopes pivoted about mid-range force estimates and were accompanied by significant elevations in slope intercept magnitudes. Subjects, thus, produced larger than expected forces when called upon to produce small exertions (i.e. 5 to 35 % of MVC), and smaller than expected exertions when forces equalled or exceeded 50 % of MVC. This finding, along elevations in force production occurring concomitantly with flattening slopes, suggests that subject's perceptions of exertions were probably perceptually-masked by ancillary sensations of discomfort. Maximum voluntary contractions are based upon both muscle contractile force capability and volitional tolerance of exertion-induced discomfort. Thus, reductions in force production, when significant levels of exertion were required, may reflect lost contractility of tasked musculature, a reduced volitional tolerance for additional exertion-induced discomfort, or both factors. It is interesting to note, however, that given sufficient time for masking effects, and perhaps loss of contractility, to decay, force magnitude estimation performance remained stable.

Telemanipulation is often characterized by repeated and sustained grasps of objects which are comparable to those studied in this experiment (e.g. object transport, part or tool transfer from one end-effector to another, or assembly or disassembly activities). Under such conditions operators of comparable master-controllers are likely to rapidly develop low to moderate levels of localized discomfort in the hand and fingers when forces of operation or reflected forces approach 15 % of MVC. If exertions reach 15 % of MVC, and the operator is not provided substantial rest between exertions, then shifts in force perception can occur. Operators will over-force when exertion requirements are low, and underforce when grasp requirements are substantial.

Aside from the consequences of less delicate grasping, overforcing of grasps serves to further provoke, or to at least maintain, fatigue and discomfort symptomatology and its negative performance consequences. Negative shifts in psychometric functions found in this study can also result in inappropriate interpretations of the magnitudes of large forces reflected to the master-controller, and in underproduction of grasp forces required for more rigorous manipulation activities. Signs and symptoms of localized discomfort and fatigue always preceded untoward shifts in force perception. Unfortunately, the presence of discomfort or fatigue in the hand occurs rapidly and in the absence of psychometric shifts; thus, discomfort cannot be used as a reliable indicator of shifts in force perception.

5. References

1. Wiker, S. F., Chaffin, D.B., and Langolf, G.D. (in press) Shoulder posture and localized muscle fatigue and discomfort. **Ergonomics**.

2. Sharum, J. and Chaffin, D.B. (1970) Force judgement decrements following muscle exertions. Technical Report. Human Performance Research Group, Department of Industrial Engineering, University of Michigan, Ann Arbor.

3. Lodge, M. (1981) **Magnitude Scaling: Quantitative Measurement of Opinions**. London: Sage Publications.

N90-29010

MEASUREMENT OF HAND DYNAMICS IN A MICROSURGERY ENVIRONMENT: PRELIMINARY DATA IN THE DESIGN OF A BIMANUAL TELEMICRO-OPERATION TEST BED

Steve Charles, M.D. and Roy Williams
Center for Engineering Applications
Memphis, Tennessee

ABSTRACT

Data describing the microsurgeon's hand dynamics has been recorded and analyzed in order to provide an accurate model for the telemicrosurgery application of the Center's Bimanual Telemicro-operation Test Bed. The model, in turn, will guide the development of algorithms for the control of robotic systems in bimanual telemicro-operation tasks. Measurements were made at the hand-tool interface and include position, acceleration and force between the tool-finger interface. Position information was captured using an orthogonal pulsed magnetic field positioning system resulting in measurements in all six degrees-of-freedom (DOF). Acceleration data at the hands was obtained using accelerometers positioned in a triaxial arrangement on the back of the hand allowing measurements in all three cartesian-coordinate axes. Force data was obtained by using miniature load cells positioned between the tool and the finger and included those forces experienced perpendicular to the tool shaft and those transferred from the tool-tissue site. Position data will provide a minimum/maximum reference frame for the robotic system's work space or envelope. Acceleration data will define the response times needed by the robotic system in order to emulate and subsequently outperform the human operator's tool movements. The force measurements will aid in designing a force-reflective, force-scaling system as well as defining the range of forces the robotic system will encounter.

All analog data was acquired by a 16-channel analog-to-digital conversion system residing in a IBM PC/AT-compatible computer at the Center's laboratory. The same system was also used to analyze and present the data. It is anticipated that the data will also provide needed information for other tasks to be explored at the Center. These include telemicrosurgery in remote locations such as space, telemicroassembly of computer, electro-optical and small mechanical systems, telemicromanipulation of miniature sensors and telemicrohandling of hazardous material.

INTRODUCTION

Procedures in the medical, biological, industrial and military fields are continually being down-sized. It is not uncommon for the microsurgeon to perform 150 to 200 micron movements during an operating procedure. In many cases the dextrous microsurgeon would like to be able to decrease these movements (1-10 microns for operation on small vessels near the retinal surface or inner ear) but cannot due to the limits of human dexterity. For those surgeons where dexterity is absent (old age or inexperience) but the expert knowledge exists, a means for improving that dexterity must be explored. Likewise, as the experimental biologist explores further and further into human anatomical structures and physiological processes (for example, manipulation of 1-10 micron patch clamp sensors), the need for smaller handling systems and exact placement of micro-instrumentation sensors arises. Finally, with the emergence of microsensors and microdynamical systems (MDS - mechanical devices and systems fabricated on silicon) assembly and inspection becomes a micromanipulation task since these devices are typically on the order of 20-200 microns [1]. These tasks will push the limits of human dexterity just as seen in the microsurgery field.

The Center for Engineering Applications (CEA) has begun design and development of a Bimanual Telemicrorobotics Test Bed featuring high-precision robotic manipulators to address these dexterity needs. This paper's primary focus is on preliminary data describing the microsurgeon's hand dynamics. These measurements and the test bed concept and their benefit to NASA space telerobotics will also be explored.

We feel the benefits exist in three primary areas. First, the concept of telemicro-operation for microsurgery can be extended to remote locations just as current work with telerobots for the Shuttle Remote Manipulator System and the Space Station systems. For extended missions an on-board surgery capability can eliminate the costs of returning the injured crewman to earth or delivering the microsurgeon to the patient site. Secondly, we feel the surgeon-machine interface (SMI) we are currently developing will aid NASA in eliminating some of the dexterity problems currently encountered in pressurized spacesuits. The interface will be small enough to fit within the spacesuit allowing the astronaut to manipulate tools attached to the end of the spacesuit arm. This will enable the astronaut to remain at the site (as opposed to manipulation within the base vehicle) for those operations in limited access areas. Finally, the test bed may also be used to manipulate samples in experiments best done in the gravity-free environment. The Space Shuttle has already performed such experiments and proven the concept. The test bed can further that ability by allowing down-sized manipulation of such samples (viral strains, microelectro-optics, micromechanical systems, etc.).

EXPERIMENTAL APPARATUS

All data was collected in the surgical environment using a Northgate PC/AT-compatible system as the host. This system operates at 12 MHz, has 1 MByte of RAM, an 80287 math co-processor and a 60 MByte hard disk. Position information was acquired using the Polhemus 3SPACE Isotrak system. The Isotrak connects to the host via a serial link capable of 19.2K baud. With data averaging incorporated position updates occur every 35.7 msec where translational resolution is 0.762 mm RMS and rotational resolution is 0.12° RMS. The Isotrack source was mounted on and above an aluminum tripod with the sensor mounted on the selected operating tool. The sensor weighs approximately 23 grams and did not adversely affect the surgeon's operation (there was some limitation due to the data cable and physical dimensions of the sensor and is discussed in the Methodology section).

Acceleration data was captured using a Vibro-Meter triaxial accelerometer (model CE505M201) connected at the back of the surgeon's hand. The accelerometer has a sensitivity of 10 mV per g at +/-10% with a frequency response of 2 Hz to 15 kHz at +/-3dB. The unit requires 15 V at 0.5 mA and was powered by a temperature compensated battery power supply. This device weighs 10 grams and also requires a cable for power and data transmission.

All force data was recorded with A.L. Design's model ALD-Micro-1 Load Cell which uses a full four arm internal Wheatstone bridge using bonded strain gages. Two cells were used to record data simultaneously. Each cell has a full scale operating range of 450 grams and features a combined error (linearity, hysteresis and repeatability) of +/- 0.5% FS. The total range of force was reduced for our experiment thus reducing this error. Each cell requires 5 Volt excitation at about 15 mA which was supplied again by a temperature compensated battery supply.

All analog data from the accelerometers and force transducers was digitized using a MetraByte DAS-16F A/D converter board residing in the host PC. This board uses a 12-bit successive approximation converter capable of 100 kHz conversion rates. It features 8 differential channels (90 db CMR) and is software programmable for conversion timing. The input features a high input impedance instrumentation amplifier with switch selectable gains. Output of both the accelerometer and the force transducer was quite low (tenths of mVolts) and thus external amplification was used. Three Tektronix AM502 Differential Amplifiers were used as buffers between the transducers and A/D converter board. These units feature switch selectable gains with an input impedance of 1 Mohm (the output impedance of the force transducer was 350 ohms nominal while that of the accelerometer was 1500 ohms nominal) thus providing no-load to the sensors. A dc-offset adjustment is also available as well as a bandpass filter. Figure 1 describes a block diagram of the overall instrumentation system. All cables used to connect transducers to the amplifiers and A/D conversion system were shielded and ground loop currents were eliminated.

CALIBRATION

Each system used was calibrated and checked against the manufacturer's specifications. All were found to be well within specifications.

3SPACE ISOTRAK Positioning System

The linearity and resolution of the Isotrak unit was verified using Melles Griot optical positioners featuring 20 micron resolution which is well below the 0.762 mm resolution of the Isotrak. Sensor and source were mounted on non-metallic rods so as to be free of the metal effects of the positioners. Adequate time was given between each reading to allow for any oscillations in the rods due to the movement. Linear regression analysis was performed on each axis (translational and rotational). Worst-case fit occurred for the x-axis translation with a correlation coefficient of 0.9754. All other axes possessed 0.99 correlation or better.

A.L. Design Model ALD-Micro-1 Load Cell

The load cells are delivered with a very detailed analysis of their specifications. Repeatability and linearity data are provided. We verified the specifications using an Ohaus mass calibration set and a Keithley Model 196 Digital Multimeter (DMM) featuring 5-1/2 digit accuracy. The Model 196 is capable of 100 nV resolution and features an input impedance of over 1 Gohm for the load cell range of interest. The load cell was powered with the temperature compensated supply. The cells were tested over their full scale range and were found to have correlation coefficients exceeding 0.999.

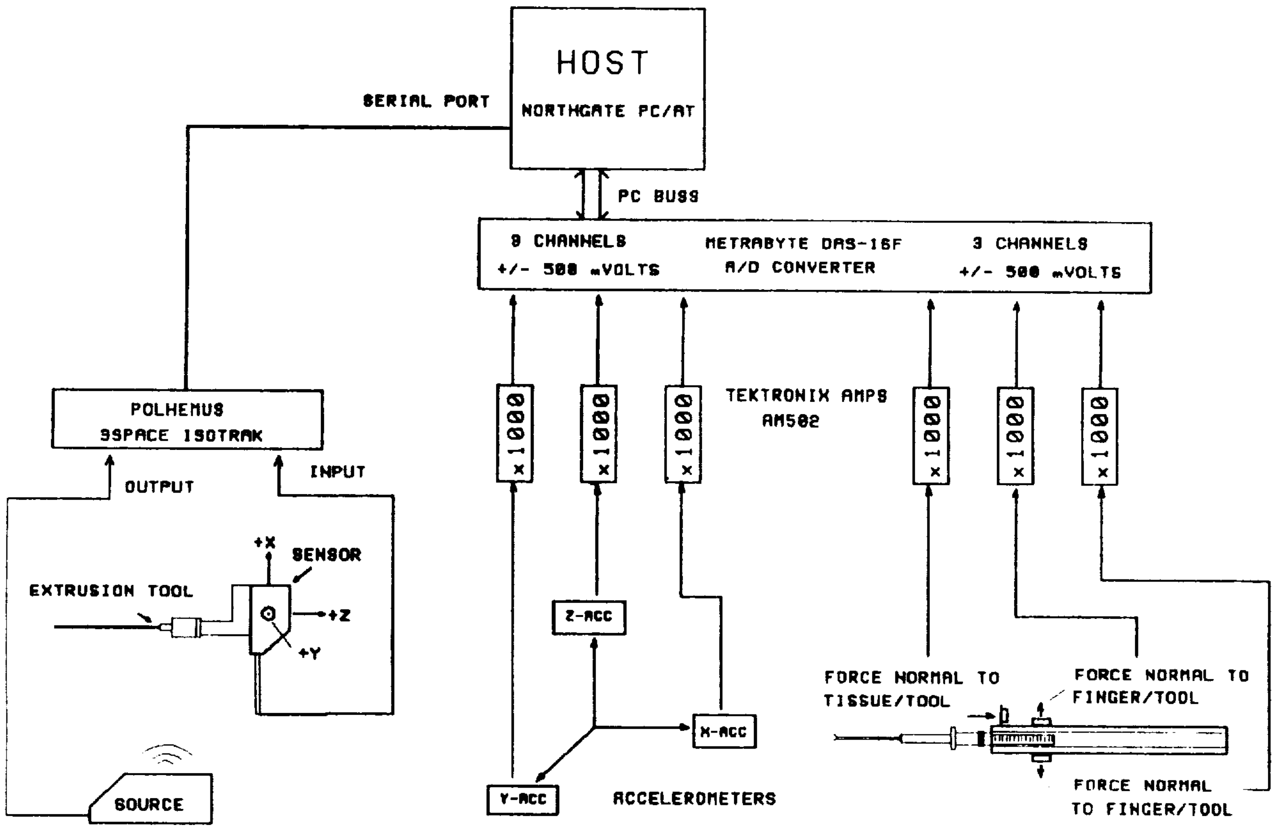


Figure 1. Data Acquisition System.

Vibro-Meter CE505M201 Accelerometer

Vibro-Meter also supplies test data for the accelerometers which includes a noise floor figure, sensitivity and a frequency response graph. At the time of this writing, we had no accurate means of verifying the specifications of the accelerometer. Therefore, those specifications were accepted for the present experiments. We plan to find a means to calibrate the accelerometers at a later date.

MetraByte DAS-16F A/D Converter System

The A/D system was calibrated using a temperature compensated battery source and the Keithley DMM. The range selected for conversion extended from -500 mV to +500 mV represented respectively by -2048 to +2047 digital levels. Linearity was verified to be within +/- 1 bit over the full-scale range.

METHODOLOGY

All data was recorded in actual eye microsurgery and thus sterile management of the transducers was followed. In all cases autoclaving was avoided for fear of damaging the semiconductor components within each sensor. Instead, sterile plastic covers were used to isolate the sensors from the sterile field. Each procedure was designed with both the surgeon and operating room technician providing input regarding sterilization .

Position Measurement

Measurement of position included both the hand and two common instruments found in ophthalmic microsurgery: the straight extrusion tool and the Sutherland Scissors. During all three measurement cycles, the Isotrak unit was in the data averaging mode with the serial communications rate set to 4.8 KBaud. This resulted in a sampling time of approximately 140 msec (about 7 Hz). The host data capture algorithm was written in Microsoft C and basically captured the data in an array while periodically writing it to a hard disk file. At the beginning of each experiment, the source-to-sensor alignment was set allowing data collection from a (0,0,0) orientation. The data was then analyzed using Quattro, a spreadsheet program. Results and discussion of this analysis are presented later in the paper.

For the hand measurements the sensor was inserted in the sterile plastic bag and placed under the surgeon's sterile gloves. The gloves provided a two-fold purpose by further isolating the sensor from the sterile field and providing a very stable mount of the sensor. The data cable for the sensor was routed under the surgeon's gown and over his shoulder to the Isotrack electronic unit. The Isotrak source was mounted approximately 65 cm above and to the right of the sensor on an aluminum tripod. The source could not be placed closer to the sensor due to sterile field conditions. Source and sensor cables were separated and kept away from the host's CRT. Figure 2 describes the orientation of the x, y, and z-axes as mounted on the hand.

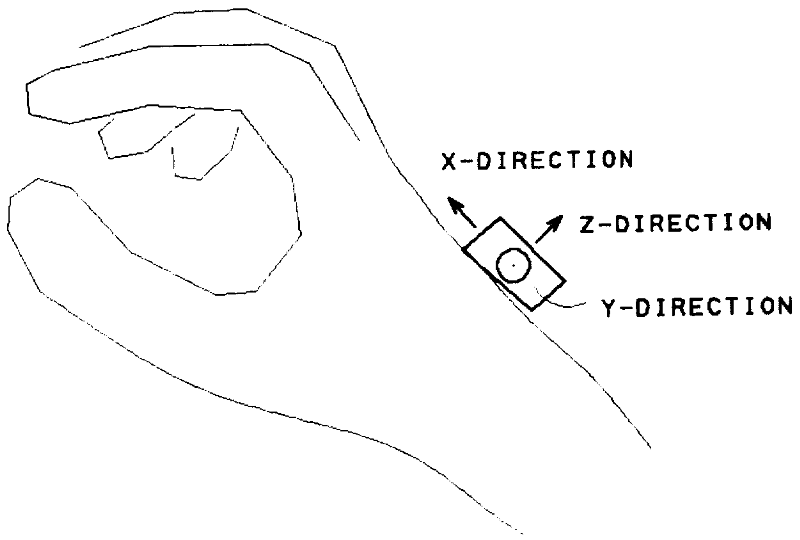


Figure 2. Hand Positioning Study: Magnetic Position Sensor Orientation

The straight extrusion tool does not require extremely precise movements, however, it's range of motion can be quite large as it is often times used to remove trapped gas bubbles within the eye chamber. The magnetic sensor was placed in a sterile bag and attached to the straight extrusion tool by means of a PVC right-angle support. This support fit into the handle of the tool and then provided a wide base to mount the sensor. The sensor was mounted to the support with sterilized tie straps and sterile tape. The data cable was again placed in a sterile bag and routed down the surgeon's arm. Figure 3 describes the mounting and sensor orientation on the extrusion tool.

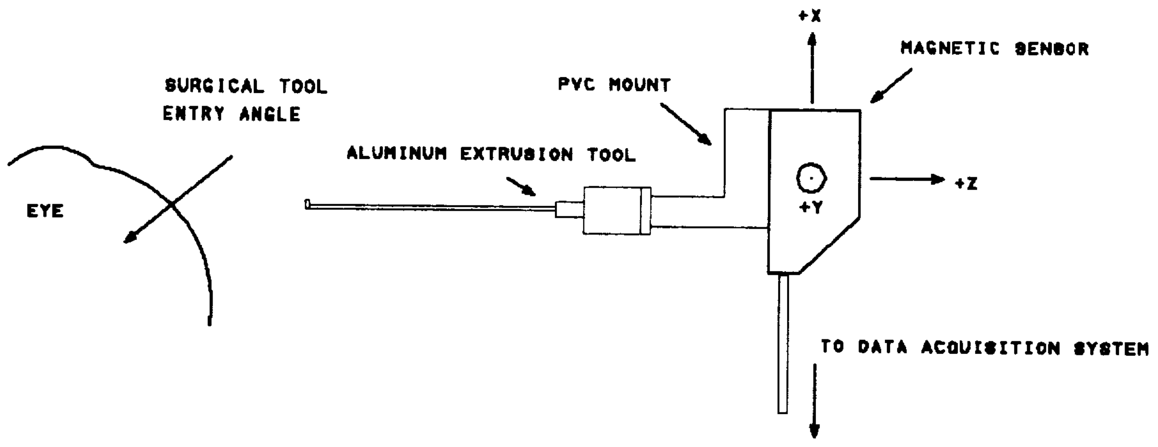


Figure 3. Magnetic Sensor Mounted to the Straight Extrusion Tool

The Sutherland Scissors is an instrument that requires precise manipulation of the tool tip. For example, scissors have become the primary method for most diabetic Epiretinal Membrane delaminations. This technique requires the blade to be parallel to the retinal surface and thus extremely small movements are required [2]. Figure 4 describes the sensor mounting. The scissors are aluminum and again the sensor was placed in a sterile bag and subsequently taped to the scissors handle. It must be noted that the sensor and data cable did impede some movement during surgery not allowing the recording of the full operating procedure. This occurred since the scissors are fairly long causing the cable to strike the operating microscope directly above the eye. However, maximum values defining the workspace were recorded.

Acceleration Measurement

Acceleration was measured in the x, y and z-axis directions for the hand only. The accelerometer was placed in a sterile bag and fitted under the surgeon's glove in much the same way as the position sensor with axis orientation shown in Figure 5. The power/data cable emerged at the junction of the glove and gown due to cable length limitations. The cable did not hinder the surgeons movements. Data was recorded for two different procedures: a forceps membrane peeling operation and a vitrectomy (removal of the vitreous). The forceps operation represents one of the most precise tool manipulations while the vitrectomy represents a wide range of motion. Both operations required the surgeon to remove the tool and re-enter yielding large acceleration values.

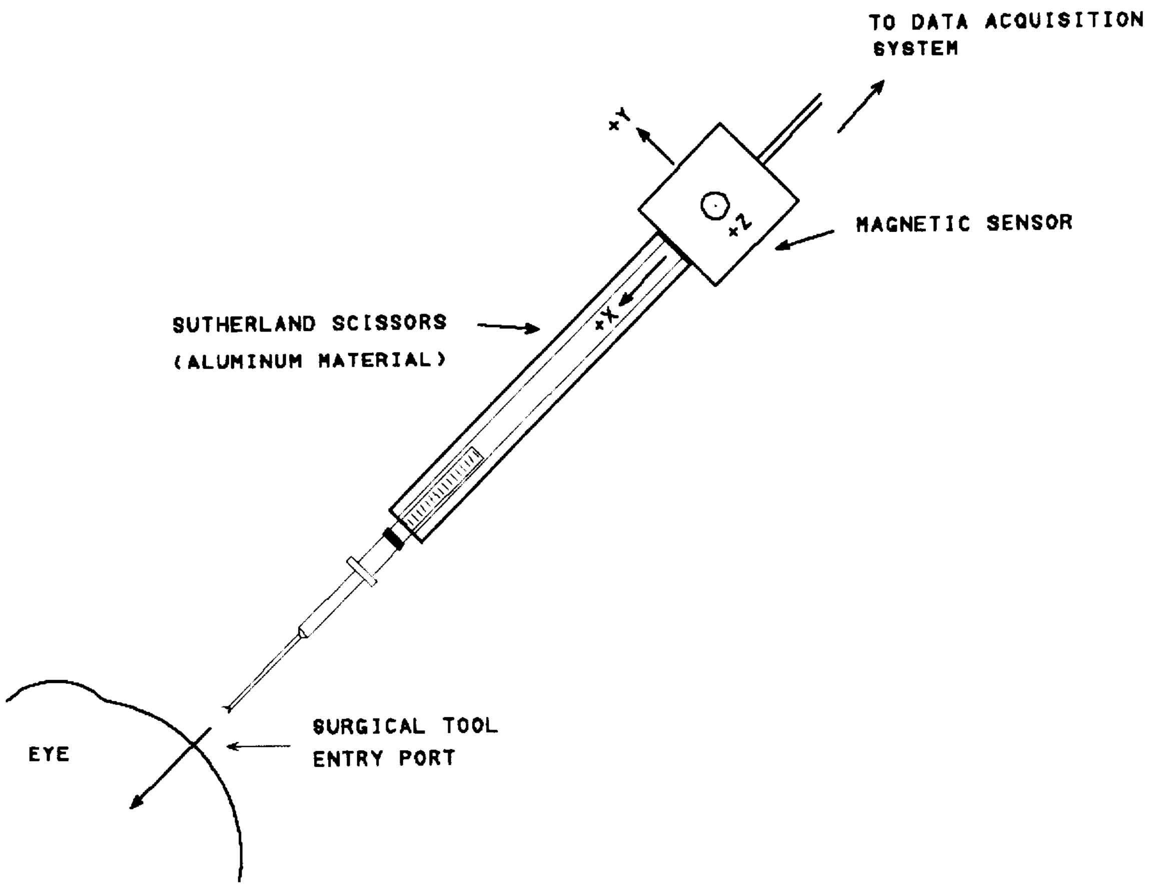

TO DATA ACQUISITION SYSTEM
+Y
+Z
MAGNETIC SENSOR
+X
SUTHERLAND SCISSORS (ALUMINUM MATERIAL)
EYE
SURGICAL TOOL ENTRY PORT

Figure 4. Sutherland Scissors Mounting and Sensor Orientation.

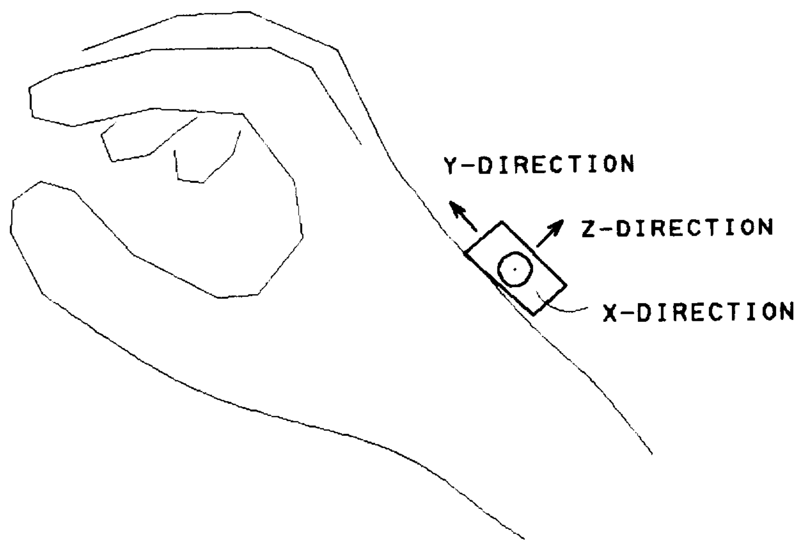


Figure 5. Hand Acceleration Study: Accelerometer Orientation

Force Measurements

Two force transducers were used to record force felt by the fingers at the perpendicular-to-tool interface and the perpendicular-to-tissue interface. Figure 6 describes the sensor mounting on the Sutherland scissors tool. This tool was chosen as it offers one of the most delicate operations due to the presence of the retinal surface. Force transducer 1 was mounted so as to measure tactile sense and the effect of tool weight and configuration on the proprioceptive sense. Force transducer 2 attempts to measure the force felt due to the tissue/tool interface. Both cells were mounted flat against the support structures and with the cell buttons accessible to the surgeon's fingers. The entire upper structure of the Sutherland scissors, the load cells and their power/data cables were placed in the sterile bag. The sterile scissors tip was attached by the operating room technician just prior to use.

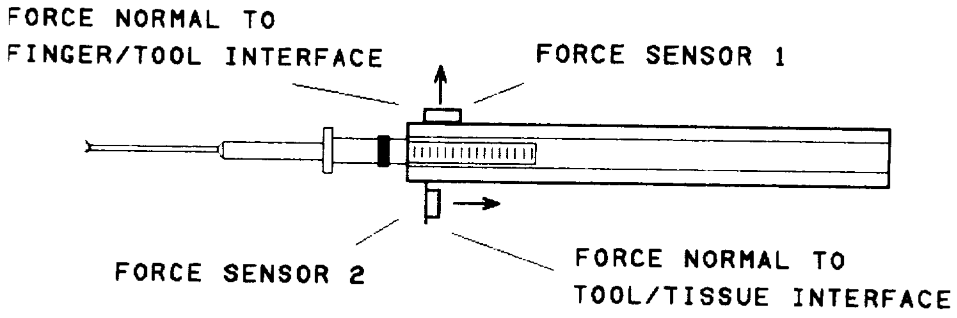

FORCE NORMAL TO FINGER/TOOL INTERFACE
FORCE SENSOR 1
FORCE SENSOR 2
FORCE NORMAL TO TOOL/TISSUE INTERFACE

Figure 6. Force Transducers Mounted to Sutherland Scissors

RESULTS

In order to model the telemicrosurgery application various parameters of the microsurgeon's hand dynamics must be known. This work concentrated on determining the minimum and maximum values of position, acceleration and force. In order to find these values large amounts of data were recorded and subsequently plotted. Due to the page limitation of the proceedings, these plots describing typical performance are omitted. Interested individuals may request this data from the Center for Engineering Applications. We report here those maximum and minimum values which the SMI and manipulators must outperform.

Position or Workspace

On examining the workspace for vitreoretinal microsurgery it appears as a cone within and outside the eye. This is due to the pivot point at the incision. Our data describes the cube that encapsulates this cone at the outside of the eye and therefore represents the workspace of the micromanipulator. Minimum and maximum values are shown below for the two tools studied.

SUTHERLAND SCISSORS - WORKSPACE DATA

	POSITIVE TRANSLATION (cm)	NEGATIVE TRANSLATION (cm)	CLOCKWISE ROTATION (degrees)	COUNTERCLOCKWISE ROTATION (degrees)
X	2.27	-0.11	42.00	0.00
Y	5.65	-2.54	9.99	13.47
Z	6.96	-1.33	20.16	79.45

STRAIGHT EXTRUSION TOOL - WORKSPACE DATA

	POSITIVE TRANSLATION (cm)	NEGATIVE TRANSLATION (cm)	CLOCKWISE ROTATION (degrees)	COUNTERCLOCKWISE ROTATION (degrees)
X	2.58	-2.00	79.38	23.89
Y	0.42	-1.35	18.11	7.44
Z	0.99	-0.65	4.53	114.61

TYPICAL WORKSPACE - OPHTHALMIC MICROSURGERY
(Orientation defined by Sutherland Scissors Sensor Mount)

	TRANSLATION (+ to -)	ROTATION (CW to CCW)
X	2.38 cm	103.27^{o}
Y	8.19 cm	119.14^{o}
Z	8.29 cm	25.55^{o}

Acceleration

Acceleration data represents hand acceleration and not necessarily tool acceleration.

	PROCEDURE: FORCEPS			PROCEDURE: VITRECTOMY		
	X-AXIS	Y-AXIS	Z-AXIS	X-AXIS	Y-AXIS	Z-AXIS
MAX +	0.82 g	0.52 g	0.63 g	0.83 g	0.26 g	0.23 g
MAX -	0.10 g	0.55 g	0.49 g	0.11 g	0.36 g	0.28 g

Force

Force was analyzed for maximum values to determine SMI conditions as well as average force to study loading of the proprioceptive senses. Graphical data indicates that the tool/tissue sensor recorded globe deformation as the tool worked against the incision. We are in the process of estimating the force due to internal eye pressure and tissue. With this data we feel we can estimate the force seen near the tool tip as it encounters tissue. Regardless of the exact force the sensor measured, it still represents the resistance felt by the surgeon.

	FINGER/TOOL: FORCE SENSOR 1	**TISSUE/TOOL: FORCE SENSOR 2**
MAXIMUM	16.20 grams	42.68 grams
AVERAGE	13.19 grams	31.13 grams

DISCUSSION AND FUTURE DIRECTION

The results shown here will aid the CEA in development of the Bimanual Telemicro-operation Test Bed which serves as a first step toward increasing dexterity of the microsurgeon and providing for remote surgery capabilities. Results were within the ranges we expected. For example, the typical eye is about 2.54 cm in diameter and the maximum value we recorded for entry and exit (X-direction) of the tool was 2.38 cm. Y and Z directions are listed at 8.19 and 8.29 cm respectively, and taking into account the approximate 4:1 leverage amplification of the tool (Sutherland scissors), these values become 2.05 and 2.07 cm. Forces measured were close to preliminary data recorded by using a micro-beam scale and having the surgeon press the tool against the scale until he "feels" similar resistance as in the eye. The preliminary measurement was very subjective but was confirmed by two surgeons.

During the development of the test bed, a down-sized surgeon/machine interface will be developed which we believe will aid NASA in solving the dexterity problem encountered in pressurized spacesuits. We are currently working with Dr. Bill Hamel of Oak Ridge National Laboratories in the analysis of the kinematics and coordinate transformations needed for the interface.

As a continuation of this paper's work we are currently developing a measurement system to correlate video images of the surgeon's hand movements to the magnetic positioning system output. We are also developing a method to measure the force and torque felt during surgery in all six degrees-of-freedom.

REFERENCES

[1] Gabriel, K., Jarvis, J. and Trimmer, W.,"Small Machines, Large Opportunities: A Report on the Emerging Field of Microdynamics," Report of the Workshop on MEMS Research, 1988.

[2] Charles, S., *Vitreous Microsurgery*, Williams and Wilkins, 2nd Edition, 1987.

N90-29011

HUMAN FACTORS MODEL CONCERNING THE MAN-MACHINE INTERFACE OF MINING CREWSTATIONS

James P. Rider and Richard L. Unger

U.S. Department of the Interior
Bureau of Mines
Pittsburgh Research Center

Abstract

The U.S. Bureau of Mines is developing a computer model to analyze the human factors aspect of mining machine operator compartments. The model will be used as a research tool and as a design aid. It will have the capability to perform the following: simulated anthropometric or reach assessment, visibility analysis, illumination analysis, structural analysis of the protective canopy, operator fatigue analysis, and computation of an ingress-egress rating. The model will make extensive use of graphics to simplify data input and output. Two dimensional orthographic projections of the machine and its operator compartment are digitized and the data rebuilt into a three dimensional representation of the mining machine. Anthropometric data from either an individual or any size population may be used. The model is intended for use by equipment manufacturers and mining companies during initial design work on new machines. In addition to its use in machine design, the model should prove helpful as an accident investigation tool and for determining the effects of machine modifications made in the field on the critical areas of visibility and control reach ability.

1. Introduction

Because of the unique environment in underground mining, mine workers are exposed to a variety of conditions and stresses that are not common in other industries. Mine equipment is usually massive in size and built to withstand a tremendous workload. Often the design of the mining equipment stresses functionality and production over human operator considerations. This has resulted in mining machines where visibility is at a premium and often the operators must lean outside the safe confines of the operator compartment in order to see and operate the equipment as shown in figure 1. Controls are often unlabeled and placed where they are difficult to reach and distinguish from one another; in panic situations, the wrong control is frequently activated. Because of height restrictions in many underground mines, the operator works in an awkward reclined seating position which leads to fatigue and stress. The Mining Equipment Safety Laboratory of the Mine Safety and Health Administration (MSHA) analyzed all fatal accident reports involving underground coal mine mobile equipment for the years 1972 through 1979. During this period, 350 fatalities were investigated and 126 of the fatalities were attributed to improper control design, inadequate visibility from the operators's cab, inadequate compartment size, operators leaning outside the cab, machines without operator compartments, and poorly designed seats. In summary, approximately 36% of the fatalities involving underground mobile mine equipment for the seven years in question relate to improperly designed operator compartments (1).

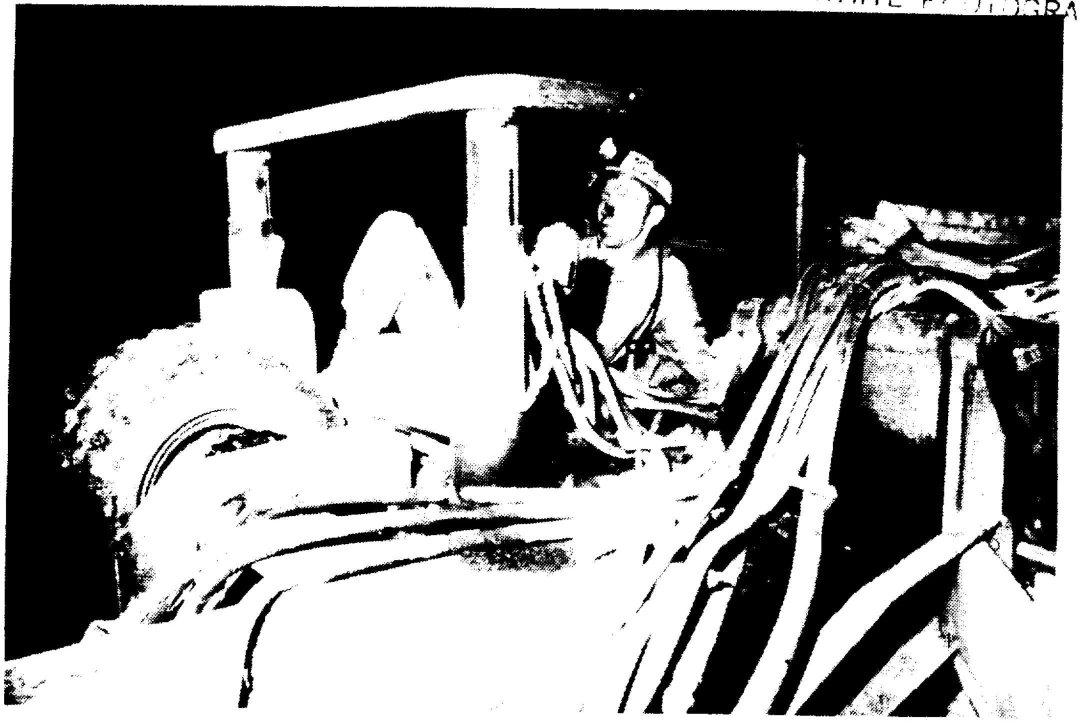


Figure 1 - Miner working in a cramped, visually obstructed environment.

The U.S. Bureau of Mines is charged with conducting research to increase both the safety and productivity of the mineral industries. As part of this research, the Bureau is developing a computer model to assist in the design and analysis, from a human engineering standpoint, of underground mine equipment operator stations. The model known as CAP (Crewstation Analysis Programs) is to be used by original equipment manufacturers and mining companies for the preliminary design work on new machines in terms of good ergonomic design principles. With the implementation of the model, the designer will be given flexibility to experiment with the design of new equipment which is not practical using conventional techniques.

For the sake of simplicity, CAP can be thought of as being composed of the following sections :

1. Input (machine and sample population).
2. Anthropometric analysis.
3. Visibility analysis.
4. Reach analysis.
5. Illumination analysis.
6. Canopy structural analysis.

The initial module defines the operator compartment, mining machine, mine layout and a sample population. The anthropometric section identifies the working posture of the miner and the reach envelop associated with the operator. The operator's field of visibility is determined in third module. The model uses an adaption of the Crewstation Assessment of Reach (CAR) model developed by Boeing Aerospace Corporation (2) to determine whether the controls are reachable for a given population. The fifth module provides an assessment of the illumination requirements of the machine's lighting system and the final section determines the structural strength of the compartment's canopy.

This paper briefly describes the input, reach, visibility, and illumination sections of the model. Interested readers are invited to contact the Bureau of Mines, Pittsburgh Research Center, for more information on the CAP model.

2. Machine and Sample Population Input

Before any analysis can begin, a 3-D, simplified model of the operating station and related machine must be entered into the computer. Often times entering the machine data is a tedious and time consuming process. For this reason a decision was made to develop a method to input the data as quickly and accurately as possible so users may concentrate their efforts on the ergonomic analysis of the machine and compartment. The method chosen takes advantage of engineering layouts drawn in an orthographic projection format and makes use of a digitizing graphics tablet. Polygonal geometries are digitized and the model performs a spatial 3-D reconstruction using a series of functions which identify specific 3-D geometries in the orthographic projection. The program does not contain any high level sophisticated network of geometric concepts but uses a concept similar to the one used by Lafue (3) and Thornton (4). By using the function menu to identify 3-D geometric shapes such as polyhedrons, right circular cylinders, spheres, and wedges the program reconstructs a 3-D object from 2-D orthographic projection drawings (figure 2).

Most of the analysis sections of the CAP model require a sample population for testing. The model allows the user to enter 12 external anthropometric measurements of one or more individuals to build a sample population. Once the 12 measurements have been entered, the model determines the validity of the input and responds accordingly. If the anthropometric input is invalid the model will prompt the user to re-enter the specific measurement in question. If valid data exists the procedure will prompt the user to enter another individual's anthropometric measurements or exit and save the population data base. The 12 external measurements for the sample population are transformed into internal link lengths and link circumferences to create a 3-D link-man. The link-man used in the CAP model is a version of the original link-man in the CAR program (2). Each link is a straight line segment between centers of joint rotation corresponding to human bone structures. The link-man consists of 31 links and is based on work done by Boeing Aerospace Corporation to develop the BOEMAN model (5). CAP computes the link-lengths from external anthropometric measurements through a series of transformations derived primarily from Dempster's (6) analysis of anthropometric data.

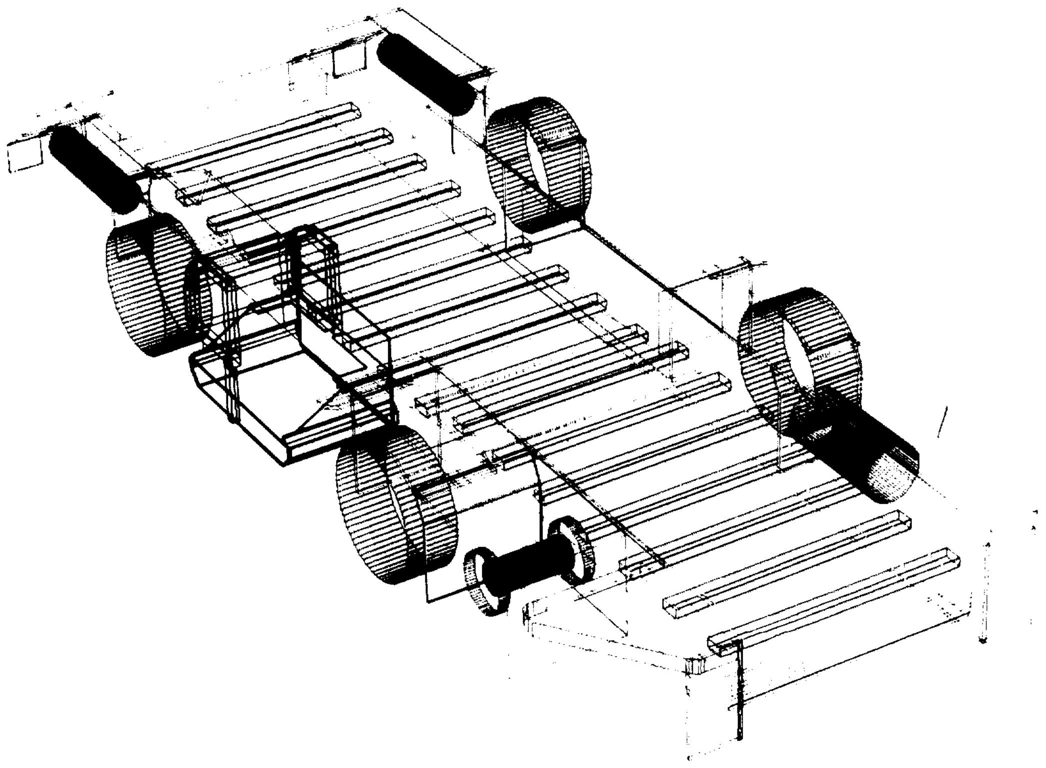


Figure 2 - A 3D representation of an underground haulage vehicle.

Once the input of the machine, compartment and the identification of the sample population is complete, the user is free to choose any analysis section desired. The flexibility of the model will allow the user to modify either the machine input or population sample based on the analysis results.

3. Anthropometric/Reach Analysis

The location of the operator and the placement of the controls is essential in the ergonomic design of operator compartments. The reach analysis sections of the model addresses both issues by using an adaption of the CAR model to position the operator in the compartment and tests if each control is reachable. It consists of a control data input module which define key parameters needed to perform the reach study, and a link-man module which define the anthropometric measurements of a sample population. A tabular printout indicates the percentage of the sample population capable of reaching the defined controls.

The reach analysis module constructs a link-man to reach for a specific point in space defined by a control location. The link-man is built using a link by link approach. The building begins at the lumbar joint which is a function of the seat reference point, the seat pan and seat back angles, and the operator's clothing. Each link is added to the previous link, with the links pointing in the direction of the defined reach point. The model takes into consideration the operator's clothing effect which reduces the angular limits of motion for each link, and alters the link lengths and the position of the joints.

Table 1 - Control Summary Report

Control Name	Hand Foot	Harness Lock	Location/ Required Movement X	Y	Z	% Not Accom	% Accom
STOP BUTTON Zone 3	LH	UNLK	14.5	-11.0	4.3	0.0	100.0
START BUTTON Zone 3	RH	UNLK	16.5	-11.0	4.3	0.0	100.0
FORKLIFT TILT Zone 3	RH	UNLK	32.0	-4.5	11.5	10.0	90.0
		Average	-0.3	-0.1	0.1		
		Worst Case	-0.9	-0.2	0.3		
HEADLIGHTS Zone 3	LH	UNLK	37.5	6.0	9.5	48.0	52.0
		Average	-0.7	0.0	0.2		
		Worst Case	-2.0	0.1	0.7		

The control analysis identifies the name, reach point location, body part used in the reach assessment, grip associated with the body part, and the harness condition for each of the defined controls. It displays the percentage of the sample population capable of reaching each control. In the case of the sample population not being able to reach the control, the program will display the average and worst case distance from the last link to the control's reach point location as shown in table 1.

4. Visibility Analysis

Because of the working environment in underground mining, operator visibility is usually at a premium. To address this problem, the Bureau sponsored research to analyze the visual requirements of mobile mining equipment and to assess the requirements through a computer model. A task analysis to distinguish the visibility requirements for a machine, i.e., what needs to be seen, from the field of visibility, i.e., what can be seen (7) was conducted. The visual requirements were defined in terms of specific locations known as visual attention locations (VALs). Using this method the VALs are specified with reference to a specific machine point and by identifying the VALs in this fashion, the location of the visual features are independent of the length, width, and height of the machine. Figure 3 displays the VALs associated with a underground haulage vehicle (shuttle car) in the fore-aft and lateral planes.

Before analysis can take place, the physical configuration of the operator compartment along with the related machine must be entered into the computer. The model will prompt the user to select and enter the digitized machine and the related crewstation database. The program queries the user for the operator's eye position or designed eye point (DEP), and the focus point. The model will calculate the DEP by vertically building successive links dependent on the defined anthropometric measurements. The focus is defined to be a point in space which forms a line of sight vector with the design eye point. The operator's field of visibility is an important parameter in the analysis of the visual requirements. The recommended visual envelope for an operator to view a working display should be within a 30 degree cone around the principal line of sight (8).

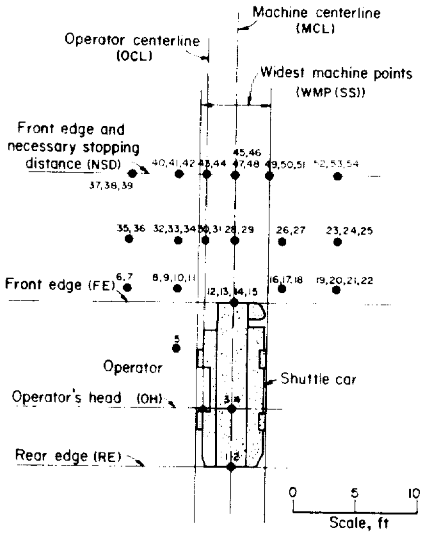


Figure 3 - Visual attention locations (VALs) in the fore-aft and vertical planes of an underground haulage vehicle.

Once the operator's eye view, focus point and the visual envelope dimension has been identified, the model allows the user to create an operator's eye view of the surrounding environment for an assessment of visibility of the VALs. The view from the operator's station is drawn, taking into account the field of vision, focus point and eye position. The user notes which VALs are visible and what modifications may be necessary in the machine design. The analysis is performed repetitively, with the user making judgments as to where the three-dimensional link-man should look. Figures 4 and 5 illustrate views from the cab of a shuttle car.

5. Illumination Analysis

The model addresses the lighting problem by providing the mining industry with a computerized method of evaluating mine illuminations systems. Proposed lighting configurations may be quickly analyzed without resorting to time consuming methods of building physical mockups and taking manual lighting readings. The model analyzes any illumination system relative to an underground mining machine and calculates the incident illuminance or illumination (in footcandles) levels on any surface surrounding the machine. The calculations are be performed so that they compare with MSHA's method for evaluating a mine illumination system.

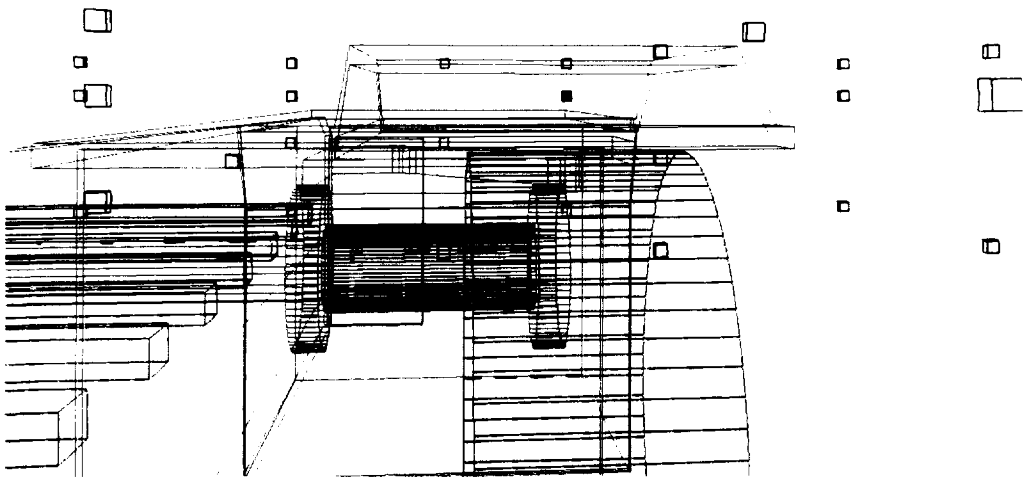


Figure 4 - A 5% female sitting in an operator compartment of a shuttle car looking at VAL #41. Note the VAL is obstructed.

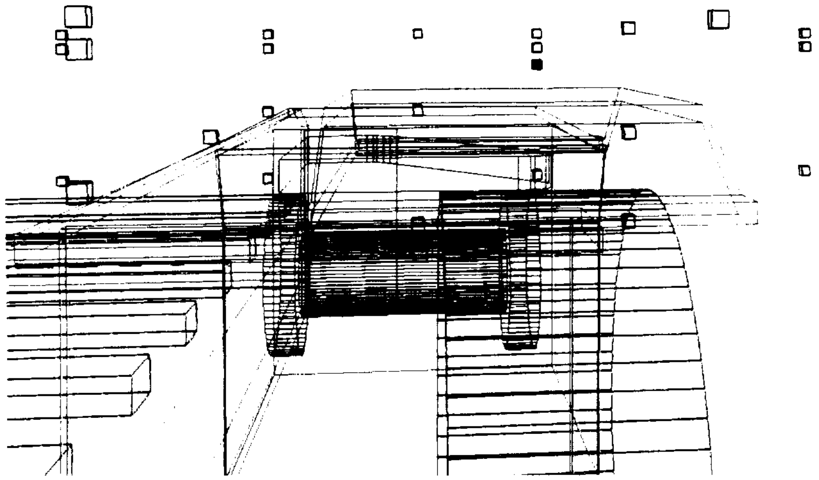


Figure 5 - A 95% male sitting in an operator compartment of a shuttle car looking at VAL #41.

In calculating incident illuminance levels, the model simulates the method of measurement used by MSHA's standard test and evaluation approach. The basic formula for calculating the illumination at any point around a machine is:

$$E = I/D^2$$

where :

- E = illuminance, footcandles
- I = intensity of the light along the vector between the light source and the point where the light is being measured, candles
- D = distance between the light source and the point, feet

The effect of objects in the path of the light source along the location of the measurement point must be taken into consideration when performing an illumination analysis. The illuminance at a specific point is the vector sum total of the illuminance from all luminaries contributing light to that point. Light is considered to a vector originating at the lamp and ending at the measurement point. If the light vector intersects a plane of the machine before reaching the measurement point, the effects of that light are canceled out of the illumination equation. MSHA requires that incident light measurements must be taken for each 2 by 2 ft area on the mine surfaces that have to be illuminated. The model divides each of the defined surfaces in a rectangular grid, 2 ft apart and computes the incident illumination level at each point. It determines the average illuminance in each square by averaging the four grid points associated with each square. Any value that is below the minimum permissible level will be marked in the grid and displayed to the user.

6. Conclusion

Original equipment manufacturers and mine operators will have the capability to use the Crewstation Analysis Programs (CAP) model as a research tool and as a design-aid in the development and modification of new and existing mining machines. The model has the capability to quickly analyze machine mounted illumination systems, identify the visual requirements of mining machines, maximize the operator's visibility, and optimize the location of the controls to accommodate an operator sample. This may allow the designers an opportunity to experiment with the development of new equipment and aid designers to better human engineer underground mining equipment.

7. References

1. Sanders, M.S., and J.M. Peay. Human Factors in Mining, BuMines IC 9182. 1988, 153 pp.

2. Harris, R.M., J. Bennett, and L. Dow. CAR II - A Revised Model for Crewstation Assessment of Reach (contract N62269-79-C-0235, Boeing Aerospace). Naval Development Center, June 1980.

3. Lafue, G. Recognition of Three Dimensional Objects from Orthographic Views. Computer Graphics, Volume 10, 1976.

4. Thornton, R.W. Interactive Modeling in Three Dimensions Through Two Dimensional Windows. CAD 78, pp 204-211.

5. Ryan, P.W., H.N. Sather, and A.W. Bearee. Cockpit Geometry Evaluation Phase II (final report D162-10126-2). Boeing Aerospace, Seattle Washington, 1970.

6. Dempster, W.T. Space Requirements of the Seated Operator (report WADC-TR-SS-159). Wright Air Development Center, Wright-Paterson Air Force Base, Ohio, 1955.

7. Sanders, M.S., and G. Kelly. Visual Attention Locations For Operating Continuous Miners, Shuttle Cars, and Scoops (contract J0387218). BuMines, 1981.

8. Woodson, W.E. Human Factors Design Handbook. McGraw Hill, New York, 1981, 1049 pp.

N90-29012

DEVELOPMENT OF A FLEXIBLE TEST-BED FOR ROBOTICS, TELEMANIPULATION AND SERVICING RESEARCH

Barry F. Davies

British Aerospace PLC, Sowerby Research Centre
P.O. Box 5, Filton, Bristol BS12 7QW
United Kingdom

Abstract

This paper describes the development of a flexible operation test-bed, based around a commercially available ASEA industrial robot. The test-bed was designed to investigate fundamental human factors issues concerned with the unique problems of robotic manipulation in the hostile environment of Space.

1. Introduction

The work to be described here forms part of a contract placed on British Aerospace Space and Communications Division at Stevenage by the European Space Agency, entitled "Teleoperation and Control." The study to be performed will be carried out by a team under the technical lead of the British Aerospace Sowerby Research Centre Human Factors Department, Filton. The remainder of the study team comprises of the United Kingdom Atomic Energy Authority Laboratories at Culham and Harwell.

The overall objective of this study is the development and delivery to the European Space Agency of a flexible (modular) teleoperation test-bed, based around a commercially available ASEA industrial robot, which will permit the experimental investigation of a variety of approaches to the control and supervision of robotic manipulation in Space. Experiments were designed to allow a comparative study of the control and display concepts in order to optimise and evaluate the use of sensor supplied feedback information to a human operator.

The experiments were designed with two mission models as a baseline:

Mission Model 1. The human operator is located on the ground whilst controlling a manipulator which is located on a free-flying servicing vehicle.

Mission Model 2. A human operator is located on the aft deck of the shuttle, controlling a manipulator servicing a payload in the cargo area.

In addition to providing information about the above scenarios, the project will address five overall objectives:

- To analyse specific MMI/human factors issues concerned with the unique problems of teleoperation.

- To provide data which will permit the optimisation of the man-machine interface before delivery of the experimental workstation to ESA.

- To provide results which may be used to formulate future research programmes, to determine the Human Computer Interaction requirements which can be satisfied in the short and long term and to determine possible roles for telepresence strategies.

- To provide results which may be applied in identifying options for the design of future workstations.

- To establish a test-bed facility and research programme which demonstrate a sound European capability in teleoperation workstation design and in research into remote handling and inspection in Low Earth Orbit (LEO).

2. Overall Test-Bed Concept

The development and use of an experimental test-bed stems from the need to establish a programme of experimentation and research into teleoperation in Space. It is important that in achieving this aim the test-bed should remain as flexible as possible. This allows the test-bed to be used both for future research programmes and for the development and implementation of new systems.

The programme of test-bed development has been driven by past theoretical research (e.g. Milgram et al, 1983; Sheppard et al, 1986) and by developments in the fields of the ESA crew workstation (CWS) and EUROSIM projects.

To obtain and maintain the maximum degree of flexibility, the test-bed was developed along modular lines. The modules were defined as follows:

- Subject control and display station
- Experimenter's control and display station
- Video system
- Robot system (ASEA IRb6)
- End-effector and sensors
- Task box and work area. (See Figure 1)

The requirements for the test-bed, workstation and video system were dictated by the nature of the experiments and pilot studies to be carried out using them. The pilot studies and experiments were concerned with the following issues:

- Time delay selection (Pilot Study 1)
- Control law optimisation (Pilot Study 2)
- Controller selection (Experiment 1)
- Feedback optimisation (Pilot Study 3)
- Comparison of feedback options (Experiment 2)

Since the test-bed is an evaluative and experimental tool its design and requirements for its construction were preceded by detailed assessments of the evaluative studies to be performed upon it.

3. Definition of Research and Experimentation

The main objective of the teleoperation and control study is to perform an experimental and comparative evaluation of a variety of concepts regarding the human operator's efficient use of control and feedback options. Previous research studies have been predominantly theoretical in their approaches to key human factors aspects of teleoperation in Space. Therefore, it is appropriate to address these issues from a practical perspective. Both quantitative and qualitative data will be collected in order to provide as complete an evaluation and analysis as possible.

Experimental Designs

The specific designs of the experiments have determined the hardware and software requirements for these investigations. However, the need for flexibility and modularity was not overlooked in this process so as to allow easy implementation of future developments and additions. It is important that there is a logical progression from one experiment to the next. For this reason, a number of pilot or optimisation studies will be undertaken. These will help the experimenters select or optimise the characteristics of the independent variables which will be used in the main experimental designs. This series of investigations places emphasis on both sensory and motor variables. The series of designs does not deal with these two key areas in isolation, but instead the emphasis of the designs will change from one area to the other as the experiments progress.

At the end of this series of investigations, the main sensory and motor factors which influence remote teleoperation will have been considered. The results of these studies will be fed directly into the development of the flexible teleoperation facility for ESA.

The experimental designs are outlined below:

Pilot Study 1. To identify from existing literature and preliminary experimentation, three time delays for use in Experiments 1 and 2.

One of the major areas of concern in the field of Space teleoperation is the performance penalties which are incurred by the human operator controlling systems in the presence of a transmission time delay. Some research has been performed, but this relates specifically to tracking tasks in the presence of a time delay. In the present investigation, consideration will be given to both tracking and pick-and-place tasks.

Three time delays will be considered for use in the main experiments:

(a) Human perceptual threshold. Identified as 60 milliseconds in Sheppard et al (1986).

(b) Human control disruption (i.e. a time delay which produces 'stop and go' motions).

(c) Time delay for LEO: maximum delay likely to be experienced in a Space teleoperation system.

This pilot study will attempt to establish a representative time delay which will produce a human control disruption. Therefore, the result of this pilot study will be the specification of a control time delay which would interfere with <u>all</u> operators' ability to perform the tasks.

The third time delay which will be used in Experiments 1 and 2 will be representative of the time delay which would be experienced when controlling a task in LEO from the ground. This time delay will be identified from literature concerned with satellite/ground station communication links.

<u>Pilot Study 2</u>. To specify and compare control laws for a variety of control devices with the overall aim of selecting the most suitable control law for each device and to optimise the law more finely on the basis of this selection.

Three different joysticks and configurations have been selected for this pilot study. These joysticks will be used in the joystick comparison and evaluation study (Experiment 1), but the control law for each individual device must be optimised before the main study can be undertaken. The joysticks for which the control laws are to be optimised are as follows:

(a) 2x3 axis joysticks. Translational movements of the robot end-effector will be controlled by a three-axis horizontally mounted, whole-hand joystick. Rotational movements of the end-effector (hence changing the orientation of the end-effector at the same point in Space) will be controlled using a three-axis, vertically mounted, whole-hand joystick.

(b) Single multifunction joystick. This joystick will be used to control both translational and rotational movements. Selection of either translational or rotational modes will be made via a switch mounted on the joystick itself. The joystick chosen for this evaluation is the same as the rotational controller used in the previous configuration (a).

(c) DFVLR hand controller. This device is a ball controller capable of controlling six independent degrees of freedom. The controller is a force-torque device with only a perceptibly small degree of motion in each axis.

With these two pilot studies completed, Experiment 1 can commence.

<u>Experiment 1</u>. To evaluate and compare a variety of control devices for the task of remote handling in the presence of time delays and to provide data for subsequent designs.

In addition to the three joystick configurations discussed in the previous pilot study, a 'replica' controller, kinematically similar to the ASEA robot, will also be considered. All the devices will be compared using the control laws established in Pilot Study 1 (position control for the replica and optimised fixed or proportional rates for the other devices) and the independent variable of time delay (as in Pilot Study 1). (See Figure 2)

Performance measures of whole and part-task time along with whole and part-task accuracy will be taken. In addition to this quantitative data, certain qualitative data such as general operator strategies and workstation interaction difficulties will also be recorded.

The outcome of this experiment will be the specification of a controller which produces the most efficient performance of benchmark operations in the presence of time delays. In the event of interaction effects between controllers and time delays, then the controller deemed, in general, to be the most satisfactory will be selected for use in later experimentation.

Pilot Study 3. The optimisation of remote cameras, display configurations and remote lighting.

The aim of this pilot study is to optimise the various chosen methods of sensory feedback, feedback being the presentation of both visual and auditory information, moreover, to optimise the presentation of this information.

Issues to be addressed include the levels of ambient lighting in the experimental environment and the positioning of artificial lighting to meet human visual requirements and task requirements as outlined in Stone et al (1985).

This pilot study, in conjunction with the lighting, will also consider the positioning of the remote cameras. This will also be carried out to meet both human visual requirements and the requirements demanded by the task.

This pilot study, in contrast to the previous pilot studies, will be of a predominantly qualitative nature, where subjects' comments will be used to determine the optimal location of both lighting and camera configurations.

Experiment 2. The evaluation and comparison of a number of sensory feedback options in terms of their effect on operator performance.

This experiment will incorporate all the results of the previous pilot experiment studies. At this stage the controller or controllers will have been selected and the sensory information will have been optimised. Therefore, the experiment will encompass all that has been learnt from previous research and experimentation and will involve a fully configured task box and test-bed.

The options to be examined in the experiment will be:

(a) Visual feedback only. Only the remote cameras and lighting will be used.

(b) Visual feedback with force-torque display. In addition to the remote cameras, the subject will be provided with a graphics display of the forces and torques being exerted at the end-effector.

(c) Visual feedback with a proximity display. In this condition the subject will be provided with remote camera views plus the view from a proximity camera mounted on the wrist of the robot. (See Figure 3)

These conditions will be carried out under the three time-delay conditions mentioned earlier and a further condition of obscured/unobscured view of the task box. The obscured view will be achieved by degrading the image of one or more of the remote cameras, thereby forcing the subject to use one camera view more than another.

As in the previous experiment, performance measures of whole and part-task timings and accuracy will be taken.

4. Overview of Requirements for the Flexible Teleoperation Facility

From the outset of the study, it was a major priority that the design of the flexible teleoperation facility should be driven by the design of the experimental investigations to be carried out on it. Secondly, the design of the facility should be flexible such that future extensions and developments could be effectively carried out by ESTEC. This flexibility should permit development of the test-bed to such a level that it can be used ultimately as a development or training simulator, and possibly even as a back-up system to an operational ground-based Robotics Telemanipulation and Servicing (RTS) workstation.

Test-bed Requirements for Human Factors Experimentation

Once the experiments were designed and the general requirements for the modular components of the test-bed were broadly defined, it was appropriate to define these more rigidly:

- The robot system chosen for the test-bed is the ASEA IRb6. This choice was made for the sake of convenience as the test-bed will ultimately be based at ESTEC where it will be used with an ASEA IRb60 already situated there. More importantly, the robot should be available with six degrees of freedom. This requirement was made to ensure that operators would develop strategies that would be representative of strategies used by operators using a fully dextrous six degrees of freedom manipulator in LEO. Control of the degrees of freedom must be available in both control of individual degrees of freedom and of control of all six degrees simultaneously.

- The subject's control and display station was designed following ergonomic principles and to allow quick and efficient interfacing of the various joystick configurations. The displays and controls for the remote camera systems were also mounted on this workstation, again following ergonomic principles.

- With regard to the video system, this was designed to provide both worksite/task feedback and subject surveillance. With regard to the former, the following criteria were adhered to in order to ensure the satisfaction of human factors issues raised in Sheppard et al (1986). The system shall:

  (a) Provide detailed and close-up views of the end-effector/task box interface for inspection and fine control.

  (b) Provide global worksite and manipulator views to enhance the operator's notion of the orientation of the remote equipment relative to the worksite.

  (c) Permit the efficient change of the remote views by the operator, both in terms of display quality and content.

  (d) Permit symbolic or graphical representations of remote sensory data to be optimally displayed to the human operator, in a way which does not conflict with his perception of the video imagery.

5. Summary

This paper has described the development of a flexible (modular) teleoperation testbed, based around a commercially available ASEA industrial robot. The intention of the study to be carried out on the test-bed is to investigate fundamental human factors issues concerned with the control and sensory feedback problems of a remotely operated robot.

This paper discussed the experiments and investigations to be carried out on the test-bed and the philosophy behind the development of the test-bed using the design of the experiments as the principal driver. The components of the test-bed are described in detail along with the need for modularisation in order to maintain the degree of flexibility required of the test-bed to ensure efficient and trouble-free development in the future.

Acknowledgements

I wish to thank Mr. Laird Evans of Sowerby Research Centre, British Aerospace, for his invaluable help and support in the preparation of this paper.

References

Milgram, P. et al. "ESA Contract 5594/83: Control Loops with Human Operators in Space Operations, Parts I-V." National Aerospace Laboratory NLR (NLR TR 84 16L) (1983).

Sheppard, J. et al. "Teleoperation and Control Study: Final Report." BAe Space and Communications Division, Stevenage, Report TP8268 (May 1986).

Stone, R. et al. "AERE Contract H2C 547430 ES: Human Factors Evaluation of the MA-23M Servomanipulator." BAe Report No. JS10561 (March 1986).

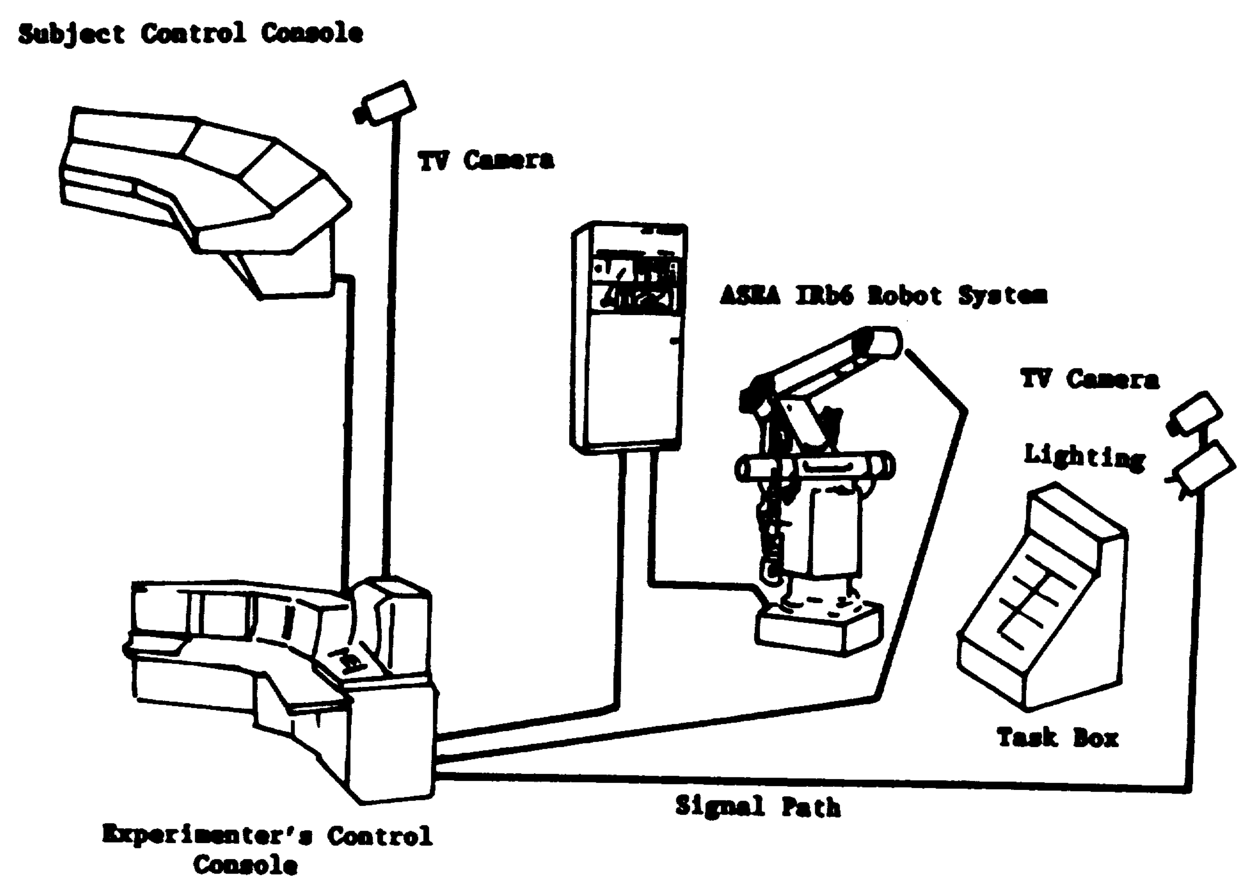


Figure 1: Schematic of Experimental Equipment

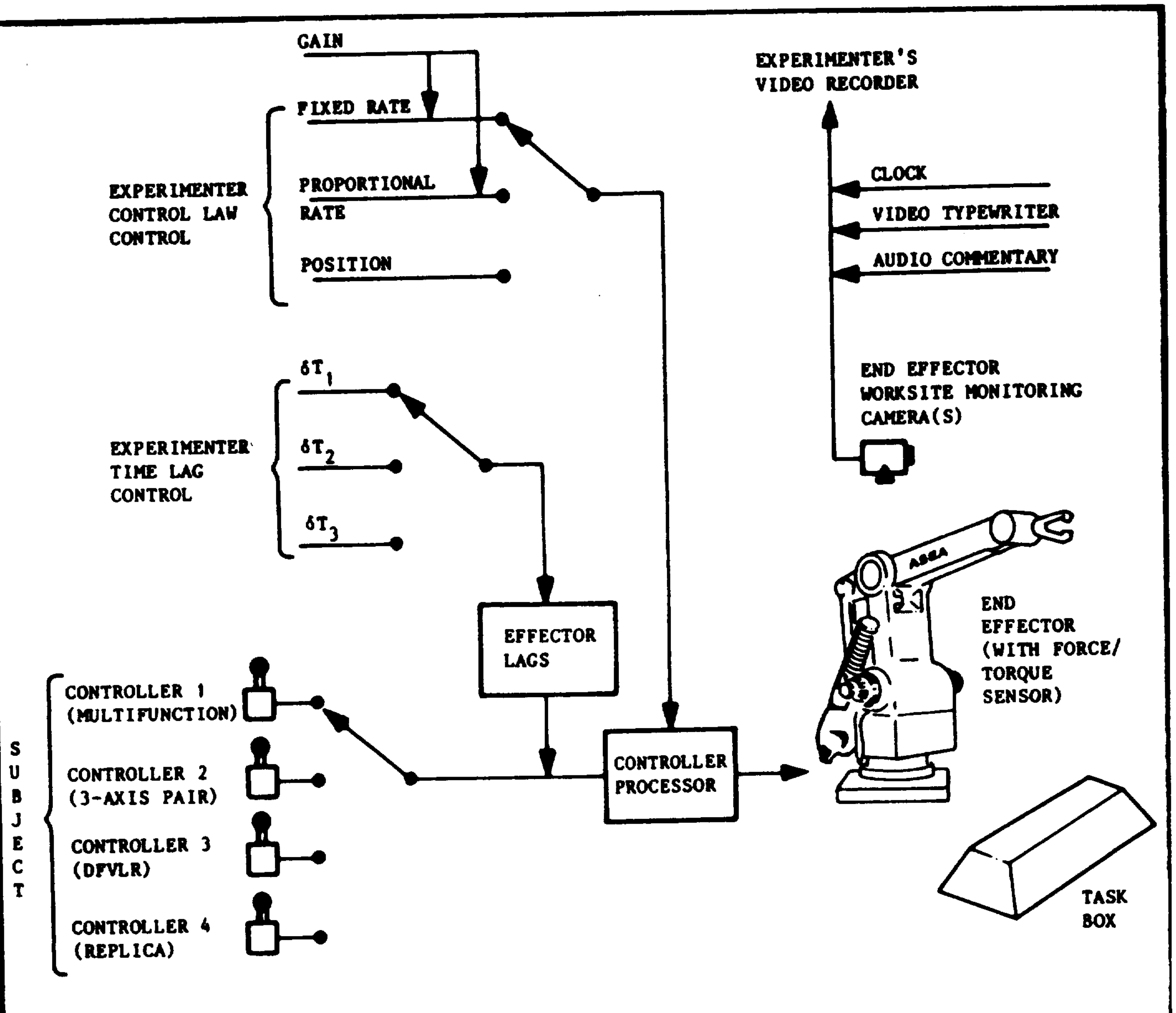

GAIN
FIXED RATE
EXPERIMENTER CONTROL LAW CONTROL
PROPORTIONAL RATE
POSITION
EXPERIMENTER'S VIDEO RECORDER
CLOCK
VIDEO TYPEWRITER
AUDIO COMMENTARY
END EFFECTOR WORKSITE MONITORING CAMERA(S)
δT1
EXPERIMENTER TIME LAG CONTROL
δT2
δT3
ASEA
END EFFECTOR (WITH FORCE/ TORQUE SENSOR)
EFFECTOR LAGS
SUBJECT
CONTROLLER 1 (MULTIFUNCTION)
CONTROLLER 2 (3-AXIS PAIR)
CONTROLLER 3 (DFVLR)
CONTROLLER 4 (REPLICA)
CONTROLLER PROCESSOR
TASK BOX

Figure 2: Schematic of Experiment 1

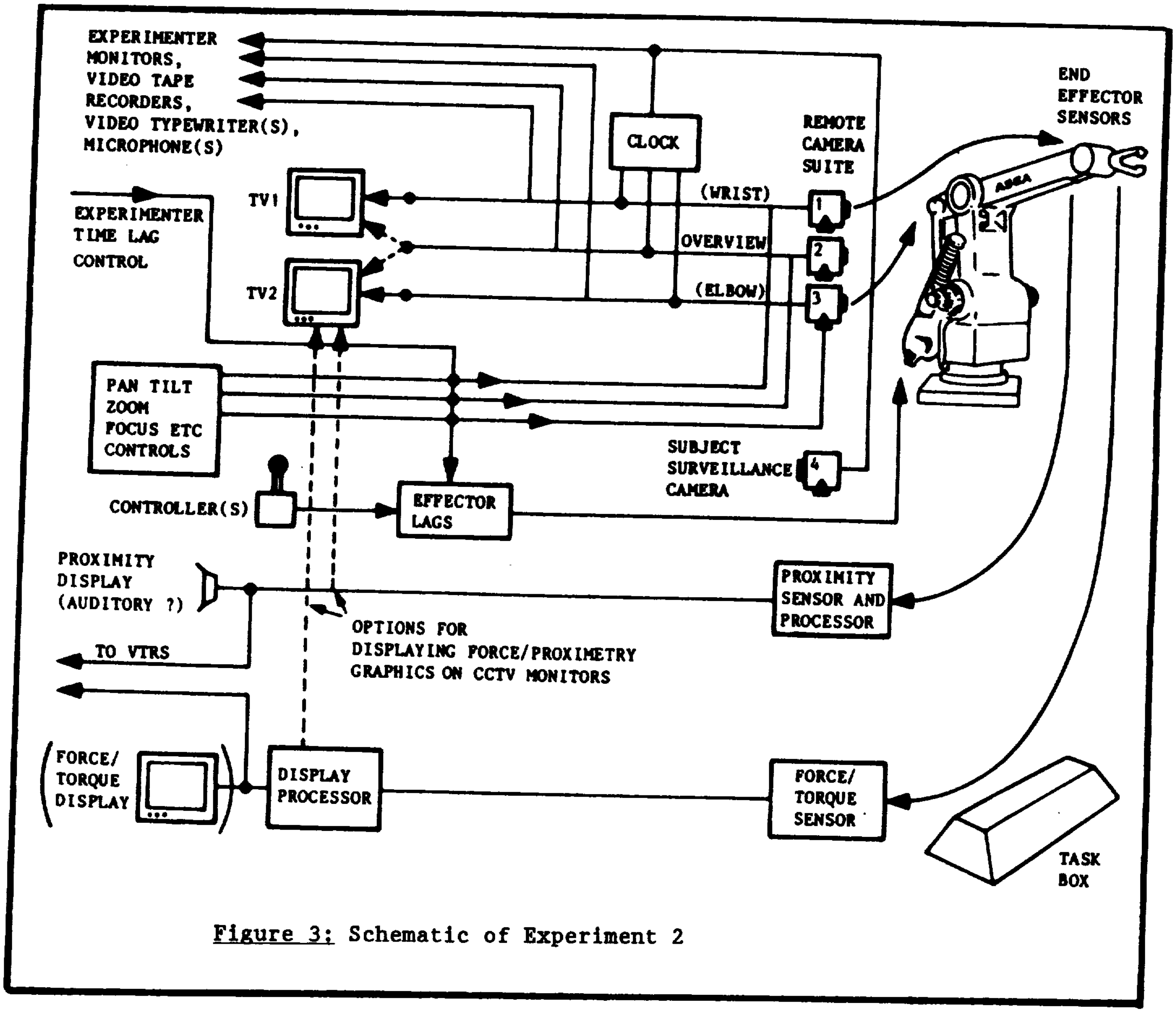

EXPERIMENTER MONITORS, VIDEO TAPE RECORDERS, VIDEO TYPEWRITER(S), MICROPHONE(S)
CLOCK
REMOTE CAMERA SUITE
END EFFECTOR SENSORS
TV1
TV2
EXPERIMENTER TIME LAG CONTROL
(WRIST)
OVERVIEW
(ELBOW)
1
2
3
PAN TILT ZOOM FOCUS ETC CONTROLS
SUBJECT SURVEILLANCE CAMERA
4
CONTROLLER(S)
EFFECTOR LAGS
PROXIMITY DISPLAY (AUDITORY ?)
PROXIMITY SENSOR AND PROCESSOR
OPTIONS FOR DISPLAYING FORCE/PROXIMETRY GRAPHICS ON CCTV MONITORS
TO VTRS
FORCE/ TORQUE DISPLAY
DISPLAY PROCESSOR
FORCE/ TORQUE SENSOR
TASK BOX

Figure 3: Schematic of Experiment 2

TELEROBOT ARCHITECTURES

CONTROL OF INTELLIGENT ROBOTS IN SPACE*

E. Freund, Ch. Bühler

Institute of Robotics Research
University of Dortmund
4600 Dortmund 50, West Germany

Abstract

In view of space activities like International Space Station, Man–Tended–Free–Flyer (MTFF) and free flying platforms, the development of intelligent robotic systems is gaining increasing importance. The range of applications that have to be performed by robotic systems in space includes e. g. the execution of experiments in space laboratories, the service and maintenance of satellites and flying platforms, the support of automatic production processes or the assembly of large network structures. Some of these tasks will require the development of bi–armed or of multiple robotic systems including functional redundancy.

For the development of robotic systems which are able to perform this variety of tasks a hierarchically structured modular concept of automation is required. This concept is characterized by high flexibility as well as by automatic specialization to the particular sequence of tasks that have to be performed. On the other hand it has to be designed such that the human operator can influence or guide the system on different levels of control supervision, and decision. This leads to requirements for the hardware and software concept which permit a range of application of the robotic systems from telemanipulation to autonomous operation. The realization of this goal requires strong efforts in the development of new methods, software and hardware concepts, and the integration into an automation concept.

1. Introduction

With respect to increasing space activities, e. g. ISS, free–flying platforms or planetary operations, it is necessary to reduce the operational costs for space systems. One major key for future operational systems will therefore be the application of robotic systems in space /1/. It is planned to use different kinds of manipulators and robots in space to support and execute several tasks inside space-modules or in free space, especially for

- Docking/Berthing,
- Repair and module exchange,
- Service and maintenance of free–flying platforms,
- ORU (Orbit Replaceable Unit) – Exchange,
- Assembly of large structures,
- Experiment execution and production tasks.

* This work was supported by a grant of the "Bundesminister für Forschung und Technologie (BMFT)" of Federal Republic Germany.

The execution of this variety of tasks for manipulators and robots in space requires a hierarchically structured modular concept of automation including as well telemanipulation as autonomous operation. This design should cover the range of possible applications and has to provide interfaces for human interaction on different levels of control, supervision, and decision. To reach this goal, intensified efforts in the development of future–oriented robotic systems are necessary, where the integration into an overall concept of automation should be included from the very beginning.

In this paper structures for the control of multi–robot systems for space applications are considered. In chapter II the general structure for an autonomous multi–robot system in space is introduced. One of the key issues of the concept is automatic task management which is discussed in section III. The structuring of the levels of coordinated operation and collision avoidance and a mathematical formulation of the substructures is presented in chapter IV. Finally a test facility for the proposed intelligent control structure and for specific control features, which was built up at IRF Laboratory in connection with the national project CIROS (<u>C</u>ontrol of <u>I</u>ntelligent <u>R</u>obots in <u>S</u>pace), is described in section V.

II. Overall System

In present space technology astronauts have to leave the space vehicle for almost every execution of activities outside this vehicle. Robots and manipulators could be used in space to reduce the risk and cost of such "Extra Vehicular Activities", where robots and telemanipulators will evolute step by step from simple telemanipulation robots to autonomous robotic systems /2/.

Teleoperation is the first step in this development. Here a human operator controls the manipulator by means of a model or from a control panel. The structure of a manipulator system for teleoperation is shown in fig. 1. The first level of the system representation is the mechanical construction of the manipulator, which includes internal sensors for the measurement of positions, velocities, forces, and torques.

The dynamic model of the robot can be described by highly nonlinear differential equations

$$\underline{x}(t) = \underline{A}(\underline{x}) + \underline{B}(\underline{x})\ u(t), \qquad \underline{y}(t) = \underline{C}(\underline{x}) \tag{1}$$

with nonlinear couplings between the variables of motion. In eq. (1), $\underline{x}(t)$ is the n dimensional state vector, $\underline{u}(t)$, $\underline{y}(t)$ are the m dimensional input– and output–vectors, respectively, and $\underline{A}(\underline{x})$, $\underline{B}(\underline{x})$, $\underline{C}(\underline{x})$ are matrices of compatible order, which describe the dynamics of the system. By use of the nonlinear control concept

$$\underline{u}(t) = \underline{F}(\underline{x}) + \underline{G}(\underline{x})\,\underline{w}(t) \tag{2}$$

with

$$\underline{F}(\underline{x}) = -\,\underline{D}^{*-1}(\underline{x})\left[\,\underline{C}^{*}(\underline{x}) + \underline{M}^{*}(\underline{x})\right], \quad \underline{G}(\underline{x}) = -\,\underline{D}^{*-1}(\underline{x})\,\underline{\lambda},$$

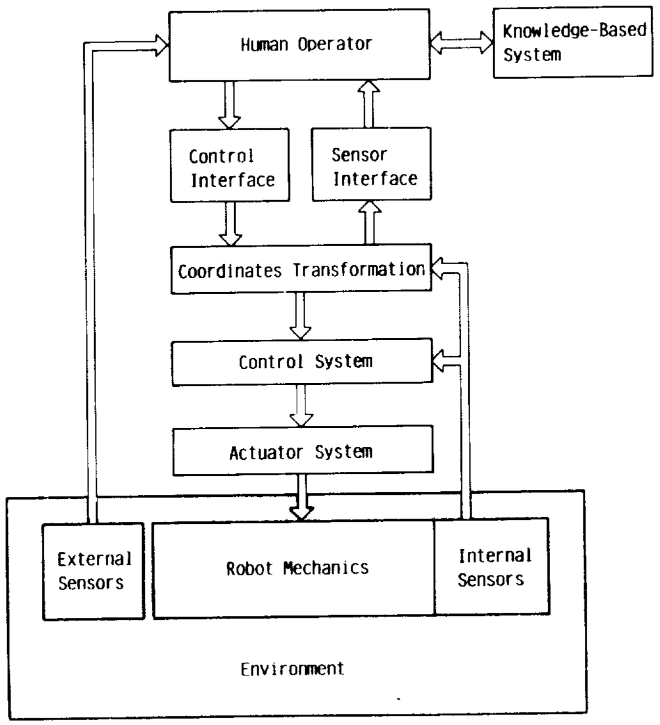


Figure 1. Control structure of a manipulator system for teleoperation

where the matrices $\underline{D}^*$, $\underline{C}^*$ and $\underline{\lambda}$ are given from the nonlinear decoupling and control law /3/, one obtains a linear, decoupled behaviour for each axis:

$$\ddot{y}_i(t) + a_{i1}\,\dot{y}_i(t) + a_{i0}\,y_i(t) = \lambda_i \cdot w(t), \qquad (i = 1,\ldots,m) \qquad (3)$$

The level of controllers computes signals for actuation which are transformed by the actuator system into forces and torques acting on the mechanical construction /3/.

At this stage man is still in the loop, responsible for motion planning, coordination, collision avoidance and supervision. The operator may get help information from a knowledge–based system, where system data can be stored and later on recalled for similar tasks. During teleoperation from ground time delays of up to 5 sec decrease the performance. Therefore it is necessary to provide the manipulator or robotic system even at this stage of development with a certain level of autonomy.

Further steps towards autonomous robotic systems are telesupervision and teleautomation, where simple tasks are accomplished automatically, while the operator controls the system as telemanipulator for specific tasks. So the operator is not required to be permanently in the loop, which eases the job of the crew considerably. Due to the variety of the tasks, possible long duration missions and from aspects of reliability of the system the highest degree of automation is desirable. The last stage in this development is a fully autonomous operating multi–robot system, whose structure is given in fig. 2. The system consists of several robots with one or more arms each. Based on the structure of a manipulator system in fig. 1, the evolution to autonomous multi–robot systems is characterized by the integration of additional hierarchical levels. Strategies for automatic task management, coordinated operation and collision avoidance became integral parts of the structure.

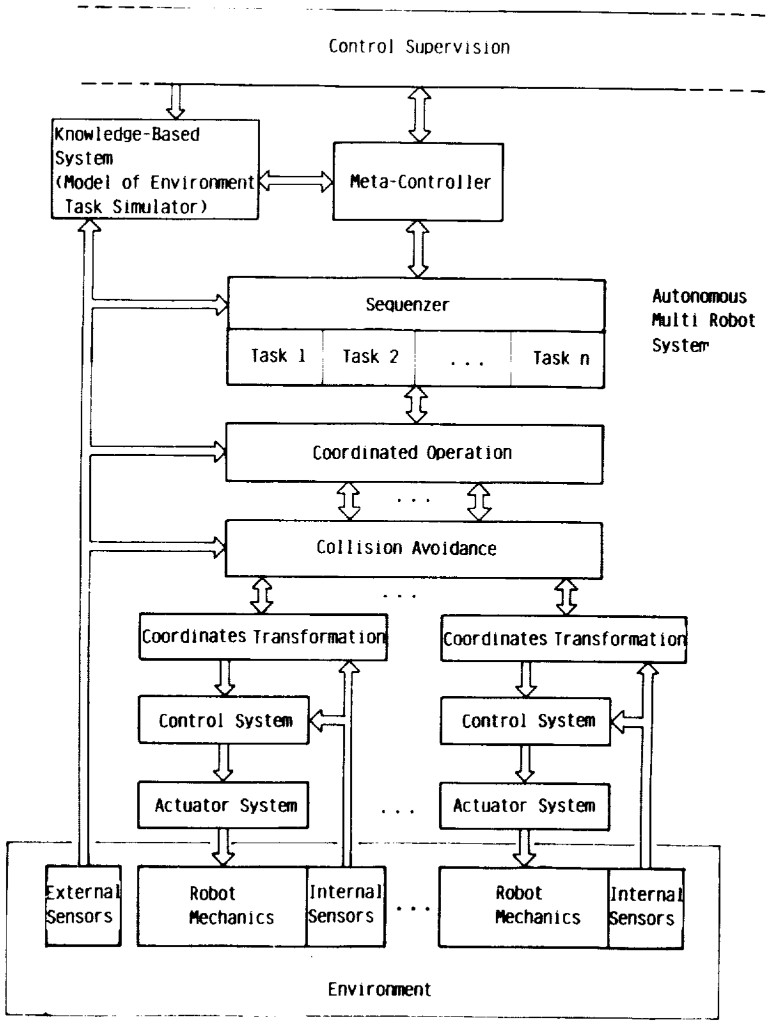


Figure 2. Structure of an autonomous multi–robot system

The levels for collision avoidance and coordinated operation are hierarchically placed above the level of coordinate transformation, which includes the various kinematics of the robots involved. The level of coordinated operation is responsible for the generation of reference values, which enable a coordinated task execution. To avoid collisions of the robots with themselves as well as with obstacles, in the layer of collision avoidance appropriate strategies are implemented. The formulation of these strategies is based on a systematic design procedure for multi–robot systems /4/, which has to be applied on a group and a system level.

The superimposed task management executive is responsible for automatic task activation choice and reservation of appropriate robots, execution control and performance control. Also included are safety and emergency reactions, which are initiated in case of failure and contingency. The operation is assisted by a knowledge–based system, which runs a model of environment as well as a task simulator, whereas the model of environment will be continuously updated by evaluation of relevant sensor data and status reports. The task simulator checks out descriptions of new tasks, which are passed through the control supervision.

After a successful testing out, the resulting executable tasks will be transferred by the Meta–Controller to the sequencer for storage in a dedicated task memory, from which the program can be executed on demand.

In this system, the task of the human operator reduces to initiation, supervision and acknowledgement of completed robot tasks by means of the control supervision unit. Nevertheless, the operator is always able to take control over the system, especially in case of failure or emergency situations.

III. Automatic Task Management

In a large robot system, e. g. in a scenario of space station including robots with various capabilities, e. g. mobility, multi–armed systems, different working modes and on the other hand a broad range of very different tasks automatic task management is one of the key issues. Due to the complexity of the system itself and of the range of applications the problem must be solved on an abstract level far beyond the level of move commands of commercial robot languages. Situated between the Meta–Controller as an interface to man and the system coordinator (CoS) as the intelligent interface to the single robots the automatic task management has to provide a lot of functions performing system control and the break down from the highest level of abstraction to a middle level at the robot side. At the input level complex tasks are described, which include implicitely the use of a group of appropriate robots for execution. The whole work of each task can be subdivided in parts, which can be performed sequentially or in parallel according to the special needs of the task. The system coordinator accepts coordination primitives, which address groups of robots on a multi–robot movement level. The break down to collision free move commands for single robots is performed at the levels of coordinated operation and collision avoidance. The structure of the levels mentioned is shown in principle in fig. 3 from the task input to the output to the robots.

The tasks are transferred for execution from the Meta–Controller with a task specific global priority. The task management activates the task with the highest priority, if the capabilities of the multi–robot system match the needs of the task. At this level groups of robots according to the task are defined but the robots are not yet booked. Also the group coordinators (CoG) are configured and the group collision avoidances (CAG) are initiallized.

The choice and requisition as well as the derequisition of the robots with the appropriate performance capabilities is executed on the subtask level, where the subtasks provide a list of performance attributes. The subtasks are initiated according to a set of rules, which consider priorities, the logic flow of execution, time critical paths and the availability of appropriate robot systems. The actual priority of a subtask is computed based on the global task priority, the attributes of the subtask and the current system status. For these reasons the system is event driven and the complete execution sequence is in general not known in advance. Each subtask contains a number of coordination primitives, which are transferred to the system coordinator sequentially.These sequences are only broken in case of failure or emergency, while in ordinary operation the prescribed sequence of coordination primitives of the subtask is executed.

IV. Coordinated Operation and Collision Avoidance

Due to the complexity of a space robotic system containing e.g. free–flying servicers, OMV's, RMS's, SMS's, etc. robot coordination and collision avoidance have to be considered on two different levels. First on a global system level, which takes a global but rough

ORIGINAL PAGE IS
OF POOR QUALITY

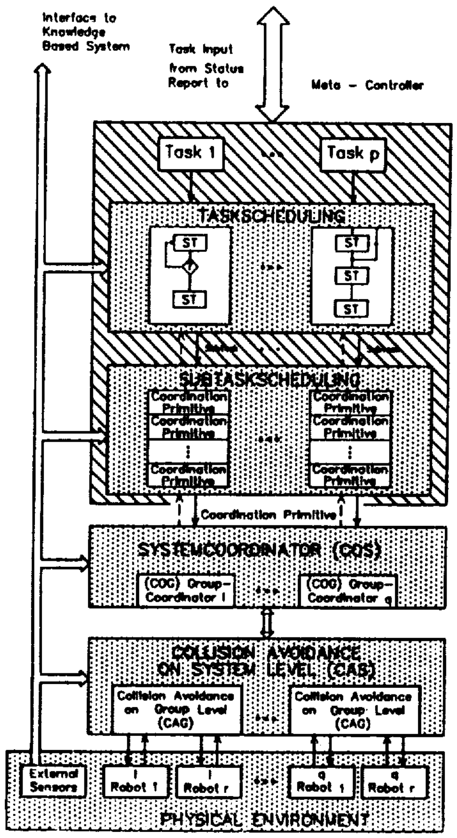


Figure 3. Substructure automatic task management, coordinated operation and collision avoidance

view over all robot systems involved. Second on a group level, which takes a close look at a number of robots working at the same task or subtask. In order to bring out the highest degree of flexibility strong real–time capabilities are required on the group level.

Considering a robot group consisting of r robots, each described by eq. (1) and controlled according to eq. (2), one obtains r sets of equations of form (3) for the closed loop robotic systems. As the robots work in coordinated operation, the reference inputs $\underline{w}_1(t), ..., \underline{w}_r(t)$ have to be coordinated by a hierarchical coordinator of the type

$$\begin{bmatrix} \underline{w}_1(t) \\ \cdot \\ \cdot \\ \cdot \\ \underline{w}_r(t) \end{bmatrix} = \underline{\tilde{H}}_G\,(\underline{x}_1, ..., \underline{x}_r; \underline{v}_1, ..., \underline{v}_r), \tag{4}$$

where $\underline{v}_1(t), ..., \underline{v}_r(t)$ symbolize the move commands in coordination space for the nonlinear controlled robots.

Applying the nonlinear control scheme eq. (2) to the individual robot arms it is /5/

$$\begin{bmatrix} \dot{\underline{x}}_1(t) \\ \cdot \\ \cdot \\ \cdot \\ \dot{\underline{x}}_r(t) \end{bmatrix} = \begin{bmatrix} \underline{A}_1^* & \cdot & \cdot & \cdot & \underline{0} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ \underline{0} & \cdot & \cdot & \cdot & \underline{A}_r^* \end{bmatrix} \begin{bmatrix} \underline{x}_1(t) \\ \cdot \\ \cdot \\ \cdot \\ \underline{x}_r(t) \end{bmatrix} + \underline{B}^* \cdot \tilde{\underline{H}}_G (\underline{x}_1, \ldots, \underline{x}_r; \underline{v}_1, \ldots, \underline{v}_r). \qquad (5)$$

This formulation contains the dynamics $\underline{A}_i^*$ (i = 1, ..., r) of the axes of motion of all r robots and defines a second high level control loop by means of the hierarchical group coordinator $\tilde{\underline{H}}_G (\underline{x}_1, \ldots, \underline{x}_r; \underline{v}_1, \ldots, \underline{v}_r)$.

The hierarchical group coordinator (CoG) eq. (5) is structured as follows eq. (6):

$$\tilde{\underline{H}}_G (\underline{x}_1, \ldots, \underline{x}_r; \underline{v}_1, \ldots, \underline{v}_r) = \tilde{\underline{H}}_G^a \begin{bmatrix} \underline{x}_1(t) \\ \cdot \\ \cdot \\ \cdot \\ \underline{x}_r(t) \end{bmatrix} + \tilde{\underline{H}}_G^b (\underline{x}_1, \ldots, \underline{x}_r) + \tilde{\underline{E}}_G \begin{bmatrix} \underline{v}_1(t) \\ \cdot \\ \cdot \\ \cdot \\ \underline{v}_r(t) \end{bmatrix} \qquad (6)$$

Equation (6) is a formulation in joint space of the robots, which shows the applicability of this method. The dynamics of the links of the robots can be arbitrarily chosen by appropriate selection of the matrix $\underline{H}^a$. Useful couplings between links of different robot can be introduced by $\underline{H}^b$. The matrix $\underline{E}$ contains gains for input vectors in coordination space /4/. The method of nonlinear decoupling offers a useful opportunity to choose the dynamics of the robotic system not only in joint space but in other e.g. task or group oriented coordinate systems as well.

With an appropriate definition of the task variables the different modes of coordinated operation e.g. synchronisation, docking and cooperated payload handling can be implemented, while the structure of the coordinator remains the same /7/. Each group coordinator independently works on one coordination primitive and generates the input commands for the robots involved. The hierarchical system coordinator initializes the group coordinators allocates the coordination primitives and supervises the execution on a middle level, reporting the system state to the units above. In case of a collision danger between single robots of different groups the system coordinator works close together with the system collision avoidance searching collision free but still group coordinated paths. This can be accomplished by priority considerations or a reconfiguration of the robot groups.

The level of collision avoidance is responsible for a collision free operation of the whole system. This includes at first the realtime detection of collision danger between robots working in the very same group as well as between robots of different groups. Only in case of collision danger this module intervenes whereas in case of no danger of collision the original reference inputs are applied to the robots. The level of collision avoidance is split in two parts: collision avoidance on group level (CAG) i.e. between robots of the very same group and on system level (CAS) i.e. between robots of different groups. The group collision avoidance has to distinguish three major operation modes: independent action, synchronised actions and fully cooperated motion of the robots involved. It is obvious that the avoidance strategy is closely dependent on the mode of operation, but in each case the strategy can be described by a similar structure as the group coordinator eq. (6):

$$\underline{H}_G = \underline{H}_G^a(\underline{H}_G^b) \cdot \underline{x}_I + \underline{H}_G^b(\underline{x}_I, \underline{v}_I) + \underline{E}_G(\underline{H}^b) \cdot \underline{v}_I \tag{7}$$

In this formulation the states and inputs of all robots have been noted in a condensed form

$$\underline{x}_I^T = (\underline{x}_1, \ldots, \underline{x}_{rI}) \ , \ \ \underline{v}_I^T = (\underline{v}_1, \ldots, \underline{v}_{rI}) \tag{8}$$

For the collision avoidance between robots of the very same group $\underline{H}_G^b$ $(\underline{x}_I)$ in eq. (7) is the essential part of the structure. The detection of danger of collision as well as the collision avoidance strategy are based on the calculation of the collision avoidance trajectories $\underline{f}_j$, (j = 1, ..., r). These trajectories are determined by on–line prediction of the robot movement, regarding the current robot states the preprogrammed paths and the task oriented right–of–way priorities of the robots /6/. These trajectories are described in the elements of the matrix $\underline{H}_G^b$ including the information if currently collision danger was detected. The matrices $\underline{H}_G^a$ permits the change of the control dynamics in dependence on the level of danger of collision and the original inputs $\underline{v}_I$ are cancelled by means of $\underline{E}_G$ in case of a predicted collision.

Considering now a multirobot system consisting of N groups of r_i (i = 1, ..., N) robots each, where all robots are feedback controlled according to eq. (2) and each group equipped with a CAG–unit. This formulation leads to N robot groups, which are completely decoupled where the single robots of the very same group are in coordinated operation by means of CoG and under online collision avoidance by means of CAG. The case of collision avoidance between robots of different groups is implicitely included by the demand for collision free paths $\underline{v}_i$ of robots of different groups, as inputs to CAG from the CAS unit.

The separation of collision avoidance on group and system level, respectively takes use of the fact, that mostly collision danger occurs inside a robot group, where the robots work close together. Here the group dedicated CAG provides a fast response in case of detected collision danger inside the group, taking the working mode of the robots into consideration. In case of collision danger between robots of different groups, which occurs less frequently e.g. with mobile robot systems but is of the same importance with respect to possible damage of the systems, the CAS intervenes. It generates a vector $\underline{e}$ containing the draw back directions, which decrease the danger of collision the most. Based on this information the CoS and CoG react, where the possible constraints of coordinated operation are still kept. If conflicts or deadlock situations occur the strongest action is the reconfiguration of the groups and the CoG's taking the conflicting systems in the very same group. Otherwise evasive actions and priority increase of the conflicting robots inside the group can solve the problem.

V. CIROS test facility

To provide a realistic environment for development and test of the modules of the proposed hierarchical control structure for robots in space an appropriate facility was built up in the IRF laboratory, which is part of a national space project called CIROS (Control of Intelligent Robots in Space). Based on the modular concept it is possible to develop and implement methods and strategies based on terrestrial robots, because for the transfer to a real space environment only the robots themselves and the low level control up to the coordinates transformation have to be changed. The upper levels of the control structure however remain the same with some minor adaptions. As test scenario an unmanned space laboratory (e. g. Spacelab, MTFF) was chosen, where typically experiment service repair tasks or experiment exchange are to be performed. The test facility is completed by a control and supervision board which could be integrated in a manned space station or in a ground based control center.

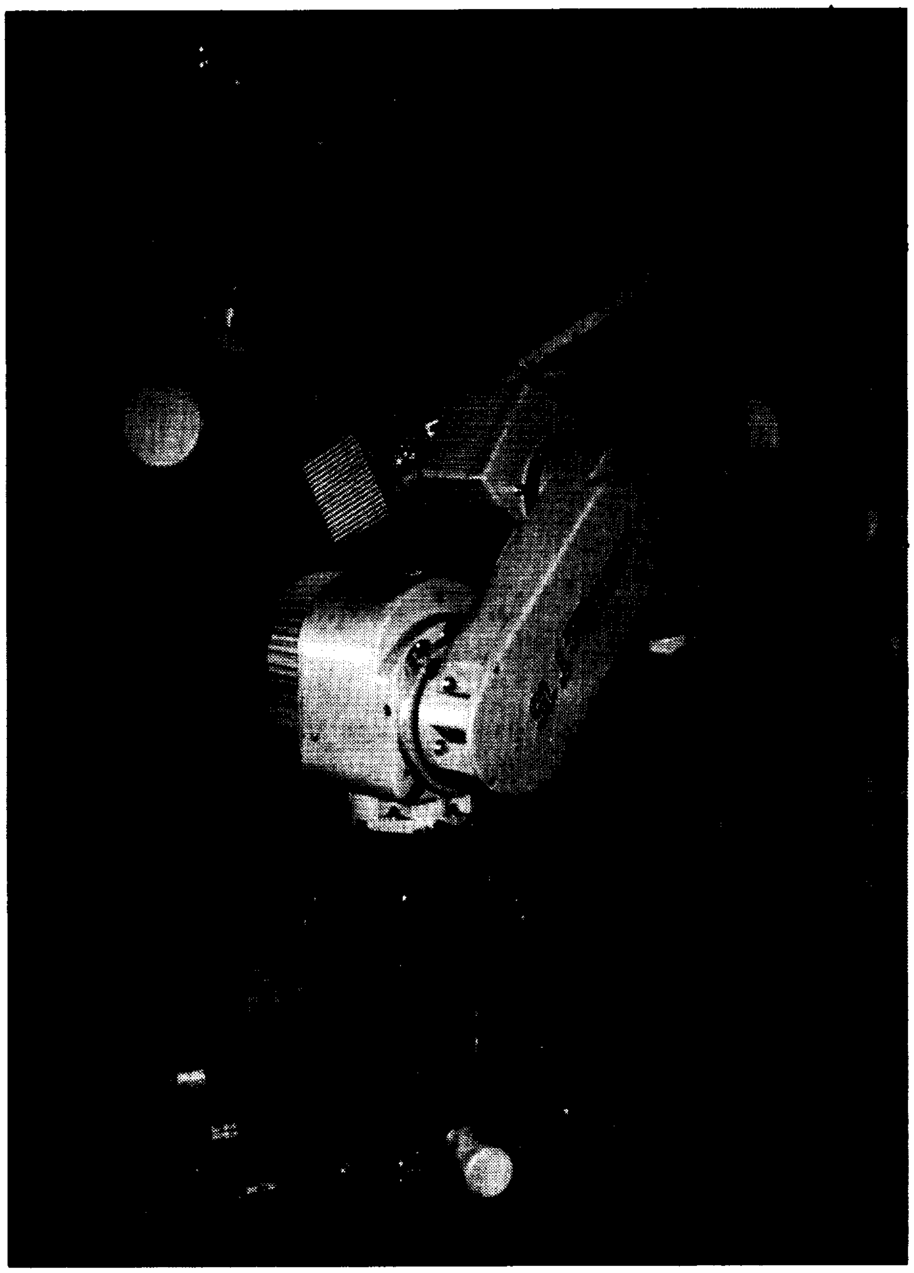


Figure 4. Robots with common working space in the CIROS test facility

This environment provides the test facilities for all the upper levels of the control structure including different grades of automation. In order to study the problems in multi–robot systems for space applications two robots with widely overlapping working spaces are integrated. Both robots are equipped with tool exchange capabilities and additional sensors e.g. force–torque sensor, arm–mounted camera, proximity sensors ect. So each robot is able to perform every task in the system. Therefore it is possible to consider the automatic task management as well as coordinated operation and collision avoidance in a realistic environment. The hierarchical control is implemented on a real–time computer, which is interconnected with a knowledge based system and the control supervision, where time delay can be simulated between the different computers. The control board contains several input/output devices like alpha numeric terminals, graphics, video sensor ball, ect. From this board the system is supervised, a runtime documentation is done and interventions of the human operator are accepted. Additionally it is used as development facility and an off–line programming system is integrated as well as a cell simulation. A picture of the two used robots working on a rack is shown in fig. 4. In this cell, which was designed similar to a spacelab environment, the main functions of the hierarchical control structure can be implemented and demonstrated exemplarily. Especially the problems and capabilities of multi–robot systems for space applications can be studied.

VI. Conclusion

In this paper a hierarchical structure for the control of multi robot systems for space applications is presented. The break down from a high level of abstraction at task management level down to the single robot control is described step by step. The splitting in a consideration on system and on group level takes the distributed character of a large space system into account. As a possible space scenario for A&R an unmanned space station is focussed and introduced as development and test environment at the IRF–Laboratory. Based on this facility A&R with multi–robot systems can be studied at IRF in practical examples.

References

/1/ Orientierungsrahmen Hochtechnologie Raumfahrt (OHR), Gesamtüberblick und Empfehlungen, Hrsg. DFVLR, Köln, 1987.

/2/ Freund, E.: "Hierarchically Structured Control for Intelligent Robots in Space", Proceedings Intern. Symposium on Europe in Space – The Manned Space System –, Strasbourg – France, 1988

/3/ Freund, E.: "Fast Nonlinear Control with Arbitrary Pole–Placement for Industrial Robots and Manipulators", International Journal of Robotics Research, Vol. 1, No. 1, 1982, pp. 65 – 78

/4/ Freund, E.: "On the Design of Multi–Robot Systems", Proceedings IEEE International Conference on Robotics, Atlanta, Georgia, 1984

/5/ Freund, E.; Hoyer, H.: "Collision Avoidance for Industrial Robots with Arbitrary Motions", Journal of Robotic Systems, Vol. 1, No. 4, 1984, pp. 317 – 329

/6/ Freund, E.; Hoyer, H.: "Real–Time Pathfinding in Multi–Robot Systems Including Obstacle Avoidance", International Journal of Robotics Research, Vol. 7, No. 1, 1988, pp. 42 – 70

/7/ Freund, E.; Kaever, P.: "Autonomous Mobile Robots", IFAC–Symposium on Robot Control '88 SYROCO 88, Karlsruhe – FRG, 1988.

N90-29014

MODULARITY IN ROBOTIC SYSTEMS

by
Delbert Tesar, Carol Cockrell Curran Chair in Engineering

Michael S. Butler, Research Assistant

Mechanical Engineering Department
THE UNIVERSITY OF TEXAS AT AUSTIN

INTRODUCTION

Most robotic systems today are designed one at a time, at a high cost of time and money. This wasteful approach has been necessary because the industry has not established a foundation for the continued evolution of intelligent machines. The next generation of robots will have to be generic, versatile machines capable of absorbing new technology rapidly and economically. This approach is demonstrated in the success of the personal computer, which can be upgraded or expanded with new software and hardware at virtually every level.

Modularity is perceived as a major opportunity to reduce the 6 to 7 year design cycle time now required for new robotic manipulators, greatly increasing the breadth and speed of diffusion of robotic systems in manufacturing. This paper focuses on modularity and its crucial role in the next generation of intelligent machines. It begins by examining the main advantages that modularity provides; the second part of the paper discusses the types of modules needed to create a generic robot. The final section examines some structural modules designed by the robotics group at the University of Texas at Austin to demonstrate the advantages of modular design.

THE ADVANTAGES OF MODULARITY IN ROBOTICS

Modularity can be approached in almost all components involved in machine design - computer software and hardware, sensors, actuators, man-machine interface, etc. The advantages of modularity are readily illustrated by examining its impact in the areas of portability, precision, reliability, and economics.

PORTABILITY

Portability of a robotic system implies that it can be broken down into pieces (or modules) small enough to be carried to the work place by a human operator and quickly assembled. A truly portable robotic system has many possible applications, such as:

- **nuclear reactor maintenance** (especially for use in different areas of a plant or in two different plants);

- **explosive ordnance disposal (EOD)**, in which a military or police unit could easily and rapidly transport a robot to a new location to defuse or detonate explosives;

- **space operations**, where small, lightweight modules would help meet critical size and weight restrictions.

Each module would have to be carefully designed to be lightweight and durable. The suggested weight limit per module is 35 lbs. Such a weight restriction creates an unusual demand to design light weight actuators and to use special light weight materials (composites or carbon fiber).

PRECISION

The absolute precision of most industrial robots is known to be not better than 0.05 inch, and many are far less accurate. Yet many assembly, welding and light machining operations require a precision of 0.01 inch. Further, fine positioning to 0.001 inch is sometimes necessary.

This level of precision puts an unusually demanding resolution requirement on the actuators and their control system. The control encoders and actuators must be capable of steps of ten seconds of angular rotation. Most actuators fall far short of this, especially if they must provide a high load capacity.

In addition to these precision requirements, the more difficult condition is to maintain precision while the manipulator experiences large load variations. It is common for external loads to degrade the unloaded precision by a factor of ten. The reader can prove this reality to himself by "shaking hands" with a few industrial robots. It is not uncommon to achieve oscillations of 1/4 inch in magnitude. Process disturbances occur from such unit processes as cutting, routing, bending, drilling, force fit assembly, etc.

One possible approach involves intelligent adaptation to system parameter variations. Industrial robots do not exhibit perfectly invariant parameters within the complex control and structural subsystems. The sources of the parametric variations may come from changes in actuator electrical resistance or hydraulic fluid properties, friction in joints, dimensional changes due to temperature fluctuations, and other inconsistent effects. Implicit variations may also be due to imperfect numerical values used in the deterministic model.

The objective is to characterize these parametric variations and to develop a self-organizing adaptive system to compensate with respect to the nominal deterministic model. This "electronically constant" computational scheme could be packaged as software either as hardened design specific ROM or in generic software packages.

Motion command shocks which occur during starting and stopping actions of robotic manipulators induce large oscillations detrimental to precision motion. These shocks represent discontinuous derivatives in the motion program - a concept long recognized in the dynamic programming of precision cam systems.[ref] Researchers at U.T. have been heavily involved with dynamic motion programming of precision high speed cam systems. This work provides a broad analytical scheme which can be applied to 6 DOF spatial motion. Again, software packaging could allow the use of the formulations in a generic or product specific sense.

Precision under loading may be accomplished through an interface of software and hardware component technologies. An "electronically rigid" manipulator could be created by a combination of real time dynamic modelling and distributed joint control.

The dynamic model formulation required is the same as mentioned previously. Using known displacement, velocity, and acceleration information for the motion of the manipulator combined with physical properties of the structure and prime movers and anticipated or measured external loads will allow prediction of nominal deformations. These nominal deformations can then be eliminated through adjustments in control commands in real time. The complexity of the formulation and the required computational speed again imply the necessity of software and computer modules.

There are two pressing problems with small scale motion compensation or "control in the small" with software alone:

- Major actuators required to resist large loads through large motion ranges are simultaneously incapable of meeting high resolution requirements.

- Real time control in large motion ranges can be accomplished adequately if the computational sampling time is modest. Such limited system update rates makes the simultaneous maintenance of high precision operation unlikely.

The best future technology capable of meeting these needs is the dual operation of a manipulator in the large and in the small. Physically, this layering can be accomplished through a high load capacity actuator and a small precision actuator. Computationally, a low speed sampling rate for the full non-linear large motion system and a high speed sampling rate for the linearized small motion system can be achieved.

RELIABILITY

Industrial robots today have established a very high operating availability of approximately 98%. These were marketed only after prolonged testing and redesign. Nonetheless, in other unique applications, this extensive history is not available to ensure high reliability. This property is especially important in remote, hazardous operations like nuclear reactor maintenance and space operations, where a failure would result in huge losses in time and money. Failure is also unacceptable where human life is involved, as in accident missions, military operations or ocean floor activity.

Failure in a robotic system might mean the high cost of total replacement. This maintenance objective would best be met by using robots made up of modules which could be easily replaced. Redundancy in some of the hardware components (sensors, encoders, local microprocessors, etc.) can be helpful. Unfortunately, the need to be lightweight and compact makes reliability more difficult to achieve.

Self-monitoring software, similar to that being used in advanced computers, would be highly desirable. In this regard, self-calibration of the robot system after maintenance or component replacement would be necessary to maintain the match between the control software and the robot hardware.

ECONOMICS

Obviously, economics is an important architectural issue in mechanical design. Compromises in the choices of materials, computer software and hardware, prime movers, sensors, etc. almost always have to be made in order to meet economic realities. Technological developments in these areas will simplify these economic choices.

The modularity approach will allow aggressive upgrading programs to be pursued in almost all major areas. Actuator modules will allow the user to expand the degrees of freedom of a manipulator or replace damaged or worn actuators inexpensively. Computer hardware can be upgraded and expanded to increase speed and expand capabilities as microprocessor technology improves. Modular software can be inexpensively added to a system to provide dynamic model compensation, metrology for an expanded or adapted manipulator, improved decision making, or integration of improved sensors. An important contribution of the modularity approach is that the system can be expanded or upgraded inexpensively, as opposed to replacing the whole system and starting from scratch.

Advances in materials have allowed manipulators to become stronger and stiffer while decreasing overall size and weight. While some of the newer composite materials are more expensive than conventional alloys, the smaller size and improved capabilities of the manipulator will be worth the cost in many cases.

Many contemporary sensor systems, though rudimentary, are prohibitively expensive. Advanced vision systems, for example, can cost upwards of $200,000. The importance of such systems has fueled a great deal of research and, as with all electronic technologies, advances in sensors will bring better technology at a lower cost. A modular approach, allowing inexpensive software upgrades and hardware replacement, will ease the cost burden of advanced sensor systems.

MODULE TYPES

Specially designed modules with standardized interfaces can be fit together to create a generic, high performance robot. The next generation of robot must be constructed from a large class of near optimum actuator modules which contain their own sub-systems for sensing and (computer) intelligence. Different types of modules can be added to or removed from the robot, depending upon the task at hand.

The modules must be readily scaled (small and large sizes) with standard physical and software interfaces for effortless assembly. Enhanced maintenance due to this modular design is an obvious benefit. This approach is the primary reason that the application of the modular micro-chip is so widespread.

ACTUATOR MODULES

The actuator module concept proposes to combine the joint and prime mover into an interfaceable package. These modules (or building blocks) would be a series of 1, 2, or 3 degree of freedom (DOF) units which could be assembled rapidly by a designer to respond to the requirements of a given application.

Most actuators presently being used in manipulators are off-the-shelf prime movers not specifically designed for precision control of the large coupled motions that occur in robots. This approach does not lead to an optimum balance between the best characteristics of the prime mover and the physical structure of the system. Presently, many actuators are too heavy, have poor response times to commands, generate backlash inaccuracies, have poor resolution, are not stiff under load, and do not contain any local intelligence.

In the design of an actuator, referring to the joint and its associated prime mover, there are four fundamental characteristics which determine the effectiveness with which a manipulator can function. These are strength, stiffness, precision positioning capabilities and component packaging.

The **strength** of an actuator is the measure of the force or torque that it can generate. The load capacity of a manipulator is a direct function of the strength of its prime movers. In a serial chain arrangement, actuators are usually ordered with the strongest component controlling the link closest to ground and subsequent components of decreasing strength. This configuration results in an efficient match of actuator strength requirements with the strength requirements of the hardware components.

The **stiffness** of an actuator is defined in terms of a functional spring rate which is equal to an applied force or torque at the actuator divided by the deflection that results due to the applied load. The stiffness of actuators in a manipulator chain determines the precision with which the manipulator can perform positioning operations under dynamically varying loads. The required actuator stiffness is directly related to the speed and resolution of the control system. The actuators must be rigid enough to hold a prescribed end-effector (to within a given tolerance) under the maximum force variation that would be expected. If this basic requirement is not met, the actuator could be displaced beyond the acceptable positional tolerance before the system could sense and compensate for the error.

In manipulator systems designed to perform tasks for which external loads are small and precision requirements are minimal, such as spray painting, stiffness is not an important design criteria. On the other hand, the stiffness of a manipulator performing such tasks as force fitting and routing is a critical design objective.

Precision positioning capabilities are determined by many factors, including the sensitivity and resolution of control components, friction in the actuator, backlash in the prime mover, and the mechanical integrity of the structure. All contribute to the precision problem the designer faces and increases the complexity of the solution. The level of positional certainty with which an actuator can function is best quantified on the basis of the "minimum reproducible step size" achievable at the joint parameter.

The control system's contribution to the minimum step size is determined by the resolution of its sensors and the deadband characteristics of its servovalves. Based on these quantities, we can predict the smallest error that can be detected and potentially corrected. In a perfect mechanical system, the control system's precision would represent the precision of the total actuator.

In a real mechanical system, the attainable precision depends not only on the control characteristics but also on the backlash and friction within the device. Backlash occurs in the actuator structure wherever there is clearance in its power train (i.e., between bearings and shafts). Backlash combined with friction and stiction in the actuator creates a mechanical deadband effect that can significantly reduce the precision of operation.

In order to minimize the mechanically related position uncertainties, the use of preloaded bearings (tapered roller bearings or angular contact ball bearings) is recommended. Clearances should be as small as possible. The entire device should be preloaded (if possible) to prevent machine elements from traversing their clearances as the direction of external loading changes. Such preloading must be used judiciously, however, since frictional problems are likely to become more significant.

Component packaging must take into account the physical size and weight of the actuator and the overall utility of the design. The size of a manipulator in relation to its working volume needs to be small in order to increase dexterity and improve obstacle avoidance in the workspace.

Weight must be kept to a minimum to increase payload capabilities and to allow portability for certain applications. The strength to weight ratio of a manipulator's actuators should be as large as possible. Strength requirements of the actuators are determined by the load requirements on the manipulator and the geometry of the manipulator itself. This means that the strength to weight ratio of the system can be improved only by reducing the weight of the actuators.

In addition to all the structural and task related objectives, the designer must also create a "smooth and polished" product attractive in the marketplace.

The actuator module concept can go a long way in combining these four fundamental characteristics. For a given joint design, prime movers could be scaled in output characteristics to meet particular task requirements. In addition, the joint itself could be scaled up or down at the discretion of the designer. Stiffness and precision questions could be addressed on a joint by joint basis, allowing a simpler solution to error buildups in serial chains. The very concept of modular actuators requires a compact, clean component easily interfaceable with similar units. Component packaging is thus addressed by the fundamental nature of actuator modules.

SOFTWARE MODULES

As the desired performance of robots is expanded, they will necessarily become more sensor-based and more intelligent. This intelligence will involve an increased level of software. As suggested for actuator modules, the software system will be more rapidly developed and diffused if it is modularized. The system designer will be able to rapidly assemble a total software package from perfected modules that can be easily debugged or replaced with more effective units as they become available. Such modules could be designed to operate at the highest available sampling rates in hardware dedicated to the software module. Since such modules would be widely used, the associated hardware would become much less expensive.

Currently, all manipulators operate open loop, where neither the dynamics nor the external loads are accounted for. The next generation of robotic manipulators will require a high level of precision under loading. Dynamic model formulation in real time will allow compensation to create an electronically precise system.

One of the primary problems limiting progress towards real time operation of intelligent robots is that existing serial processors are poorly suited to treat the fundamentally parallel nature of robotic manipulators. For example, future systems will involve many sensors generating a large information array of all roughly equal significance to the controlling algorithm. There are six distinct computational levels which must be implemented serially in the dynamic model formulation. Within each of these levels, 100 to 800 distinct independent functions can be calculated in parallel. Hence, advances in parallel computer architecture, in association with the modular software, will allow a "smart" module approach.

Candidates for the modular processor approach are sensors, prime movers, joint encoders, end effectors, and vision systems. Each task level would involve sensory data from below interpreted by the module combined with commands from processors higher in the computational hierarchy. The spinal column serves a similar function in the human nervous system.

SENSOR MODULES

Vision, position, proximity, and force information are critical elements of intelligent control, machine intelligence, and precision. Recognition of this fact has generated much in the way of research and development activity. The vast array of sensing systems currently available for implementation on existing manipulators demonstrates the essential modular approach already being pursued in this area.

Vision has long been perceived as an important information feedback technology for intelligent machines. The primary barrier to applications of vision in autonomous operation is that the scene quantification of visual shape data requires high computational times. Further applications of vision systems will depend on increasing computational speeds through parallel processing or other specialized computer architecture. Clearly, the problems in image analysis will continue to be solved using component technologies in this highly successful example of modularity.

Positional information is required to determine the joint orientations and thus locate a manipulator in space. Precision operation requires high resolution joint position information. Progress in this area is such that angular resolution of 1 part in 1,000,000 is now feasible. Cost does become a factor at these high resolutions. In addition, structural deformation in the manipulator and lack of accurate dynamic modelling can often render such information useless.

Force sensing provides invaluable information which can be utilized within the control loop or as feedback to a man-machine interface. One major advantage of force feedback is that it provides information directly related to system accelerations. This can be useful to supplement or even replace the information obtained by differentiating the position data. Another important usage of force feedback data is in the formulation of the dynamic model.

All methods of measuring resultant forces depend upon the accurate measurement of elastic deformation of a structure of known compliance. Measurements are extracted by a variety of means, including metal film strain gages, potentiometers, piezoelectric materials, and diffused semiconductors in which strain is sensed by a change in resistance. To obtain a complete characterization of forces at the end effector, for instance, requires measurement of six orthogonal force components. It is obvious that a large amount of raw data processing may be required.

Since real time computational speeds are a critical aspect of obtaining intelligent control, the speed at which sensory data is reduced becomes important. Modularity applied to sensor technology could produce a "smart sensor", where preliminary reduction is done at the sensor. This concept would decrease the requirements on the central processor, allowing implementation of advanced control algorithms.

UNIVERSITY OF TEXAS STRUCTURAL MODULES

The robotics group at U.T. Austin has been involved in the design of structural robotic modules for many years. All of the joint designs stress the modular concept. It is possible then, as with any component designed with modularity in mind, to scale the module to the desired task. Additionally, in applications not requiring 6 DOF, combinations of actuator modules could be assembled as dexterity requires. In all, the joint module approach is seen as a way to quickly implement a manipulator into a given task. The designer is relieved of the burden of an entire system synthesis and can instead concentrate on applications.

ELBOW MODULE

Two separate 1 DOF elbows have been designed. One incorporates a single four bar mechanical amplifier while the other employs two four bar amplifiers in parallel. The latter design (Figure 1a) has been built; two hydraulic cylinders can operate in push-push mode for high positional resolution and a push-pull mode for maximum load capacity. This design goes far in addressing the points of strength, stiffness, and precision and at the same time is compact and modular as required in the component approach.

KNUCKLE MODULE

The 2 DOF knuckle (Figure 1b) again utilizes antagonism. The resulting gains in positional resolution and load capacity are similar to that of the 1 DOF elbow. The knuckle appears isometric in many of its structural properties. Because of intersecting journal axes, it becomes very rigid for its material content. Also, the stiffness of the joint is approximately the same in all directions, which means its assembly into a larger system as a module is not orientation limited. This design does suffer from a limited range of motion about each of its axes; however, this might be quite satisfactory for most applications.

WRIST MODULE

The wrist module (Figure 1c) is another conceptual parallel joint structure. Input from the prime movers is through a triaxial torque tube arrangement. Similar to the shoulder module, the geometry consists of a pair of tetrahedra, but with a moveable base and single link construction. Again, the benefits of parallel structure allow an increase in structural integrity and positioning ability.

SHOULDER MODULE

The 3 DOF shoulder module (Figure 1d) has a parallel structure, with prime movers driving one axis of each link. In essence, the joint consists of a pair of tetrahedra joined together at their edges by three spherical dyads. The parallel structure includes favorable characteristics such as precision positioning capabilities, distribution of loads, and increased stiffness. These characteristics enhance the structural integrity and subsequently reduce the amount of positional error produced. As such, the parallel shoulder could be used as a module in an otherwise serial structure.

CONTROL-IN-THE-SMALL MODULES

Conventional industrial robots have inherent limitations on the accuracy and resolution of end-effector motion, due primarily to the effects of friction, backlash, compliance, and inertia. One approach in dealing with this problem is the addition of a small-motion device, referred to here as a micromanipulator, between the terminal link of the robot and the end-effector. The augmented robotic system thereby retains the gross motion capabilities of the supporting robot while the micromanipulator provides an additional layer of high-bandwidth, high-resolution motions for error compensation, fine manipulation, and delicate force control.

While many researchers have devoted their efforts to the development and implementation of micromanipulating systems, this summary concentrates on the development of a unique, fully-parallel 6 DOF micromanipulator. The rationale for 6 DOF motion is that a typical spatial robot has, in general, corresponding spatial errors. Parallel rather than serial architecture has been chosen for reasons of compactness, rigidity, load capacity, and load distribution.

The particular mechanism that has been selected is a six-legged platform-type device that is specifically designed for small motions. A conceptual hardware design for the micromanipulator is shown in Figure 1e. Direct connection of the upper and lower end of each leg is made through a 3 DOF spherical joint. The desired platform motion is obtained by driving the six independent rotary inputs. Four-bar linkages are used to increase the mechanical advantage of the inputs and to improve the positional resolution of the output, relative to direct actuation of the grounded base joints. Flexural revolute and spherical joints suitable for small displacements have been suggested to avoid the backlash and friction associated with more conventional connections.

As another example of control-in-the-small, the 1 DOF small-control module (Figure 1f) uses a small, secondary prime mover to adjust for minor errors at the joint. This type of module is specifically designed to improve precision.

MINIATURE MANIPULATOR

Another unique concept under study at U.T. is the development of a high precision miniature manipulator. Such a system could be used for inspection, soldering, and electronic circuit assembly. An important new application of robotics could be realized in the field of microsurgery, where a precision manipulator could be operated remotely by a surgeon. An increase in precision of operation by a factor of ten could be achieved by filtering out jitters and oscillations at the input and by changing the scale of motion of the manipulator relative to the surgeon's input.

The conceptual miniature manipulator (Figure 2) consists of three universal (Hooke) joints of 2 DOF each. Since prime movers mounted at the joints would encumber such a small device, control is achieved by using three cables for each joint. Additionally, since the cables are always in tension, backlash is eliminated. Friction problems in the joints can be solved by using jeweled or ceramic bearings. Though by design it is an integral unit, modular concepts in software and control can readily be applied to the miniature manipulator.

In conjunction with the miniature manipulator, U.T. has conceptualized a **miniaturized 6 DOF force sensor** which would provide the extraordinary sensitivity necessary for control in microsurgery and micro-assembly. Figure 3 shows the sensor dome, which would undergo significant controlled deformations under light loading. The inner surface is etched with a micro-circuit in the same manner as used to form foil strain gages. As the circuit is deformed, the resistance measured through 6 distinct circuits would allow determinations of six components of force. Additional circuits may be desirable in more accurate control algorithms.

Due to the large amount of raw data generated, a local processor could be dedicated to create a highly modular package. Calibration and algorithm implementation within the micro-force sensor would create a package suitable for any operation requiring the detection of small forces.

CONCLUSION

Modularity allows each part of a robotic system to be optimally designed, scaled, and interfaced with other modules to produce a generic, versatile robot. The number of manipulator systems that can be derived from a series of such modules is virtually limitless. Once given a broad spectrum of choices, system designers would be able to quickly provide an optimum solution for a particular operation, without being forced to enter a lengthy design and construction phase.

The standardized modules would decrease the cost of a new robotic manipulator and eliminate the possibility of obsolescence - the module can be upgraded when a better model becomes available. The small, standardized modules could be improved less expensively than a whole new robot arm, allowing the robotic industry to make "tech mods" rapidly and take advantage of the most advanced technology. The final result would be a rapidly growing, efficient industry whose impact on manufacturing would rival the impact of the microchip on the field of electronics.

REFERENCES

1. Tesar, D., "The Development and Demonstration of a Teleoperated Modular "Snake" Robot System for Nuclear Reactor Maintenance", Proposal to the U.S. Department of Energy, Nuclear Energy Applications of Robotics, NPI NE-86-001, May 12, 1986 .

2. Tesar, D., Soldner K., "Assessment for the Physical Structure and Hardware of General Robotic Manipulator Systems", CIMAR, University of Florida, 1978.

3. Tesar, D., et al, "Final Report, Nuclear Reactor Maintenance Technology Assessment", CIMAR, University of Florida, March, 1980.

4. Tesar, D., Cox, D.J., "The Dynamic Modeling and Command Signal Formulation for Parallel Multi-Parameter Robotic Devices", CIMAR, University of Florida, September 28, 1981.

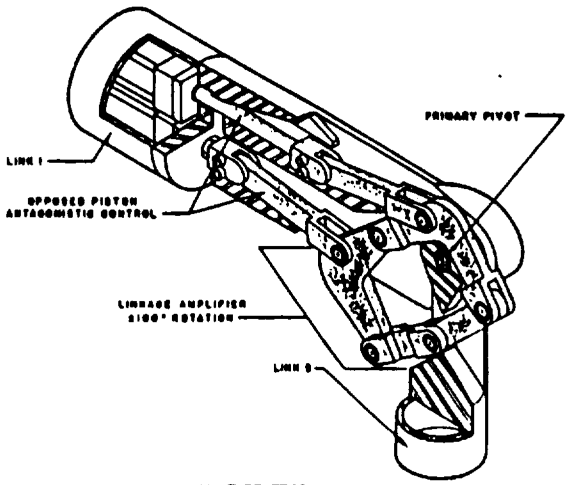
5. Tesar, D., Dalton, G.R., et al, "Assessment for the Design and Implementation of Robotics to the Secure Automated Fuel Fabrication Plant", October 1, 1983.

6. Tesar, D., et al, "Modular Actuator Designs Using Hydraulic Prime Movers in Antagonism", CIMAR, December 10, 1981.

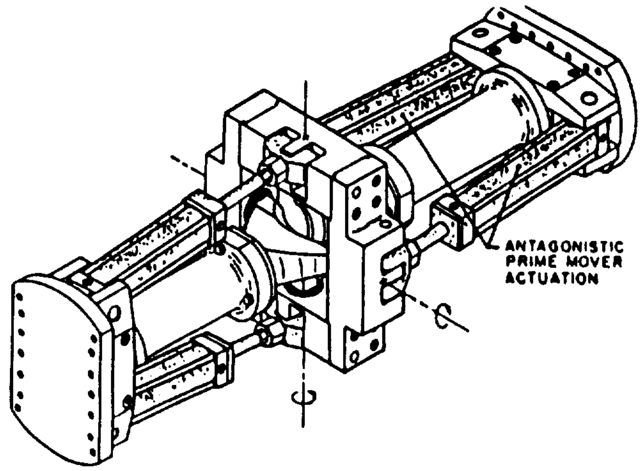
7. Tesar, D., "Next Generation of Technology for Robotics", University of Texas at Austin, February, 1985.

8. Craver, W., Tesar, D., "The Deflection and Force Analysis of a 3-DOF Shoulder Module", Report to DOE, The University of Texas at Austin, December, 1988.

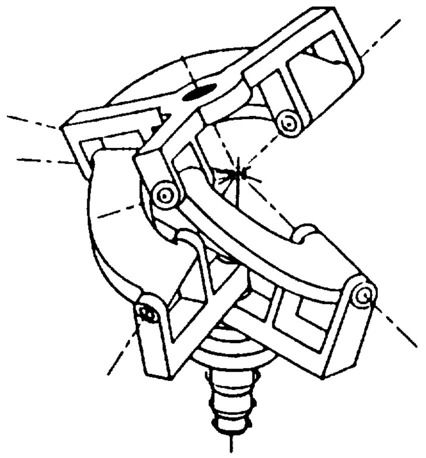
9. Tesar, D., U.S. Patent No. 4,505,166, "Control-in-the-Small System For Precision Under Load Control of Robot Manipulator", March 19, 1985.


PRIMARY PIVOT
LINK 1
OPPOSED PISTON ANTAGONISTIC CONTROL
LINKAGE AMPLIFIER ±180° ROTATION
LINK 2

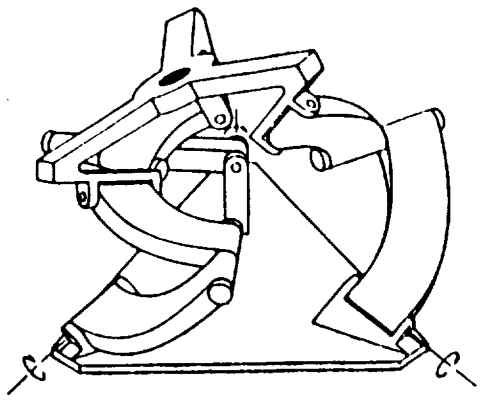
a. 1 DOF Elbow



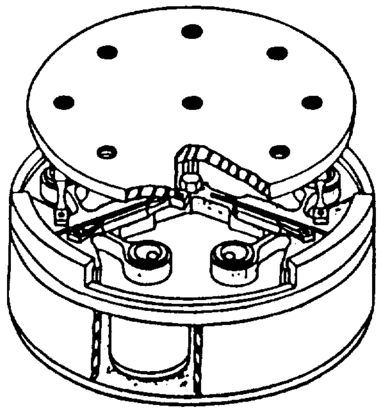
b. 2 DOF Knuckle



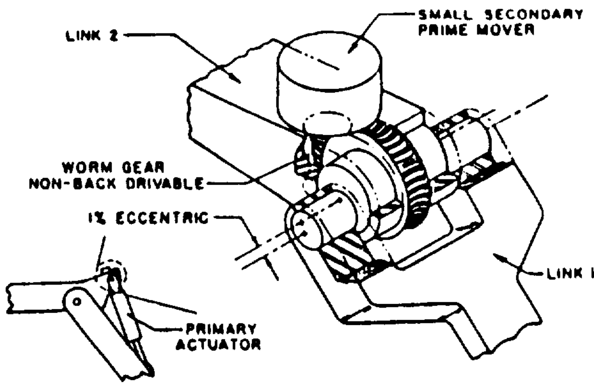
c. 3 DOF Wrist



d. 3 DOF Shoulder



e. 6 DOF Micromanipulator



f. 1 DOF Small-Control Module

Figure 1. University of Texas Structural Modules

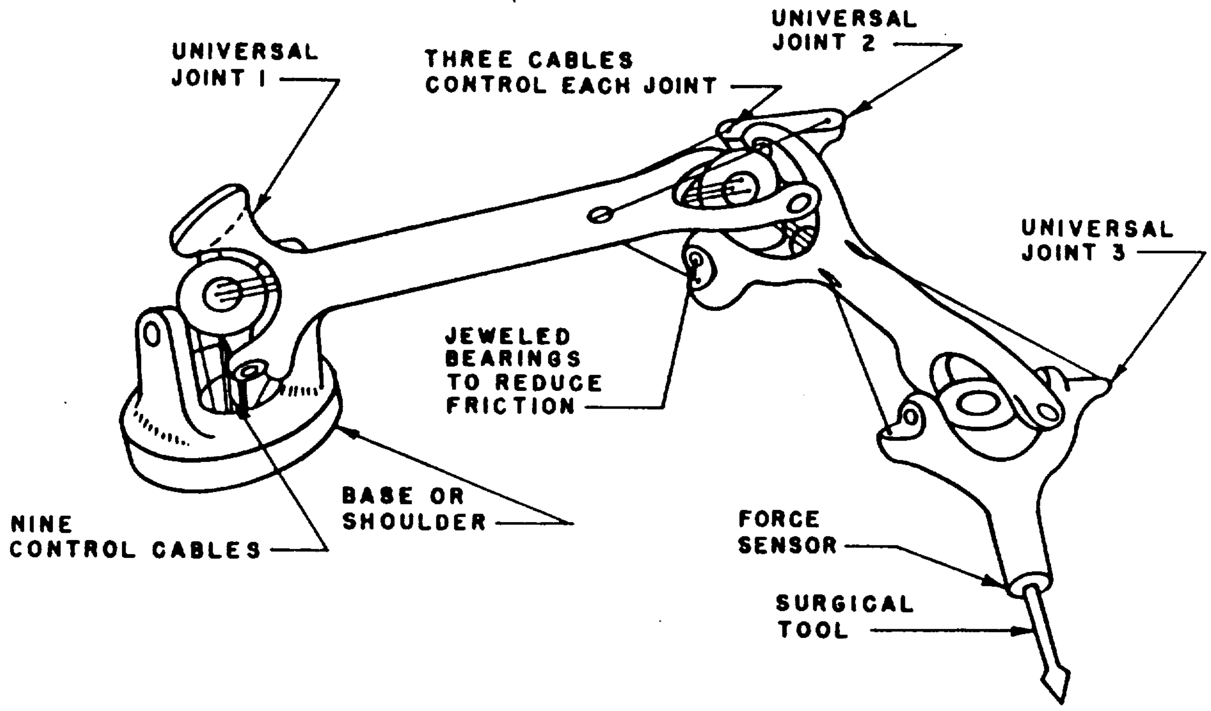


Figure 2. 3 Inch Miniature Manipulator

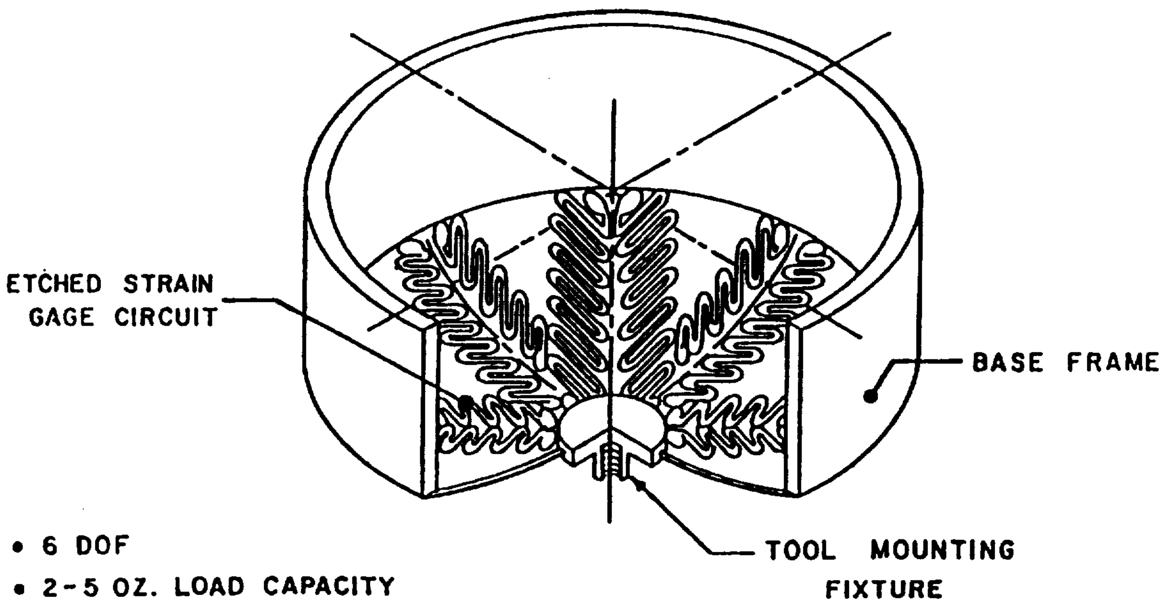


Figure 3. Micro-Force Sensor

N90-29015

A SYSTEM ARCHITECTURE FOR A PLANETARY ROVER

D.B. Smith and J.R. Matijevic
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
M/S 198-105 and 303-308, Pasadena, California 91109

ABSTRACT

Each planetary mission requires a complex space vehicle which integrates several functions to accomplish the mission and science objectives. A Mars Rover is one of these vehicles, and extends the normal spacecraft functionality with two additional functions: surface mobility and sample acquisition.

This paper assembles all functions into a hierarchical and structured format to understand the complexities of interactions between functions during different mission times. It can graphically show data flow between functions, and most importantly, the necessary control flow to avoid unambiguous results.

Diagrams are presented organizing the functions into a structured, block format where each block represents a major function at the system level. As such, there are six (6) blocks representing Telecomm, Power, Thermal, Science, Mobility and Sampling under a supervisory block called Data Management/Executive. Each block is a simple collection of state machines arranged into a hierarchical order very close to the NASREM model for Telerobotics.

Each layer within a block represents a level of control for a set of state machines that do the three primary interface functions: Command, Telemetry, Fault Protection. This latter function is expanded to include automatic reactions to the environment as well as internal faults.

Lastly, diagrams are presented that trace the system operations involved in moving from site to site after site selection. The diagrams clearly illustrate both the data and control flows. They also illustrate inter-block data transfers and a hierarchical approach to fault protection. This systems architecture can be used to determine functional requirements, interface specifications and be used as a mechanism for grouping subsystems (i.e., collecting groups of machines ,or blocks consistent with good and testable implementations).

1. INTRODUCTION

The history of operational planetary rovers begins with the USSR Lunokhod-1 mission on the moon, a nearly one year mission beginning in November, 1970. The Lunokhods were interactively controlled, starting and stopping according to planned sequences created by a ground mission team receiving TV images.

An advancement on this design principle is the JPL's Computer-Aided Remote Driving or (CARD) system implemented on a six-wheeled test vehicle. This system was developed under sponsorship of the U.S. Army Tank-Automotive command and demonstrated the capability of on-board execution of a human operator selected path drawn on a frozen image of the local terrain (Reference 8).

Current rover concepts vary from advanced concepts of this design (CARD) to highly automated vehicles performing automatic collision avoidance (for example, JPL's Semiautonomous Navigation or (SAN), Reference 1). Unlike the Lunokhods, concepts under study for a Mars rover mission (Reference 9) must accommodate in-situ sampling, very much like an automated geologist. This complexity dictates a flexible systems architecture invariant to different levels of automation.

Because of the nature of the mission, the architecture must combine the functionality of planetary spacecraft with the additional functions of mobility and sampling. Inclusion of these last two major functions dramatically expands traditionally layered architectural structures for spacecraft systems.

An architecture which incorporates a mobility function, must provide a structure for accommodating simple to complex walkers as well as the more traditional wheeled carriage vehicles. One of the simplest walker concepts is Brooks' 1 kg rover consisting of fifty-six (56) state machines cycled individually in a particular pattern to produce a "gait" (Reference 2). A more complex walker concept is the Carnegie Mellon University "ambler" (Reference 3). In between may be considered the elegant "beam" walker concept of Martin Marietta Corporation (Reference 4). In each case, a multi-layered architectural structure is suggested where control of individual walker link motors or wheels are coordinated at higher levels to produce the desired motions.

The addition of a sampling function introduces the added complexity of control of manipulations. As recent studies suggest (see for example the architectural concepts for telerobotics proposed for the Goddard Space Flight Center (GSFC) Flight Telerobotics Servicer (FTS) (Reference 10) and for the NASA/OAST Telerobot Testbed (Reference 11)) multi-layered control structures are required to coordinate manipulator and end-effector/tool link controllers. Precursor missions such as a Mars Sample return concept require a high level of automation of the sampling function. But later manned missions, with associated human interaction with the rover, require consideration of a range of supervisory control options for the sampling function. Consequently, human interactions must be smoothly integrated into a control architecture for the rover.

In this paper, a single architectural concept integrating all of the above is proposed for the general class of planetary rover concepts in the 1990's. The architecture is a loosely coupled, state-machine concept incorporating the hierarchical control concepts of NASREM (NASA/NBS Standard Reference Model for Telerobot Control System Architectures, Reference 5).

The following describes the architectural concept and provides a mobility scenario, tracing the sequential execution of several functions within the control layers of the architecture. A sampling scenario has also been done for completeness but is not presented here for the sake of brevity. Both scenarios validate the architecture's flexibility and accuracy.

Lastly, some final thoughts are presented on the uses of the architecture. These observations should apply to any good architecture, not just this one. For example, a typical systems design task is the mapping of the system functions into an implementation by a number of subsystems, defining boundary interfaces at simple junctions (e.g., functional levels). This architecture naturally decomposes into the standard subsystems for planetary spacecraft and provides a structure for the evaluation of alternate decompositions.

2. PLANETARY MISSIONS

Before discussing details of the architecture, we introduce some of the basic functions and subsystems of any planetary spacecraft. A Mars Rover, after all, is at least a planetary spacecraft and much more.

Basic subsystems of any planetary spacecraft include Telecommunication, Power, Thermal, Attitude Control, and Science. Common to these subsystems are three activities: receipt and/or processing of commands, telemetry output and fault protection. Due to the delays in transmission to a ground station, these subsystems must at a minimum fail safe, and the spacecraft as a whole must fail operational, albeit in a degraded mode. Consequently, each subsystem must detect errors and take local action such as the use of a redundant string. Once the error correction is accomplished the subsystem notifies a command and control subsystem (if available) of the configuration change. Of course, this information is passed on to ground command as soon as possible for evaluation and further corrective action.

No planetary spacecraft has been designed to be fully autonomous. Instead, these spacecraft are semi-autonomous (in the sense of the above discussion), relying heavily on ground control for mission commanding, analysis, and planning. Therefore, any general architecture for spacecraft control will include a significant ground segment. Functions may move from the ground to the spacecraft if more risk is assumed or capability made available. An example of this enabled the extended Viking mission. After the primary mission and with suitable confidence in the spacecraft capability, the Viking program reduced its operational ground staff significantly, by reprogramming the flight computers on the orbiters with automatic routines for fault protection.

The MRSR (Mars Rover and Sample Return) mission contains all the subsystems and functions of a planetary spacecraft except for the classical Attitude Control subsystem. Attitude of the Rover is monitored as a part of mobility but not controlled until some limit is exceeded. For example, an inclinometer may detect a dangerous tipping-over attitude which requires system action for correction.

The additional rover-specific subsystems are Mobility and Sampling. Mobility contains the local navigation function vital to semi-autonomous traverse. This local navigation function is very important, requiring interactions among other functions. Some architectures for a rover represent only this viewpoint. For example, Kovtunenko (Reference 6) represents the local navigation function as the main interface to the Earth and central to all other functions within the rover. This view is useful for describing control principles, but neglects other major rover functions.

3. MRSR MISSION

A MRSR mission has many variations (including some without a separate rover vehicle). The following summarizes main mission and operational requirements for a rover in the MRSR mission.

The rover greatly extends the number and types of samples which can be returned from Mars (Reference 7). A nominal rover operation entails a sequence of local traversing segments and sampling, constrained by on-board resources and ground interactions. Prolonged or extensive decision time by the ground operations can severely limit the rover's integrated range (from a goal of 40 km-100 km).

After each traverse, a sample may or may not be collected, depending on science value as determined by an on-board evaluation or ground mission team decision. Some strategies allow collection of samples without in situ discrimination. After a collecting tour, the rover locates the MAV (Mars Ascent Vehicle) for a rendezvous and sample hand-over. (The MAV may have resulted from an integrated rover or separate launch).

For the purpose of this paper, consider that each traverse segment is using semi-autonomous navigation. Figure 3-1 is a functional diagram of the activities. At a certain point, the ground up-links a topographic map from a ground-based Global Route Planner. This map contains a first-order ground swath for the rover's traverses and a designated path(s) avoiding obstacles and dead-ends that remains scientifically interesting.

The rover takes a panoramic view of the local scene using stereo cameras, laser scanning, or structured light, linked to machine vision and image processing functions. The rover computes a local map from the processed view and matches this map to the local portion of the global topographic map sent from Earth. Using the results of the match, constraints of previous rover positions and output of any other navigational devices, the rover determines an accurate position of itself. A revised (fused) map is formed from the two sources (ground/global and rover/local) to produce a high resolution map in the vicinity. A new path then is computed via simulation, revising the approximate path sent from Earth by including (and thus excluding from the path) small obstacles not detected by the low resolution images of the global map. The original global resolution is required to be accurate to 1 meter. The fused map could be accurate to 10cm. The rover then traverses the path.

In planning the route, the simulation of the traverse of the path is used to compute slope changes and tilt expectations. These are used in a predictive way to set limits on inclinometer and local proximity sensors. These predictions control the rover's reflex actions in the event an errant path is followed.

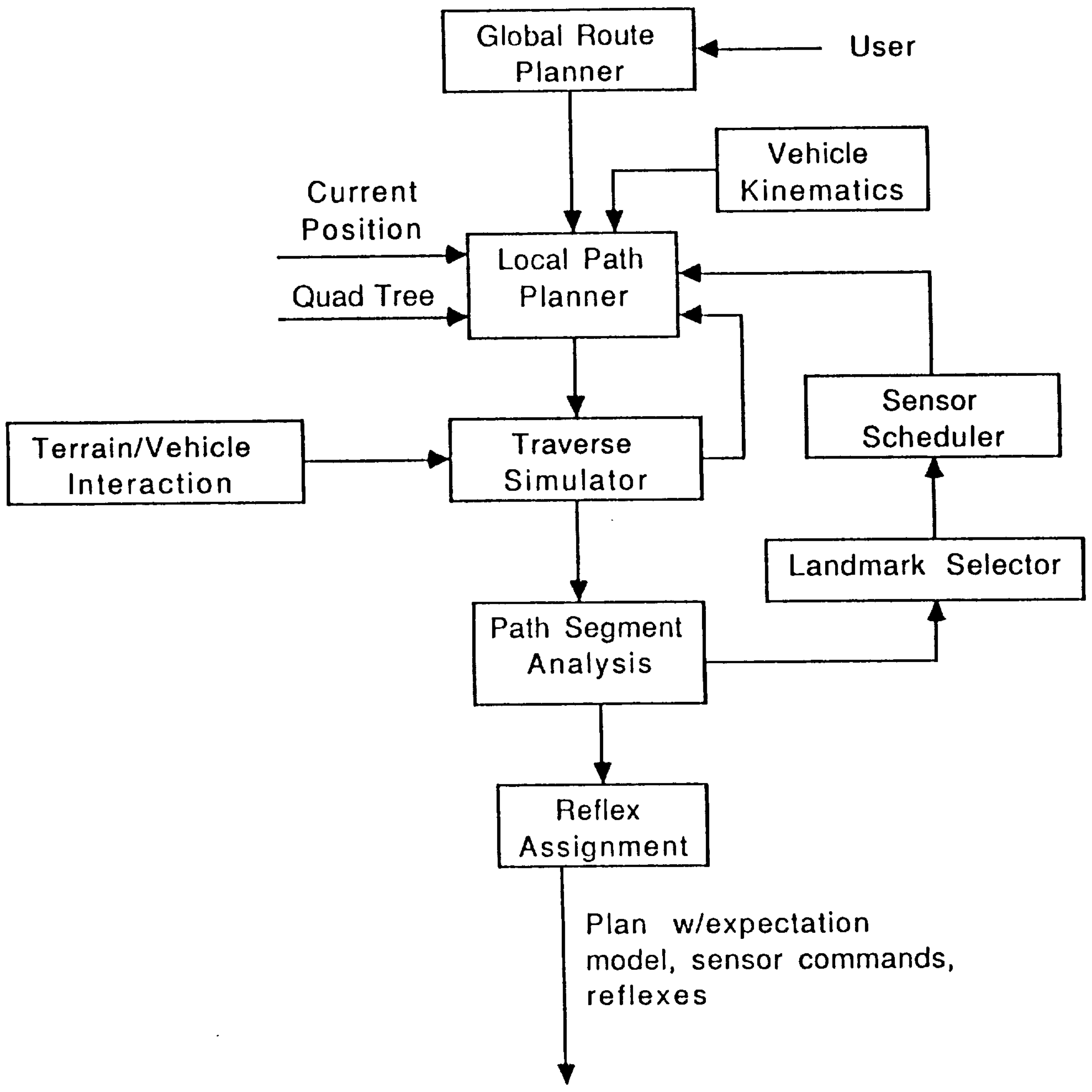

Global Route Planner
User
Vehicle Kinematics
Current Position
Quad Tree
Local Path Planner
Terrain/Vehicle Interaction
Traverse Simulator
Sensor Scheduler
Landmark Selector
Path Segment Analysis
Reflex Assignment
Plan w/expectation model, sensor commands, reflexes

FIGURE 3-1

If enough computational resources exist, the cycle of global map up-link(as needed) - fused map development-path planning - traverse can be repeated every few minutes, for a resultant average speed of 10cm/sec (see Reference 1). Typically a local path is computed for about thirty (30) meters, with the rover pausing every ten (10) meters or so to re-evaluate its condition.

4. NASREM

NASREM is a standard telerobotic architecture now adopted by the FTS, its contractors and many others associated with the Space Station program (Reference 5).

The NASREM is a modular, hierarchical conceptual architecture for telerobotic system control (see Figure 4-1). The main feature of NASREM are six layers of control:

(0) the world: hardware elements, being controlled by the telerobot and the environment of operation
(1) the servo level: coordinates are transformed and outputs servo the arms/end-effectors
(2) the primitive level: telerobot dynamics are computed and coordinated arm commands issued
(3) the E-move level: obstacles (including those inherent to arm operation) are observed and commands issued to avoid them
(4) the task level: tasks to be performed on objects are transformed into movements of effectors
(5) the service bay level: task on groups of objects in the vicinity of the telerobot are sequenced and scheduled
(6) the mission level: objects are collected into groups, resources are assigned between telerobots and parts/tools are routed and scheduled

Each NASREM layer is partitioned into three modules: sensory processing, world modeling and task decomposition. Additional features include a global memory to support the flow of information and coordination between levels in the hierarchy and an operator interface to support operator input and display capabilities at all levels of the hierarchy.

Since the control levels are well ordered, unambiguous commands flow from the top of the hierarchy down to the lowest level of the servomechanism. This is role of the task decomposition elements of the architecture. At each level in task decomposition, a job assignment manager partitions task commands into distinct jobs to be performed by one or more planner/executor modules. As such, each planner is responsible for decomposing a job command into a temporal sequence of planned subtasks. The executor evaluates the sequence prior to execution.

Data or status flows in reverse from the lowest levels of the hierarchy to the top levels. This data is available from three sources. The data may be fed back from one level to the next through the hierarchy in task decomposition. Alternately, depending on the use/need, the data may be read from the global memory. This global memory is a data base where knowledge about the state of the task space, task environment and internal state of control system is stored. Each layer of the hierarchy and all processing modules within a layer contribute to global memory. Lastly, data may be received through the interaction of the telerobot system with the environment. This data is obtained through the sensory processing elements of the architecture. Sensor information is read and processed in a hierarchy which allows the system to recognize patterns, detect events, filter and integrate information over space and time.

The processing of the data or status is performed using models of the effectors, sensors or environment of the telerobot. This is the role of the world model elements of the architecture. These models include estimates and evaluations of the history, current state and possible future state of the telerobot system and task space. These models help maintain the data in global memory, offering confidence levels/statistics of model predictors and sensory observation.

The last elements of the architecture are contained in the operator interface. The functions in these elements enable interface to each level of the hierarchy. The operator can enter the control flow to monitor a process, insert information, interrupt automatic operation, take over, and apply human intelligence to the processing. Feedback ranges from force reflection at the lowest levels to displays for interactive scheduling at the highest levels.

In guiding functional partitioning, this architecture constrains functions to a given layer by processing rate or bandwidth. At the lowest levels, the processing is severely time dependent in stabilizing servo control loops. As such, rates of execution may be as high as 1000 times per second (or 1000Hz). At each higher level the rates decrease generally by a factor of 10 or more.

In implementing the architecture, the available technology and operational priorities dictate additional constraints. The processing load at the lowest levels lead to optimization of communication paths. Thus, the data in global memory which serves inter-process communication at these levels may be kept separate from data at other levels, with access prioritized by need for the control loops. The need for verification of command sequences at the higher levels of the architecture lead to the inclusion of simplified (though correct) models of specific hardware in the world models at these levels.

5. ROVER SYSTEM ARCHITECTURE

The system architecture for the rover discussed in this section is based on the NASREM model. We generalize this model in two ways. We unify treatment of rover subsystems and specific functions such as command and telemetry by modeling each in a two-dimensional NASREM architecture. We collect the elements in a four dimensional array space, allowing a simple mechanism for sorting elements by like function and processing. In addition, we generalize the definition of the intermediate layers (Levels 1 through 3) of the architecture allowing expansion to such functions as mobility and telecommunications, while maintaining the spirit of the definition of these levels for a telerobot.

In considering then the definition of a NASREM model for use in a rover system, we utilize the concept of small state machines introduced by Brooks (Reference 2). Level 1 functions can simply be interpreted as those state machines which implement the settings of dynamical systems or the states to be controlled. In the Brooks' walker, these states are the different positions of the legs. In the control of a robotic arm the states are the various settings of the joints of the arm. Level 2 functions set permissible states as represented by a control law, constrained by various dynamic and kinematic models. At the next level, a sequence or function of permissible states implements a subtask or operational process for the rover. At this level, for example, the movement of a walker along a path is a sequential execution of repositions, with each reposition itself a

JPL

NASREM - NASA/NBS STANDARD REFERENCE MODEL

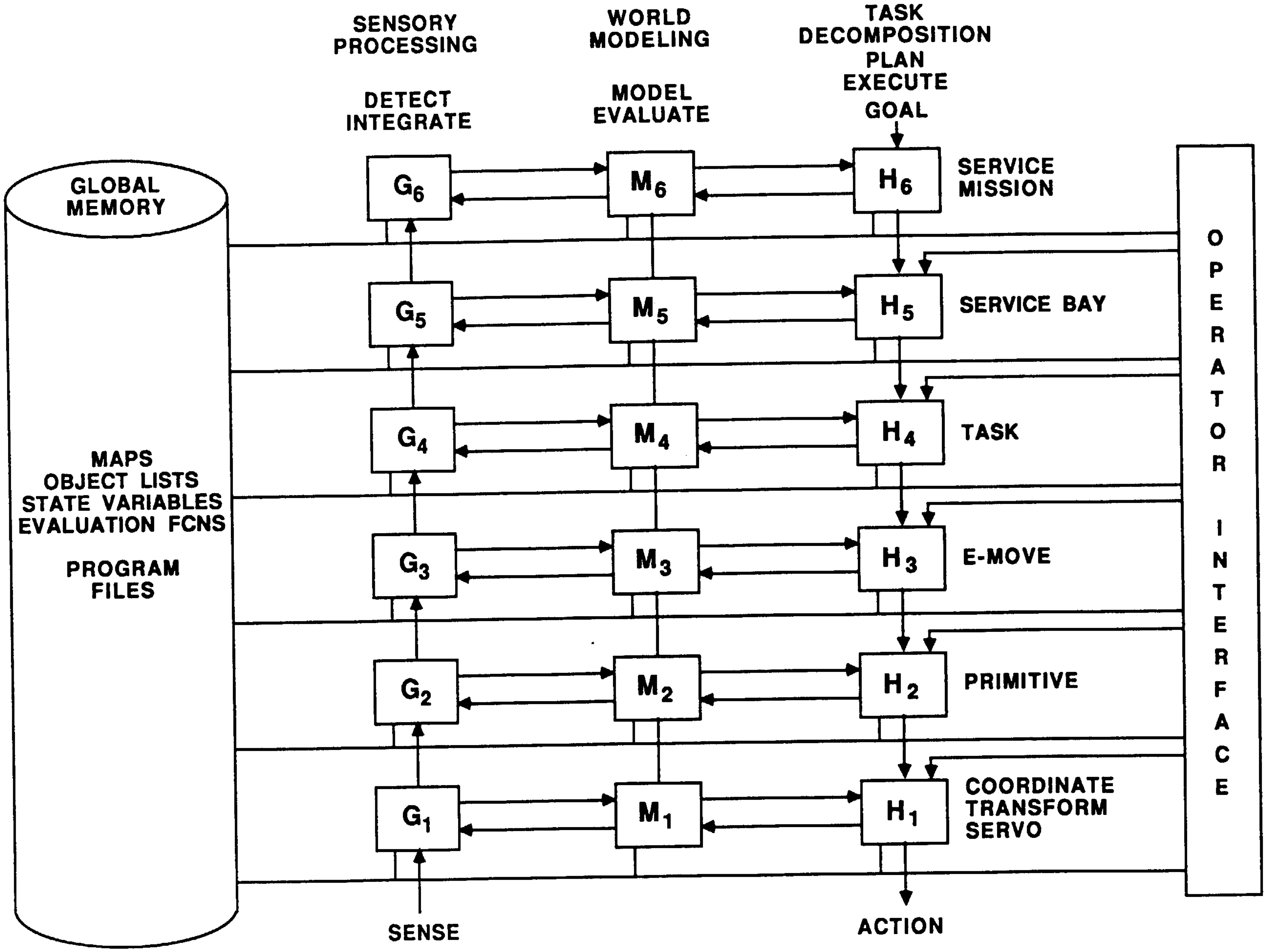


FIGURE 4-1

sequence of leg motions in a controlled, coordinated (or gaited) move. For an arm, a cartesian-space referenced path is achieved by moving the arm joints to achieve a series of end-points. Task planning at the highest level generates collections of sequences which accomplish a task. For example, movement of an arm or rover from point A to B is achieved by the execution of a set of commanded path segments.

Figure 5-1 is a generalized version of the NASREM model where this new state-definition terminology is applied. In another change from Figure 4-1, we have re-ordered the columns adding a slice so that a distinction is made between world models of the environment being sensed and world models of devices being commanded. The global data base becomes the middle slice and contains the constants and parameters of the world models as well as the data comprising the state of the overall system.

To the functional processing machines, we have added a third dimension to the architecture: a stack of command and a stack of telemetry machines. Commands flow down and telemetry flows up. In between is a separate stack of fault protection machines which can be excited and alter commands and telemetry at any given time. Fault protection machines execute as a result of a detected error read through telemetry. They implement recovery actions through commanding of the functional processing elements to assume a new state. As an example for a mobility subsystem, an inclinometer may sense an excess tilt angle. A corresponding fault protection machine will detect the error based on the telemetry and then act by issuing a command to cease forward motion. In addition, an appropriate routine may be executed such as carefully retracing the last series of commands until acceptable tilt angles are achieved (i.e., back-up). Once completed and the vehicle is safed, a route replanning will be commanded.

Notice that in the above example command and telemetry machines interact with the fault protection machines. The fault protection machine orchestrates the actions of the functional processing machine(s) implementing recovery. In detail then for the above example, an inclinometer at Level 1 registers excessive tilt causing a Level 1 fault machine to interrupt commands from Level 2 thereby halting the system. Telemetry is then sent to the Level 4 fault machines which have been receiving some subset of the last successfully commanded states. The Level 4 fault machine then calls for the task execution of a traverse back to a point along the route. Commands then flow normally downward until this previous state is achieved. When this happens, system control is then passed back to the task planner which knows its path plan has been altered and must replan a new path, which has a prescribed set of greater margins of expected tilt angles along its simulated path.

If any of these actions require more power or other subsystem intervention, then there is data flow between subsystem. Each such subsystem is represented by a block as shown in Figure 5-2. This is the fourth dimension of the architecture.

Returning to the basic functions of a spacecraft, we can now construct the entire model of a Mars Rover (see Figure 5-2). It consists of seven "cubes" in all. The Data Management System is an automated version of the Command Data System, complete with a tasking level. Each element is a state machine, so there are 560 state machines not counting the world elements at the bottom. Notice this is exactly ten (10) times the 56 machines of the insert Brooks' walker. Furthermore, each small machine is considerably more complex.

There are some interesting features of this architecture. For example, the front faces all represent the control flow, while the sides are data flow. This gives one a complete look at the total information system during the design process. Also, it is important to note the information flows <u>between</u> blocks. This is accomplished using the global memory of each face of each block. The four dimensional property of the architecture allows a an arrangement which make communication possible among the global memory (i.e., global memory of the system architecture is a 'block' cross-section of the four dimensional 'cube').

There is much information being exchanged between the Data Management System block and the other subsystem blocks. Also note that data from the ground first appears in the Telecommunications block and then to Data Management. Data Management controls the other blocks through its commanding sequence in order to accomplish a task.

As an example of a planetary spacecraft viewed in this architecture, Voyager's task planning was a function of the ground mission control team. Its Data Management System is the CCS or Command & Control Subsystem. Its command and control functions can be mapped to the lower two (2) levels of the DMS architecture (see Figure 5-3). The mobility function (Figure 5-8) degenerates to a three level Attitude & Control Subsystem. Sampling disappears altogether. In this architecture the Voyager FDS (Flight Data System) is the sum of all the telemetry state machines. The Science block contains the commanding of the science platform, which may be represented in a single level architecture.

The Fault Protection Subsystem for Voyager consisted of automatic recovery routines; little on-board tasking based on sensory information was allowed. Therefore we at once can write down the totality of the Voyager Flight Fault Protection system as the sum of the Fault Protection slices up to level 3. This sum is given in Figure 5-2.

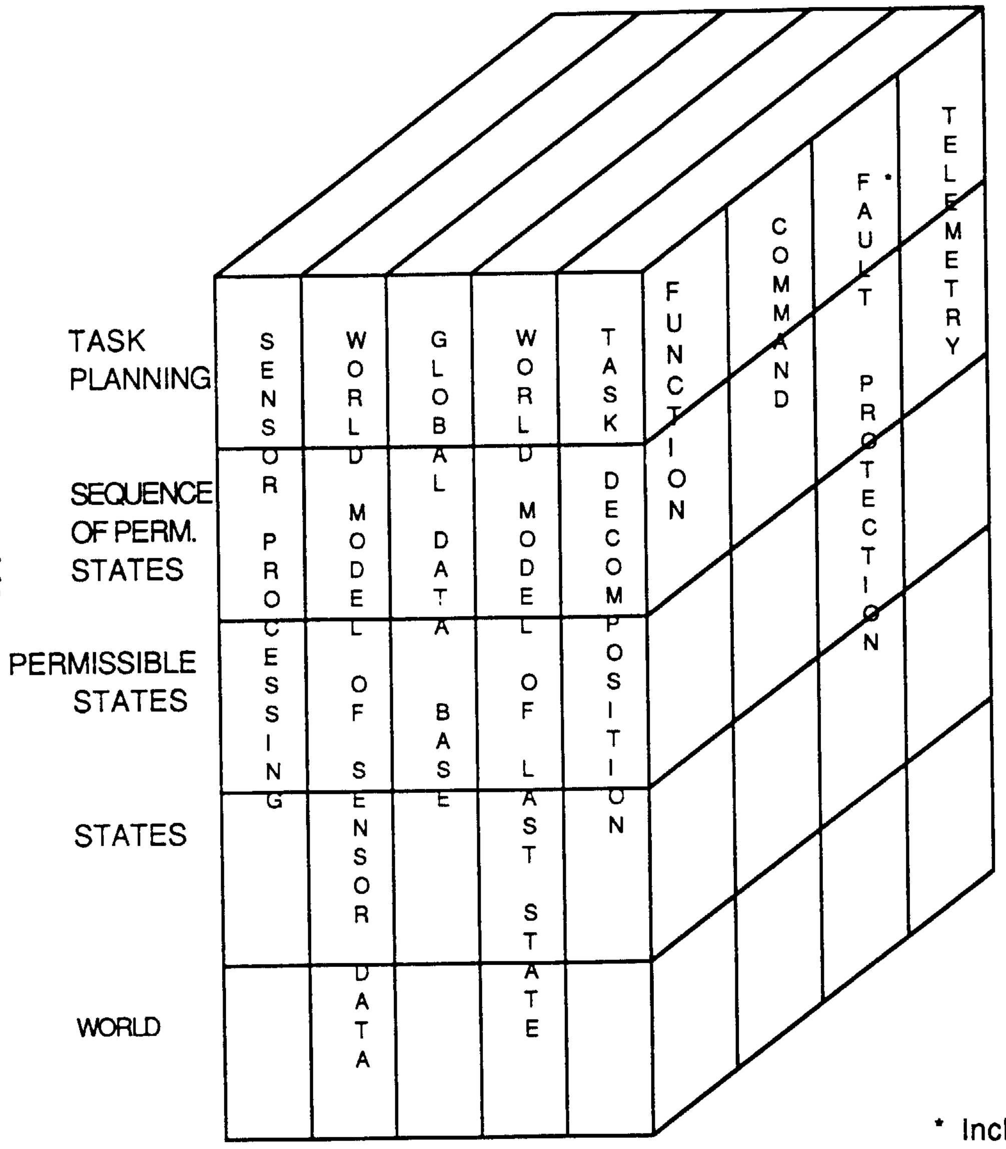
We wish to emphasize that all parts of the blocks exist on Voyager and only certain levels were flown. The remaining levels for Voyager were all on the ground. As spacecraft become more automated, functions and/or levels can be transferred to the spacecraft. These trade-offs are not always implemented since a degree of risk is incurred by this transfer.

The current concept of the Rover has more of these functions on-board than standard planetary spacecraft. In this case, increased on-board autonomy not only reduces ground operations but increases range and (thereby) science measurably.

The remaining Figures 5-3 through 5-9 round out the complete architecture. Notice that some state machines are inoperative, a reflection of today's level of conceptual design. In most cases, however, these state machines are simply not needed (see e.g., Thermal above level 1 in Figure 5-6).

In order to penetrate the design further, the following section will focus on the mobility block which contains the local navigation function. As we mentioned earlier, this is so important to the rover designers that they often view the world as a large mobility block supported by other subsystems.

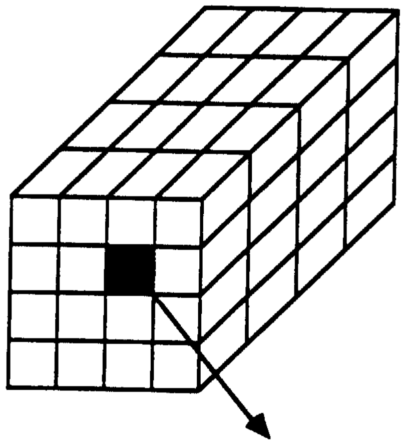
ROVER SYSTEM ARCHITECTURE



* Includes Health and Maintenance

FIGURE 5-1

ROVER SYSTEM ARCHITECTURE



EXAMPLE : (VOYAGER FAULT PROTECTION)

$$FP(3) = \sum_{k=1}^{6} \sum_{i=1}^{5} \sum_{j=1}^{3} M^{k}_{ij}(3)$$

l = 1, ..., 4 = faces
k = 1, ..., 7 = cubes (subsystems)
i = 1, ..., 5 = columns (proc. categories)
j = 0, ..., 4 = rows (hierarchy)

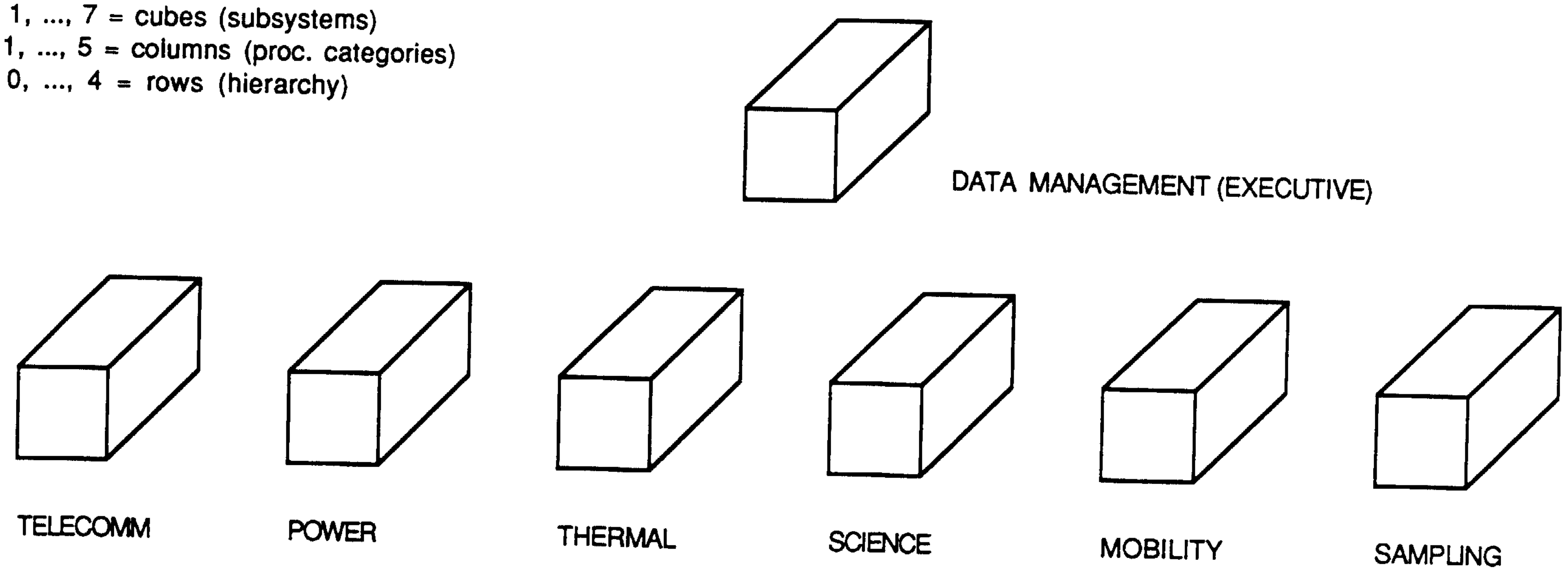


FIGURE 5-2

DATA MANAGEMENT (EXECUTIVE)

SENSOR PRSING.	(SENSOR) WORLD MODEL	GLOBAL DATA B.	(STATE) WORLD MODEL	TASK DECOMP.	
o Task progress reports	o Task error models	o Sampling locations and constraints o Science tasks	o Simplified task execution models o Task constraint and resource models	o Planner for task execution o Task commands	TASK PLANNING
o s/s operational condition	o s/s operation error models		o Simplified s/s operational models o Simplified s/s constraint and resources models	o Planning for s/s operation o Sequencing of s/s operations	SEQUENCE OF PERMISSIBLE STATES
o s/s state condition	o s/s state error models		o Simplified s/s state model o s/s state constraint and resource models	o Organization of s/s states o s/s state commands	PERMISSIBLE STATES
o Data reports from s/s	o Command acceptance feedback o Emergency safing condition model	o Emergency state condition	o Last commanded state in s/s's command table	o Decomp. of state commands o Commands to subsystems	STATES
o Reporting functions of subsystems				o s/s's of TELECOM, POWER, THERMAL, MOBILITY, SCIENCE, SAMPLING	WORLD

FIGURE 5-3

TELECOMMUNICATIONS

SENSOR PRSING.	(SENSOR) WORLD MODEL	GLOBAL DATA B.	(STATE) WORLD MODEL	TASK DECOMP.	
					TASK PLANNING
		o Transmission point in inertial space o Vehicle location in inertial space	o Trans-mission models o Trans-formations	o Timing and transmission sequence o Direction in inertial coordinates	SEQUENCE OF PERMISSIBLE STATES
o Decoded signal o Location of antenna in vehicle coordinates	o Decoding scheme	o Data for/ from transmission o Vehicle location	o Encoding scheme o Transforms	o Direction for pointing in veh. coord. o Trans. to servo cmds. o Transmitter configuration o Coded data	PERMISSIBLE STATES
o Decommu-tation signal o Change in encoder counts	o Decommu-tation scheme o Readouts for feedback to motors and antenna configuration	o En/Decoded data for transmission o Servo state condition	o Commuta-tion scheme o Servo gains o Configura-tion model	o Commutation signals for transmissions o Servo motor control o Transmitter configuration commands	STATES
o Encoders/ motors o Receivers				o Antennas and motors o TWT's and trans-mitters o Trans-ponders	WORLD

FIGURE 5-4

P O W E R

SENSOR PRSING.	(SENSOR) WORLD MODEL	GLOBAL DATA B.	(STATE) WORLD MODEL	TASK DECOMP.	
					TASK PLANNING
		o Load state changes	o Load transition models	o Load transition commands	SEQUENCE OF PERMISSIBLE STATES
		o Load commanded state and last state achieved	o Load state o Load/ vehicle state model	o Load calculation o Load balance commands	PERMISSIBLE STATES
o Open/close junction state o State of discharge/ charge	o Feedback for control loops	o Charge/ discharge state o Array servo state	o Charge/ Discharge model o Distribution model o Servo commands	o Network distribution switching commands o Charge/ Discharge control loop o Servo control	STATES
o Voltmeters o Ampmeters o Network junctions				o Power generation: RTG's, Arrays o Power storage: batteries o Power distribution network	WORLD

FIGURE 5-5

T H E R M A L

SENSOR PRSING.	(SENSOR) WORLD MODEL	GLOBAL DATA B.	(STATE) WORLD MODEL	TASK DECOMP.	
					TASK PLANNING
					SEQUENCE OF PERMISSIBLE STATES
					PERMISSIBLE STATES
o Heater state o Temperature o Change in counts	o Temperature feedback	o Heater commands	o Heater state model o Vehicle thermal model	o Heating control loop o Heater state commands	STATES
o Rheostats o Thermostats o Encoder on louvers				o Louvers o Heaters	WORLD

FIGURE 5-6

SCIENCE

SENSOR PRSING.	(SENSOR) WORLD MODEL	GLOBAL DATA B.	(STATE) WORLD MODEL	TASK DECOMP.	
					TASK PLANNING
o Features correlated with expect. o Locations of inst. in inertial coordinates	o Expectation models: geology form. mineral content	o Terrain-map-based features of interest o Vehicle inertial location	o Sampling or science constraint/ locales o Transforms	o Planning of science imaging for sampling o Inertial coord. directions	SEQUENCE OF PERMISSIBLE STATES
o Science/ Sampling features extracted o Locations of inst. in vehicle coord.	o Geometric and spectral processing parameters o Feedback of inst. pos. in vehicle coordinates	o Vehicle location o Commanded state	o Transforms	o Directions for imaging o Coordinate transforms o Processing into state commands	PERMISSIBLE STATES
o Spectral and signal processing o Digitized imagery o Change in encoder counts	o Sounder parameters o Spectral band param. o Feedback for servos	o Servo state condition	o Last servo cmd. o Servo gains	o Servo command o Servo/motor control law	STATES
o Encoders for the motors o Imaging Spectrometer o Sounder/ Antenna				o Motors for Imaging Spectrometer scan platform o Motors for Sounder, Antenna articulation	WORLD

FIGURE 5-7

M O B I L I T Y

SENSOR PRSING.	(SENSOR) WORLD MODEL	GLOBAL DATA BASE	(STATE) WORLD MODEL	TASK DECOMP.	
o map for use in correlations		o route commanded	o expecta- tions model o sampling objectives o simplified vehicle models	o updated expectations o path planning o correlation of features with constraints o local path in the terrain	TASK PLANNING
o feature determination o vehicle state progress	o vehicle state model	o fused terrain map o commanded local path	o simplified vehicle model with con- straints o last cmd.'d path o obstacle model	o way points of interest to achieve o obstacle detection/avoid o local path calculation and commands	SEQUENCE OF PERMISSIBLE STATES
o feature map o vehicle inclination and heading o ground speed o distance traversed	o sensed state of vehicle and terrain o sum.'d/ diff.'d readouts	o commanded vehicle state o readouts from sensors	o commanded states o vehicle kinematics and dynamics	o commanded turns, moves up/down o coordinated commands to wheels and steering	PERMISSIBLE STATES
o digitized local map o range cnts. o angular cnts. o abs./rel. change in position in veh. coord.	o range reads o camera models o encoder, gyro readouts o digitized imagery	o servo state cmds	o servo gains o gear state o last cmd.'d servo state	o servo commands o servo/motor control laws	STATES
o Encoders o Cameras o Lasers o Inclinometers o Gyros/IMU o Accelerometers o Satellite cameras o Odometers/ speed sensors				o wheel motors o gears o drive train o steering motors o motors for pan/tilt head	WORLD

FIGURE 5-8

SAMPLING

SENSOR PRSING.	(SENSOR) WORLD MODEL	GLOBAL DATA B.	(STATE) WORLD MODEL	TASK DECOMP.	
o Fusion of features with terrain	o Task error models o Local terrain models o Feedback of sequence states	o Sampling site maps o Commanded sequences	o Simplified kinematic models o Spatial models of sampling sites o Resource models	o Planning of sampling task o Commanded sequences of grasps, manipulations and placement	TASK PLANNING
o Fusion of position, force with data base of objects sampled	o Sampling error models o Sensor-based feedback	o Manipulation sequence o Sampling states, site data	o Kinematics and dynamics of sets of links, end-eff. o Sampling site data base of positions and sizes	o Inertial coordinate transforms o Collision detection/ avoidance o Cmd.'d pos. forces, way pts.	SEQUENCE OF PERMISSIBLE STATES
o Ranges o Edges, Vert. centroids o Features extracted o Positions, forces in cart. coords., of veh	o Position and force control loop feedback	o Commanded state of links and end-effectors	o Kinematics and dynamics of sets of links o Tool and end-effector sizes/char. models	o Reference coordinate transformation o Trajectory calculations o Coordinated link/motor/end-effector cmds.	PERMISSIBLE STATES
o Stereo correlation o Digitized images and sensor readouts o Change in counts of pos. rate, range, acceleration	o Servo loop feedback	o Servo state condition	o Last servo state o Servo gains	o Servo cmd. o Servo/ motor control	STATES
o Encoders o Sensors of force/torque position, proximity o Cameras o Lasers				o Mechanical links o Motors o Tools and end-effectors: sampling devices	WORLD

FIGURE 5-9

The mobility block is shown in Figure 5-8. The highest level is focused on the task of developing then utilizing a local terrain map in route planning as the discussed under Section 3. After a 30 meter planned path is selected, the vehicle can begin to roll. The rover executes this 30 meters 10 meters at a time. The execution of this 10 meter traverse is detailed below.

The presented example shows the interaction between the blocks as well as control loops within the layers of given blocks. As we mentioned before, sampling is not presented for the sake of brevity. However, it should be noted that a sampling example was easily constructed since sampling is (assumed) implemented using telerobotics and the basic model was derived from NASREM, an architecture for telerobotics.

6. MOBILITY

In this section we will give an example of an operation of the rover vehicle and the accommodation provided by the functional architecture. In this operation, a command has been received by the rover from the mission team to move to a new location, based on the transmitted views of the site. The rover system must evaluate the surrounding terrain and generate a detailed route which reaches the commanded location. In arriving at a decision that the route so generated is suitable for traverse, the rover system must evaluate its state and determine on-board the feasibility of execution of the route. If so feasible, the rover begins the traverse. Periodic evaluation of progress to the goal state (i.e., the commanded location) during the traverse allows the rover system to determine the end of the operation as well as to proceed within available resources and within limits of safe operation.

In summary fashion the steps executed by the rover system in accomplishing the traverse include:

(1) tasks performed by the ground support team in determining the new location for the rover
(2) the up-link and
(3) on-board processing of command sequences which result in receipt by the rover of new goal state
(4) the determination of a feasible route to the goal state by the rover system
(5) the planning necessary to determine a route
(6) the execution of a traverse along the planned route.

In performing these steps several subsystems within the architecture interact to execute the required functions. Particular capabilities of these subsystems in accomplishing these functions are identified in summary fashion:

(1) EXECUTIVE: sequencing of subsystem support in the determination of a feasible route;
collection/evaluation of periodic reports of progress during execution of the traverse
final acceptance of reaching of the goal state

(2) TELECOMMUNICATION:
commutation and de-commutation of commands and telemetry

(3) POWER: determination of available power resources in support to route planning

(4) MOBILITY: provide imaging data of the near vicinity of the rover;
perform route planning;
execute rover movements which effect the traverse along the route

(5) SCIENCE: provide instrument data products which support evaluation of the site

The following is a detailed discussion of the example of execution of a traverse. The flow of activity follows the 'exploded' pictorial representation of the subsystems given in Figures 6-1 to 6-4 . Interspersed in the discussion is bracketed ([...]) references to specific steps shown as a flow of activity in the architecture on these figures.

6.1 GOAL DEFINITION

The mission science team evaluates the latest data concerning the geological and mineralogical properties of the site surrounding the rover. This data is a compilation of over-flight imaging taken by the companion orbiter, data available from past missions (e.g., Viking), observations by the on-board rover science instruments (possibly an imaging spectrometer and sounder), and range and feature data provided by the imaging components of the on-board rover mobility system. A new location (or set of locations) of interest is selected and registered as a (set of) mission objective. In this example a location 10m from the current site of the vehicle is identified [1a of Figure 6-1].

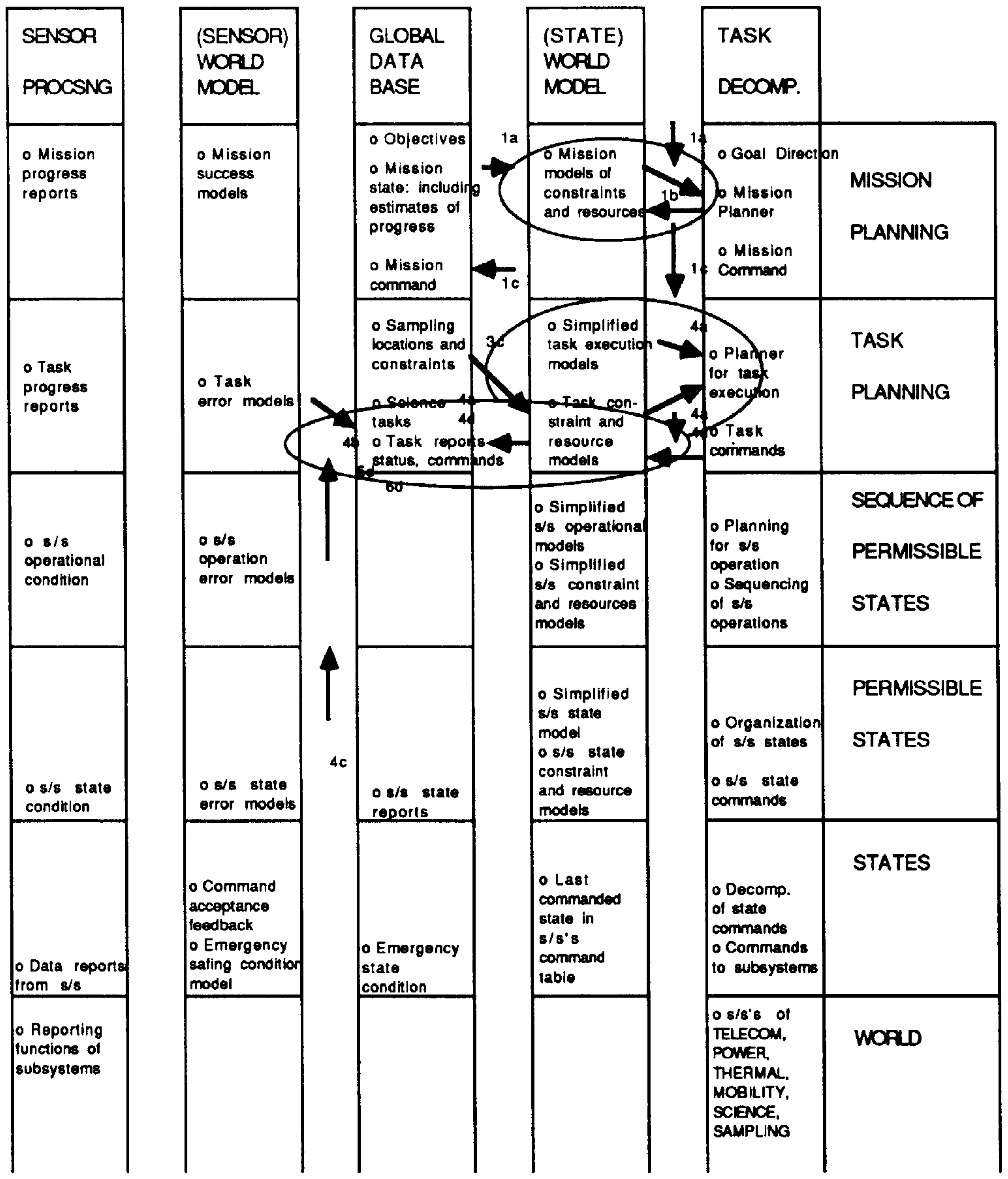
The vehicle team in mission control evaluates the selected locations based on models of (recent) past rover performance. An evaluation of feasibility in terms of vehicle health state and resource availability is performed [1b of Figure 6-1]. Simulations of the vehicle traverse over the terrain (available from over-flight and on-board rover imaging) assist in providing the feasibility check [interactive loop of Figure 6-1 between world model and task decomposition at the Mission Planning level of the executive]. A recommendation is provided to a mission director [the mission planner of task decomposition at the Mission Planning level of Figure 6-1], who in turn weighs the science return/objective(s) against the vehicle utilization. Conflicts (if any) are resolved and a new location (or set of locations) are identified for the rover [1c of Figure 6-1].

6.2 UP-LINK

The new location (in an appropriate inertial frame) along with related route information, including the expected length of the traverse (time, distance) and points of interest along the route (for collateral imaging and science instrument observation), is passed to Telecommunications for encoding [2a of Figure 6-2], commutation and up-link transmission [2b of Figure 6-2]. [N.B., These steps are performed by a separate though similar Telecommunications subsystem located on earth. For purposes of illustration these steps are shown on Figure 6-2.]

In addition, appropriate support data is commanded to other systems which will provide the latest update to the rover. This includes a topographic map generated as a result of over-flight by the planetary orbiter and a new reference location in inertial space for the rover developed by interferometry from tracking data.

D A T A   M A N A G E M E N T   (E X E C U T I V E)



SENSOR PROCSNG	(SENSOR) WORLD MODEL	GLOBAL DATA BASE	(STATE) WORLD MODEL	TASK DECOMP.	
o Mission progress reports	o Mission success models	o Objectives o Mission state: including estimates of progress o Mission command	o Mission models of constraints and resources	o Goal Direction o Mission Planner o Mission Command	MISSION PLANNING
o Task progress reports	o Task error models	o Sampling locations and constraints o Science tasks o Task reports status, commands	o Simplified task execution models o Task con-straint and resource models	o Planner for task execution o Task commands	TASK PLANNING
o s/s operational condition	o s/s operation error models		o Simplified s/s operational models o Simplified s/s constraint and resources models	o Planning for s/s operation o Sequencing of s/s operations	SEQUENCE OF PERMISSIBLE STATES
o s/s state condition	o s/s state error models	o s/s state reports	o Simplified s/s state model o s/s state constraint and resource models	o Organization of s/s states o s/s state commands	PERMISSIBLE STATES
o Data reports from s/s	o Command acceptance feedback o Emergency safing condition model	o Emergency state condition	o Last commanded state in s/s's command table	o Decomp. of state commands o Commands to subsystems	STATES
o Reporting functions of subsystems				o s/s's of TELECOM, POWER, THERMAL, MOBILITY, SCIENCE, SAMPLING	WORLD

Goals/Objectives: 10 m from current location with constraints X.

Evaluate objectives with mission model and mission status, generate mission commands. ⬭

Mission commands: destination, length of traverse, points of interest in inertial coordinates.

Plan a sequence for Mobility planning and execution of a traverse task. ⬭

Task commands to Mobility to simulate and plan a traversal against the given objectives.

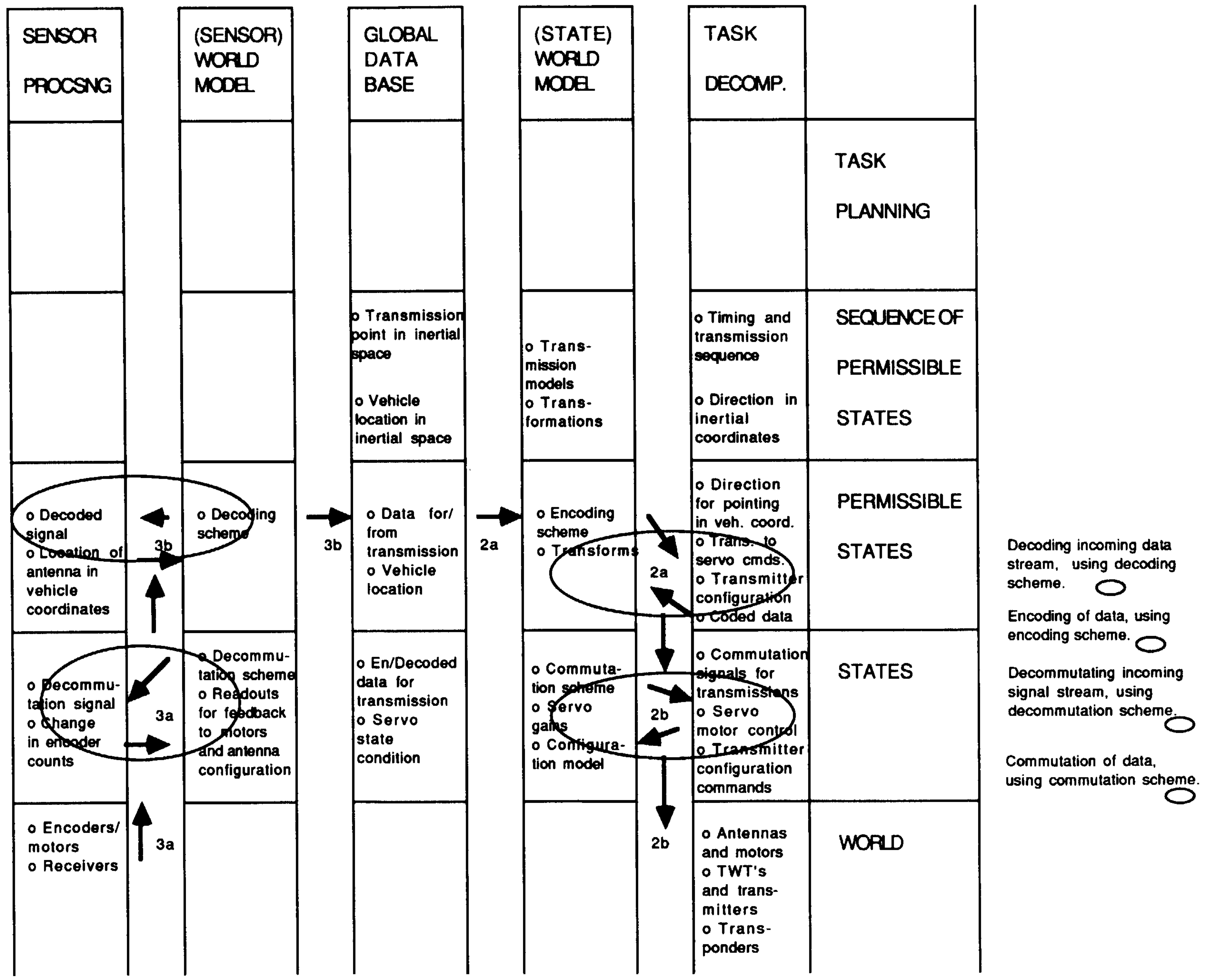
Receives report from Mobility, compares expected resource utilization with that available and gives a go command. ⬭

Receives report on progress/completion of traverse. Logs/monitors. ⬭

Current power state report for use in mobility plan evaluation.

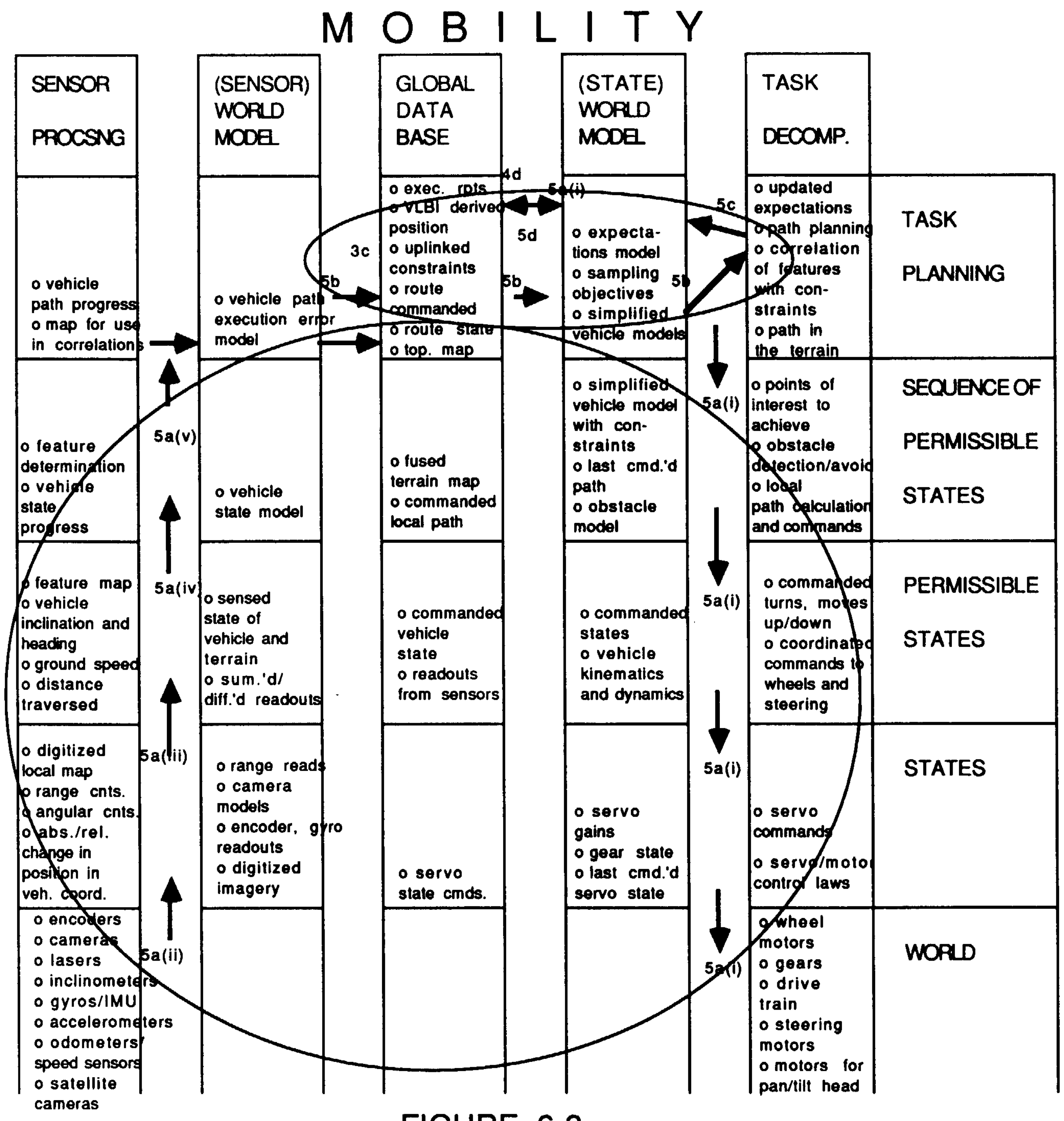
FIGURE 6-1

TELECOMMUNICATIONS



SENSOR PROCSNG
(SENSOR) WORLD MODEL
GLOBAL DATA BASE
(STATE) WORLD MODEL
TASK DECOMP.
TASK PLANNING
o Transmission point in inertial space
o Vehicle location in inertial space
o Trans-mission models
o Trans-formations
o Timing and transmission sequence
o Direction in inertial coordinates
SEQUENCE OF PERMISSIBLE STATES
o Decoded signal
o Location of antenna in vehicle coordinates
o Decoding scheme
o Data for/ from transmission
o Vehicle location
o Encoding scheme
o Transforms
o Direction for pointing in veh. coord.
o Trans. to servo cmds.
o Transmitter configuration
o Coded data
PERMISSIBLE STATES
o Decommu-tation signal
o Change in encoder counts
o Decommu-tation scheme
o Readouts for feedback to motors and antenna configuration
o En/Decoded data for transmission
o Servo state condition
o Commuta-tion scheme
o Servo gains
o Configura-tion model
o Commutation signals for transmissions
o Servo motor control
o Transmitter configuration commands
STATES
o Encoders/ motors
o Receivers
o Antennas and motors
o TWT's and trans-mitters
o Trans-ponders
WORLD
3b
3b
2a
2a
3a
3a
2b
2b
Decoding incoming data stream, using decoding scheme.
Encoding of data, using encoding scheme.
Decommutating incoming signal stream, using decommutation scheme.
Commutation of data, using commutation scheme.

FIGURE 6-2



SENSOR PROCSNG	(SENSOR) WORLD MODEL	GLOBAL DATA BASE	(STATE) WORLD MODEL	TASK DECOMP.	
o vehicle path progress o map for use in correlations	o vehicle path execution error model	o exec. rpts o VLBI derived position o uplinked constraints o route commanded o route state o top. map	o expectations model o sampling objectives o simplified vehicle models	o updated expectations o path planning o correlation of features with constraints o path in the terrain	TASK PLANNING
o feature determination o vehicle state progress	o vehicle state model	o fused terrain map o commanded local path	o simplified vehicle model with constraints o last cmd.'d path o obstacle model	o points of interest to achieve o obstacle detection/avoid o local path calculation and commands	SEQUENCE OF PERMISSIBLE STATES
o feature map o vehicle inclination and heading o ground speed o distance traversed	o sensed state of vehicle and terrain o sum.'d/diff.'d readouts	o commanded vehicle state o readouts from sensors	o commanded states o vehicle kinematics and dynamics	o commanded turns, moves up/down o coordinated commands to wheels and steering	PERMISSIBLE STATES
o digitized local map o range cnts. o angular cnts. o abs./rel. change in position in veh. coord.	o range reads o camera models o encoder, gyro readouts o digitized imagery	o servo state cmds.	o servo gains o gear state o last cmd.'d servo state	o servo commands o servo/motor control laws	STATES
o encoders o cameras o lasers o inclinometers o gyros/IMU o accelerometers o odometers/ speed sensors o satellite cameras				o wheel motors o gears o drive train o steering motors o motors for pan/tilt head	WORLD

Correlation with topographic map

Fused data used in feature determination

Stereo correlated ranges determined

Images and pulsed laser range readouts collected

Receives command to plan for traversal using latest constraint models.

Collects data from cameras and lasers, through commanding a pan/tilt sequence.

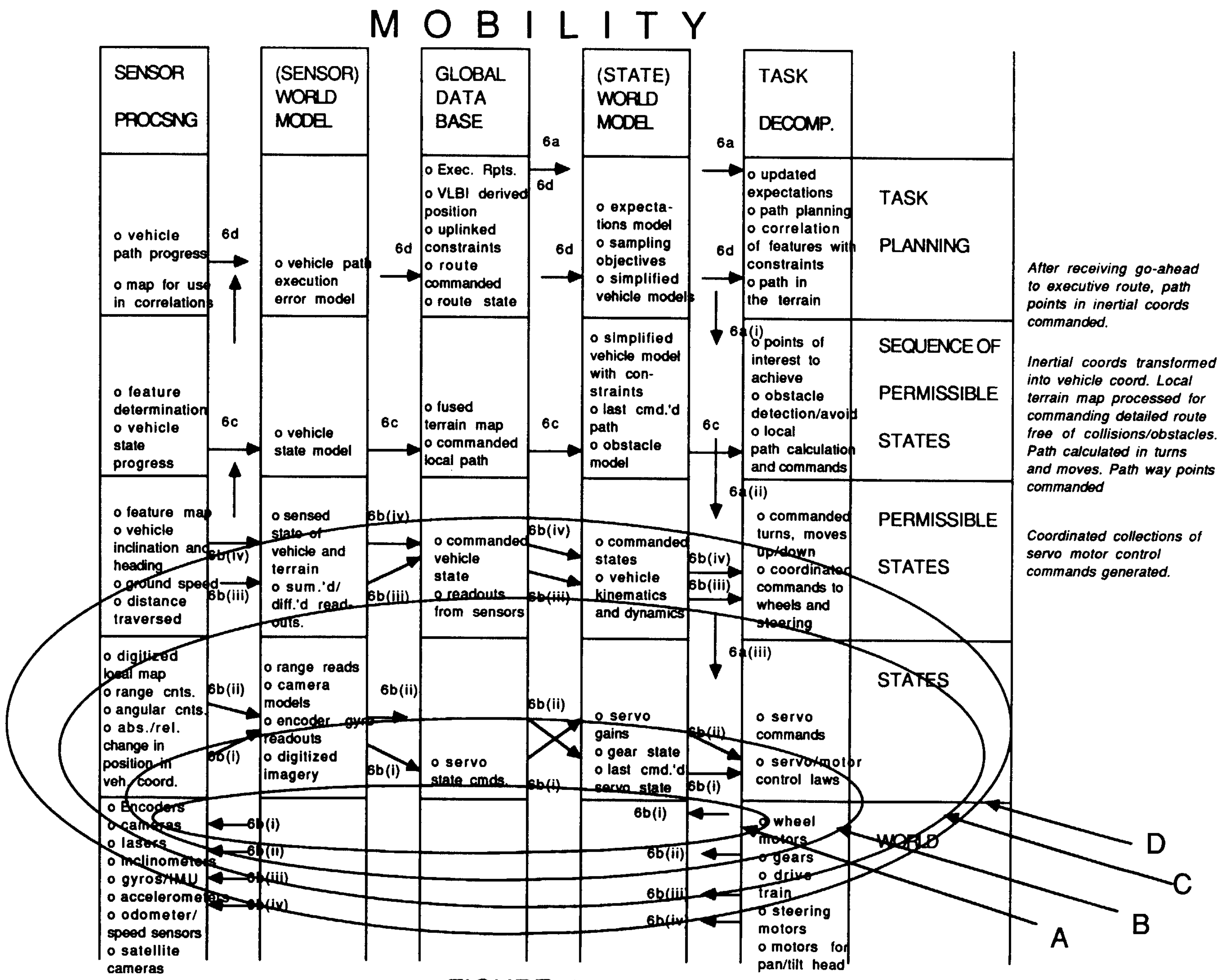
When data is collected, plan a path.

Pan/tilt sequence comd.'d in vehicle coordinates.

Commands in the form of steps to the motors of the pan/tilt head and laser scanning apparatus given

Servo command sequence executed

FIGURE 6-3



After receiving go-ahead to executive route, path points in inertial coords commanded.

Inertial coords transformed into vehicle coord. Local terrain map processed for commanding detailed route free of collisions/obstacles. Path calculated in turns and moves. Path way points commanded

Coordinated collections of servo motor control commands generated.

FIGURE 6-4

6.3 RECEIVE COMMAND

The commanded position and associated data are received through Telecommunications. The data is de-commutated [3a of Figure 6-2], decoded [3b of Figure 6-2] and stored [through the use of the global data base connecting the subsystems] for use in subsequent operations [3c in Figure 6-1 for the command and associated constraints, and 3c in Figure 6-3 for the topographic map and reference location].

6.4 ROUTE PLANNING

The receipt of a command for a traverse to a new location at the Executive [3c of Figure 6-1] begins a planning activity which leads to the generation of a feasible route. The executive calls upon the subsystems [4a of Figure 6-1] to report the current state of resources available for the traverse [4c of Figure 6-1]. These status reports become a part of the constraint space used by the route planning function [4a of Figure 6-1]. A major portion of this function is located in the Mobility subsystem [Figure 6-3]. Here the current state data gathered from the imaging systems on-board the rover are evaluated against the models of the terrain to determine the route to the new location (see 6.4.1 below). Once the route is generated (with an expected resource utilization) a final go/no go evaluation is performed by the executive [4d of Figure 6-1] using the data in any subsystem or other reports received during route planning. A 'go' from the executive to Mobility begins the traverse to the new location.

6.4.1 MOBILITY: ROUTE PLANNING

Route planning begins with the gathering of the latest imaging data of the site surrounding the present position of the rover. A panoramic view of the site is developed through the execution of a sequence of moves of (for example) a pan and tilt mounted camera system on the rover. The precise sequence, based on the current vehicle state, is developed (at the task planning level of Mobility) and commanded for execution [5a(i) of Figure 6-3]. The commands, initially in inertial coordinates, are transformed into specific motor drive commands to the motor mechanisms supporting the camera system [the hierarchical flow of commands through the levels of the Mobility subsystem, labelled 5a(i)]. At each position in the sequence, an image pair (for stereo) is captured and digitized for further processing [5a(ii) of Figure 6-3]. The application of camera models, correlation of multiple images and image processing result in developing range and dimension of specific features [5a(iii) of Figure 6-3]. A correlation of image features to models of terrain features (e.g., boulders, ravines) across several image pairs results in a list of position- and dimension-registered objects (or obstacles) for use in route planning [5a(iv) of Figure 6-3]. A final correlation to an on-board topographic map allows computation of rover position in the terrain and a selection (based on the commanded new position) of the portion of the map for use by the route planner [5a(v) of Figure 6-3]. These results are stored [in the global data base] in the form of a fused map of the route locale of greater resolution than available from the up-linked topographic map alone for use in subsequent route planning.

The route planner generates a path to the goal state (the commanded location of the rover) which satisfies the intermediate view point criteria (if any), the vehicle physical constraints (e.g., clearance, power utilization) and local obstacle avoidance [5b of Figure 6-3]. As part of the verification of the suitability of the path, the movement of the vehicle in the terrain is simulated. In addition, during the simulation expectation models of vehicle performance are generated for use in monitoring the actual traverse of the rover along the path [5c of Figure 6-3]. Only a portion of the planned path is slated for execution, as the errors in planning increase as a function of range. In the case of goal of 10m nominally the entire path can be executed. A report to this effect is generated for final go/no go disposition by the executive [5d of Figure 6-3].

6.5 MOBILITY: PATH TRAVERSE

The planned path is executed in Mobility upon receiving a go ahead command [6a of Figure 6-4]. The planned path in inertial coordinates is transformed into vehicle coordinates [6a(i) of Figure 6-4], specific coordinated moves and turns by the vehicle carriage [6a(ii) of Figure 6-4] and servo commands to the motors [6a(iii) of Figure 6-4].

Each command type represents the output command of different type of control law used in the execution of vehicle motion. At the lowest level [State Level of Mobility] a loop is closed around the servo command using the feedback from motor encoders [6b(i) and the innermost loop A in Figure 6-4]. At the next level a heading/dead reckoning loop is closed around the turn or move commands as measured through the feedback of gyros/IMU or compass [6b(ii) and loop B in Figure 6-4]. A control loop around the distance traversed command is closed through feedback measurements from accelerometers and over-the-ground sensor [6b(iii) and the loop C in Figure 6-4]. A final control loop is centered around the constraint of providing a stable platform during the traverse. The feedback for this loop is provided by inclinometer and integration of the readouts of gyros/IMU [6b(iv) and loop D of Figure 6-4]. Each loop stores feedback data, commands and state estimates for evaluation of vehicle performance during the traverse.

Other processing during the traverse monitors and performs the vehicle state evaluation based on the progress reported by lower levels of the hierarchy. In particular, the progress of the traverse is compared against the model of performance along the path. Corrections in the form of commanded turns and moves ensure compliance to way point and obstacle avoidance constraints [6c of Figure 6-4]. Lastly, progress to the goal state is monitored with periodic reports issued to the executive [6d of Figure 6-4 and Figure 6-1]. A final report of reaching the new location ends the traverse and monitoring loop in the executive.

7. UTILITY AND APPLICATION

The architecture as it is laid out forms the basis of a complete Rover Architecture. When completed it should contain all of the functions both on the ground and in flight. It is at once a complete description of both the control and information flows. Step by step sequences and loops can be worked out for various strategies so the designer can see the interface complexities directly.

Evaluating the rate of execution of these loops can aid in identifying technology alternatives to achieve a greater mission capability. As was discussed at the end of section 4, execution rate is one factor in determining where functions sit in the architecture. If a mission planner wishes to increase the range of the rover, more functions must be 'pushed further down' in the hierarchy or greater processing technology brought to bear to achieve the required performance.

In addition to basic design strategies, detailed fault protection scenarios can be worked out, making sure control loops are continuous and unbroken. Often, in fault protection design, the required elements of the design are difficult to identify. This architecture shows that a fault protection machine must be considered for each state machine in each subsystem. The architecture aids in identifying communication paths (commands and telemetry) needed to achieve the fault protection capability. Tracing scenarios of recovery (as was done in section 5 in brief) reveal the communication paths required.

Finally, by computerizing this model, representing each state machine by the convention $M^k_{ij}(l)$ for each element and including these elements in a data base, control loops can be easily described by strings of execution of elements. Strings repeatedly used in these control loops can be identified and represent the state machines which must be developed and tested first.

8. CONCLUSION

A general architectural concept for a planetary rover is presented, based on an expansion of the NASREM concept for telerobotic control. A mapping of this architecture with the functions for a rover in a MRSR mission has been performed. An example of a mobility scenario executed within this architecture validates the concept. This example and associated discussion illustrate the capability of the architecture to serve as a tool for design and functional trade-off analysis.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the Office of Aeronautics and Space Technology, National Aeronautics and Space Administration.

REFERENCES

1. Wilcox, B.H., Gennery, D.B., Mishkin, A.H., "Mars Rover Local Navigation and Hazard Avoidance," Mobile Robots III, Proceedings of SPIE (to be published).

2. Brooks, R., "A Robot That Walks; Emergent Behaviors from a Carefully Evolved Network," Computation, Volume 1, December 1, Spring, 1989.

3. "Carnegie-Mellon Mars Rover", NASA Review held October 14, 1988.

4. "Mars Rover/Sample Return (MRSR) Rover Mobility and Surface Rendezvous Studies", Final Report, JPL Contract 958073, Martin Marietta Space Systems Company, Denver, Colorado, July, 1988.

5. Albus, J.S., McCain, W.G., Lumia, R., "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," National Bureau of Standards Robot Systems Division, 12/4/86.

6. Kovtunenko, V.M., Kemudzian, A.L., Sukhanov, K.G., "Mars Rover: Fundamental Control Principles," Babakin Engineering and Research Centre, IAF-88-397.

7. Dias, "Revised Rover Surface Operations Scenarios Baseline," JPL IOM (internal document), dated 29 June 1988.

8. "Robotic Vehicle Computer Aided Remote Driving", JPL D-3282 (internal document), June, 1986.

9. Kwok, J.H., "MRSR Reference Missions Summary," Version 23. (internal project document), September 14, 1988.

10. "Final Report: Flight Telerobotic Servicer (FTS) Tinman Concept; In-House Phase B Study", SS-GSFC-0042, Volumes I, II, Goddard Space Flight Center, September 9, 1988.

11. "Functional Requirements for the Telerobotic Testbed," JPL D-3693 (internal document), Revision 3 December, 1988.

N90-29016

THE NASA/OAST TELEROBOT TESTBED ARCHITECTURE

J.R. Matijevic, W.F. Zimmerman and S. Dolinsky

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, M/S 303-308, Pasadena, California 91109

Through a phased development as a laboratory-based research testbed, the NASA/OAST Telerobot Testbed provides an environment for system test and demonstration of the technology which will usefully complement, significantly enhance, or even replace manned space activities. By integrating advanced sensing, robotic manipulation and intelligent control under human-interactive supervision, the Testbed will ultimately demonstrate execution of a variety of generic tasks suggestive of space assembly, maintenance, repair, and telescience. The Testbed system features a hierarchical layered control structure compatible with the incorporation of evolving technologies as they become available. The Testbed system is physically implemented in a computing architecture which allows for ease of integration of these technologies while preserving the flexibility for test of a variety of man-machine modes. This paper reports on the development currently in progress on the functional and implementation architectures of the NASA/OAST Testbed and capabilities planned for the coming years.

1.0 PERSPECTIVE

With the advent of a manned Space Station and renewed Shuttle missions and in response to rising world competition, Congress has mandated the National Aeronautics and Space Administration (NASA) to vigorously develop automation and robotics with the goal of improving productivity in space while lowering overall mission cost, reducing risk to manned space missions, and, in the longer term, transferring robotics technology to industry so as to strengthen its global economic position.

NASA has apportioned each of its centers a role in bringing this mandate to fruition. The Jet Propulsion Laboratory (JPL) has been designated by the Office of Aeronautics and Space Technology (NASA/OAST) to be the lead center for identifying and developing flight qualifiable robotics system technologies through the development of a Telerobot Testbed. Technologies developed at JPL will be transferred to Goddard's Space Flight Center for integration with their Space Station Flight Telerobotic Servicer (FTS) and Shuttle Development Test Flight (DTF-1, DTF-2) arms. This paper describes JPL's ongoing efforts to realize these goals.

1.1 THE NASA/OAST TELEROBOT TESTBED PROJECT - PROJECT OBJECTIVES

The NASA/OAST Telerobot Testbed (TRTB) project is implementing a Telerobot Testbed at JPL for the purpose of developing, integrating, and testing telerobot subsystems and demonstrating new telerobot technologies. As a goal, the Telerobot Testbed seeks to identify and implement system technologies envisioned to be cardinal to flight telerobot systems. Technology research and development is conducted in support of NASA's manned and unmanned space programs and is designed to sustain on-orbit servicing, assembly, inspection and maintenance tasks.

Under the current plan, the Testbed will be upgraded each year to meet technology objectives identified in Reference 1. With time the Testbed is expected to progress to greater levels of machine autonomy. Testbed demonstrations are expected to grow in complexity, duration and automation. Successive years will build upon capabilities of previous years and technologies developed in earlier years will be incorporated into the Testbed permanently.

Technologies currently envisioned for implementation into the Testbed include traded and shared control allowing for enhanced man/machine interaction, Teleoperation with short time delays, autonomous operation in uncertain and cluttered environments, system fault recovery, operation in a dynamic environment, and dexterous manipulation. Testbed deliverables include mature Testbed Interface Specification and Functional Requirements documents, a database of Telerobot system and subsystem performance, and a series of capability demonstrations which provide an indication of the Testbed technologies' maturity and their degree of readiness for transfer to space operations. The TRTB project also expects to deliver a hardware and software database for ground and flight prototype systems which identifies, for the first time, Telerobot system performance criteria, power requirements, computing, data storage and bandwidth requirements, software algorithms for control laws, fallback approaches to task execution, and margins for system growth.

Through the Testbed project, future flight programs will come to understand technical tradeoff issues, understand requirements for qualifying flight Telerobot systems, and benefit from standardized interfaces and modularized hardware and software developed in the Testbed. From its experience the Testbed may grow to become a national resource for validating new space telerobot technology and flight operations sequences.

2.0 THE '89 NASA/OAST TELEROBOT CONCEPTUAL ARCHITECTURE

Conceptually, the NASA/OAST Telerobot Testbed architecture follows a hierarchical design philosophy which places the human and machine intelligences towards the top of the control hierarchy and the primitive or mechanical telerobot functions towards the bottom (Figure 1). Five subsystems, not including the human operator, comprise the Telerobot Testbed system. In descending order on the hierarchy they are: the Operator Control Station (OCS), Task Planning & Reasoning (TPR), Run Time Control (RTC), Sensing and Perception (S&P), Manipulators and Control Mechanization (MCM). Although the Testbed subsystems are physically located in the same facility, an artificial division was introduced between the higher level subsystems (Operator,OCS,TPR) and the lower level subsystems (RTC,MCM,S&P) in anticipation of having to accommodate missions where Operator and manipulators are separated by time delay. Such delays occur whenever the Operator and the worksite are separated by signal propagation time.

The Telerobot (TR) manipulator arms are controlled through one of two possible paths. In teleoperated modes, the Operator commands the manipulators directly through Hand Controllers available to him at the OCS. In autonomous modes, the machine intelligence (TPR) manipulates the arms through RTC. With time the Testbed project expects to fuse the direct path between MCM and the Operator with the autonomous path so that teleoperations, including shared control, will pass down through TPR. For telerobot systems whose local and remote sites are collocated, TPR/RTC will look like a wire connecting MCM and the Operator. Whenever the local and remote sites are separated by distance, the TPR will perform the function of simulating the remote environment so that in effect the Operator teleoperates locally. Task execution at the remote site will occur one delay time later.

The Testbed architecture may also be thought of as being composed of three layers. At the lowest layer is its physical makeup which includes subsystem hardware and software. At the next layer are the operational modes which define the states subsystems take on, and at the top layer are the Telerobot's fundamental capabilities. Capabilities may be defined as a specific configuration of selected subsystem states arranged to focus on a common mission goal. Complex tasks are constructed from these capabilities. These three layers are discussed in greater detail next.

3.0 THE '89 TELEROBOT TESTBED SYSTEM IMPLEMENTATION

Figure 2 is a functional diagram of the Telerobot Testbed as it is currently implemented. Higher level functions are grouped in subsystems toward the top of the hierarchy and lower level functions are grouped in subsystems toward the bottom. The Telerobot architecture is also divided between lower level functions concentrated at the remote site (all subsystems to the right of the Ethernet) and higher level functions concentrated at the local site (all subsystems to the left of the Ethernet). The TRTB project expects to introduce in FY '90 a delay capability into the Testbed to investigate teleop control algorithms with propagation delays between the Operator and manipulators. Testbed subsystems communicate over a common Ethernet local area network. A Network Interface Package software hosted on subsystem VAX computers supports the functions of accepting and transmitting packets of formatted commands or data. A description of the six TRTB subsystems follows:

3.1 OPERATOR CONTROL STATION SUBSYSTEM

The Operator Control Station sits at the top of the Telerobot Hierarchy providing an efficient, user friendly physical interface between the Telerobot Testbed Operator and Test Conductor (TC) and Testbed subsystems. OCS is composed of two work stations, multiple video monitors switchable to different camera or video buffer sources, a stereo vision display, speakers, microphones, three keyboards, a mouse, function switches, two Force Reflecting Hand Controllers (FRHC), and support computers.

The OCS software provides a table-driven system for easy editing or updating of command definition and data. A terminal emulation capability allows the Operator to interface directly through the OCS console with all other Testbed subsystems. Over the Testbed's common Ethernet, OCS accepts and displays information to the Operator or TC from the subsystems, and relays Operator commands back to the subsystems. Through the two Hand Controllers, the Operator teleoperates the two Testbed manipulator arms. Force/torque sensors at the end-effectors backdrive the Hand Controllers, allowing the Operator to "sense" forces and torques induced at the end effectors. Both the Operator and TC have limited voice control command capabilities, including system on/off/halt, camera arm movement, and selected teleop commands. Two cross-strapped "Panic Buttons" interface directly to the manipulator arms providing the Operator and TC with an overriding emergency halt capability.

3.2 TASK PLANNING & REASONING SUBSYSTEM

The Task Planning & Reasoning subsystem sits at the top of the autonomous control hierarchy providing the Telerobot's machine intelligence. TPR performs functions of high level task and gross motion planning. The subsystem interacts with the Operator accepting task assignment, plan changes, plan concurrences, and direct action requests and translates them into processes for RTC execution.

The subsystem consists of a gross motion spatial planner, task planner, a kinematics simulator, and a coordinator to pass knowledge between these reasoning engines. The task planner generates over-all task plans and selects the actions to be performed as appropriate to the current state of objects in the workspace and recently experienced manipulation failures. The gross motion spatial planner generates collision free paths through the workspace for the manipulator arm and carried object. The kinematics simulator conceives possible manipulator arm configurations to reach objects, approach points, or other features of the workspace.

TPR maintains a database of objects (World Model) in the worksite, including their locations/orientations, connectivity, and semantic relationships. During Testbed operations the database is routinely updated from sensor information provided either by S&P and MCM through RTC, or by Operator designation of objects in the workspace. The World Model also incorporates a Collision Detection unit and a geometric reasoner which maintains rules and information trees on relationships between objects in the workspace, logically deduces which changes in the relationships are permissible, and assists the Operator in correcting or completing positional information about objects in the knowledge database.

3.3 RUN TIME CONTROL SUBSYSTEM

The Run Time Control subsystem, together with TPR, provides the Telerobot with the capability to function autonomously. RTC's role is to provide fine motion and grasp planning commands to MCM.

RTC consists of a subsystem System Executive supported by robotics, interface, communications, and infrastructure support modules. Briefly, upon receiving commands from TPR or the Operator/OCS, RTC reformats them into internal RTC data structures, selects a script to match the requested TPR process, selects a path for the arms, kinematically simulates the selected sequence, checking for collisions, pose flips and joint stops, generates local motion and coordination level commands for the manipulator arms and end-effectors, and passes executable macros on to MCM and S&P. During operations RTC monitors sequence execution, evaluating and modifying ongoing actions as needed.

RTC maintains and intermittently updates a database of workspace object locations/orientations based either on information gathered from S&P and MCM or the Operator through TPR. The database maintains accurate geometric and inertial models of the three Telerobot arms and immobile objects in the workspace. A Geometric Relationship Evaluator accomplishes frame transformations and maintains correct connectivity relationships among objects in the workspace.

3.4 SENSING AND PERCEPTION SUBSYSTEM

The Sensing and Perception Subsystem performs four system functions: 1) It provides the Operator on five OCS monitors with live or still, stereo and mono black and white video images of the workspace from nine Testbed cameras and four video frame buffers and provides MCM with object location/orientation state data. Five of the cameras also serve to provide S&P with stereo machine vision; 2) S&P tracks an object in the workspace as it moves about, supplying estimates of the position, orientation, velocity and angular velocity of the object to the other subsystems. S&P's Time Code Generator provides both S&P and MCM with the synchronization signal required to coordinate, for example, machine vision and spinning satellite grappling in real time; 3) When commanded by RTC, S&P performs fixture verification on a stationary object or part of a stationary object in the workspace, supplying machine generated estimates of its position and orientation to RTC and MCM; 4) From a database of objects in the workspace, S&P provides wire-frame models of the objects for display as graphic overlays on OCS monitors. These overlays support the Object Designate and Fixture Verification functions.

3.5 MANIPULATORS AND CONTROL MECHANIZATION SUBSYSTEM

The Manipulator and Control Mechanization subsystem sits at the bottom of the Telerobot hierarchy providing the Telerobot with manipulation capability and the mechanical interface to the workspace. The subsystem consists of two six degree-of-freedom robot arms, actuators, servoed end-effectors, force-torque sensors, universal controllers, the two Force Reflecting Hand Controllers at the Operator console with their attendant electronics, Universal Controllers, a SUN which hosts trajectory generation software, a MicroVax which hosts communications interface software, Macros to enable a variety of Telerobotic actions, and a variety of other support software. MCM also provides and controls a third six degree-of-freedom arm for positioning the stereo vision camera arm.

MCM receives commands from and transmits information back to the Operator through one of two control paths. In teleoperation mode MCM receives position/orientation commands directly from the Operator through the FRHC's and returns force/torque information from the end-effectors. In autonomous modes MCM receives position/orientation and force/torque commands over the Ethernet from RTC and, if in shared control mode, returns position/orientation and force/torque data back to the Hand Controllers. Position/orientation states of objects in the worksite come to MCM from S&P.

4.0 THE NASA/OAST TELEROBOT TESTBED '89 OPERATIONAL SYSTEM CAPABILITIES

In FY'89 five new technology capabilities will be introduced into the Testbed: teleoperation with force reflection, traded control, single and dual arm shared control, Operator designation, and self calibration. These capabilities will augment the Reactive Control and Verification capabilities currently available in the Testbed. These capabilities were conceived as being cardinal or so-called generic in nature allowing complex tasks to be constructed from elementary ones. Shared control permits the human and machine intelligences to work cooperatively while traded control allows them to work sequentially. These capabilities are described next.

4.1 FORCE REFLECTION IN TELEOPERATION

In Teleoperation, the Operator controls the TR's manipulator arms by providing the six position/orientations through the Hand Controllers to MCM. Manipulator path planning, collision avoidance, arm coordination, and object manipulation are performed by the Operator in real time. The force-reflection capability returns force/torque information back through MCM to the Hand Controllers from the robot wrist sensors allowing the Operator to "feel" the force/torques at the end-effector.

4.2 TRADED CONTROL

In the most general sense, traded control is a transfer of control between Operator teleoperation and Telerobot autonomous control anywhere and at any time during task execution. In the TRTB's '89 version of Traded Control the Operator performs all gross motion planning, maneuvering the end-effectors to a point in the proximity of an object and transfers control to the Telerobot for autonomous manipulation of the object. Upon completion of the task the Telerobot moves its end-effector to a point in the vicinity of the object and offers to transfer control back to the Operator. During autonomous execution the Operator may elect to transfer control and continue task execution in teleoperation. Also, the Operator may elect to transfer to autonomous control during fine teleoperation execution. At all times the Operator has overriding control and can elect to Halt a task. MCM's role during traded control is to provide a smooth transition from teleoperation to autonomous control and back to teleoperation as well as the continuous control of arm trajectories through singularities.

4.3 SINGLE AND DUAL ARM SHARED CONTROL

In the most general sense Shared Control allows for manipulator control to be shared jointly between the autonomous Telerobot and the Operator teleoperating force reflecting Hand Controllers. Both single and dual arm shared control have been implemented into the Testbed.

In single arm shared control the Operator selects to control one or more of six possible object positions/orientations through one hand controller and MCM controls the remaining positions/orientations, as well as the six force/torque compliances applied to the object by the end-effector. Force reflection from the end-effector is optional.

The dual arm shared control capability makes possible coordinated dual arm manipulation of rigid objects. The Operator selects to control through one Hand Controller one or more of six possible positions/orientations of an object and the Telerobot controls the remaining positions/orientations as well as all force/torque compliances applied to the object by both arms.

4.4 OPERATOR DESIGNATE

The Operator Designate capability provides wire-frame models (WFM) of objects in TPR's database to the Operator to manually overlay over still camera images of the objects in the workspace on the Operator's OCS console and read out the locations/orientations of the objects. The Operator thus locates objects in the workspace for TPR for subsequent manipulation. Designation can also be used to update the location/orientation of known objects in the workspace, define obstacle regions, and designate generic objects.

4.5 SELF CALIBRATION

Self Calibration is an autonomous capability similar to Verification which provides the Telerobot databases with improved knowledge of an object's location/orientation in the workspace. It improves on the systematic error limitation inherent in Verification by measuring the relative distance between two objects instead of the distance between the objects and the camera.

4.6 REACTIVE CONTROL

Reactive Mode is a capability which enables spinning satellite grapple. S&P provides a continuous updated state vector of the satellite to MCM. MCM then determines arm trajectories required to grapple the rotating satellite.

4.7 VERIFICATION

Verification is an autonomous capability which provides Testbed databases with refined knowledge of the Testbed objects' location/orientation in the workspace. It improves on the error limitation inherent in Operator Designation. A verification is executed only after a Designation.

5.0 THE 1989 NASA/OAST TELEROBOT TESTBED VALIDATION DEMONSTRATION

Telerobot Testbed demonstrations are a synthesis of telerobot technology capabilities, convoked elementary task sequences, and human participation which, when arranged intelligently, engender robust telerobot activity mimicking human activity. They are the deliverables against which the degree of success of attaining TRTB project objectives is measured and against which the worthiness of identified technologies for space applications can be evaluated. Successful demonstration outcomes are a prerequisite to the Testbed technology receiving acceptance for space-based operations on manned and unmanned missions. Technology transfer to a flight project happens once a demonstration proves the technology to be safe and reliable and telerobot risks and performance are well understood.

The task selected for the 1989 Telerobot Testbed technology validation demonstration is an Earth Orbiting System (EOS) Orbital Replacement Unit (ORU) changeout. This demonstration will validate the five new TRTB technologies. In an operation mimicking on-orbit satellite servicing, a tray-looking ORU subtended by a large instrument mockup is exchanged with a smaller instrument mounted on a nearby stowage rack. Two bolts attaching the ORU to the platform are unbolted and, in a dual arm cooperative action, the ORU is detached from the EOS platform and mounted on the rack. The smaller instrument mounted on the stow rack is then removed by a single arm, attached to the EOS platform, and then bolted. Figure 5 depicts the ORU with its accompanying instrument, the stow rack, and the smaller instrument on the rack. Table 1 is a step by step top-level description of the demonstration. The first column lists the EOS tasks while the second identifies the '89 technology capabilities validated. The third column is an attempt to look beyond the demonstration and to identify those capabilities which are generic to flight telerobots--that is, those capabilities which a mature flight telerobot system is envisioned to possess.

The Operator's role in the EOS validation demonstration will be to initiate each task step, select the control modes, designate fixtures in the workspace, and perform all gross arm motions. The Telerobot's role in the EOS validation demonstration will be to visually identify familiar objects in the workspace after a Designation, calibrate the relative positions of objects before a transfer to traded control, perform fine motion planning and arm/tool manipulation, and while in shared control, control selected position/orientations and all force/torques applied to object by the manipulator arm or arms.

6.0 MEASURING TELEROBOT TESTBED SYSTEM PERFORMANCE

The Telerobot Testbed performance will be measured at three levels and evaluated at a fourth. These levels are inclusive of all possible functions for the TRTB. More generally, these levels are valid for other robot architectures and are suggested as a framework from which to evaluate the adequacy of telerobot performance.

At the lowest level of system performance are level 1 subsystem stand-alone tests which validate the hardware and software designs of the telerobot subsystems. These tests seek to verify performance against design requirements, and typically consist of software execution checks, intramodule information transfer, hardware voltages, etc. At level 2, performance tests seek to verify performance against subsystem interface requirements and consist of inter-subsystem compatibility tests between the telerobot subsystems. Level 2 tests typically consist of transmitting and receiving commands correctly between subsystems, properly processing commands, switching among video displays, timing checks, etc. Tests at levels 1 and 2 when they are unsuccessful are typically typified by rework of hardware or software elements.

Ultimately, however, telerobot performance must be measured at the system level. It is here that the telerobot's technology capabilities are tested. Unlike a single purpose tool, thousands of tests can be performed to demonstrate telerobot capability performance. However, if chosen intelligently, a finite number of tests or technology validation demos are sufficient to prove technology capability robustness to perform demonstration tasks and in turn the telerobot's performance limits can be assessed. Of course telerobot work in space will undoubtedly be reduced to a finite number of tasks and those tasks will be specifically checked multiple times in multiple configurations in the Testbed before attempt is made on-orbit. Level 3 system performance, not yet wholly defined, is measured in such terms as tolerance to expected and unexpected changes in the environment, reliability, error recovery, tolerance to measurement errors, stability, and database consistency. When tests at level 3 are unsuccessful or degraded, they are typically corrected by design modification or capability modification/reconceptualization. Limits to performance are assessed through multiple demonstrations with multiple tasks.

At the highest level of test the Telerobot's generic capabilities are validated and its degree of readiness to perform specific space servicing operations is evaluated. No physical tests are made at this level. Rather performance observed during level 3 demonstrations serves to validate the TRTB's readiness to perform multiple space servicing operations. The criteria for evaluating system performance here is to match the Telerobot capabilities against those envisioned for flight missions, including backup operations, redundancy and fault protection in system/operations, delineate limits to the Telerobot's performance, understand its handicaps, and understand risks inherent in its design.

7.0 THE NBS NASREM CONCEPTUAL ARCHITECTURE

The National Bureau of Standards (NBS) has advanced a conceptual telerobot architecture known as the NBS Standard Reference Model as its candidate for space Telerobots. In its most general form NASREM (Figure 3, Reference 7) is partitioned into three hierarchies, each with six vertical levels plus the interface between the robot and the World. As with the NASA/OAST architecture, higher functions are placed towards the top and lower functions towards the bottom. Conceptually, the Operator can interact directly at any level. All modules have access to a Global memory, NASREM`s database. Modules within each hierarchy (vertical data flow) accept commands from higher level modules and transform them into instructions for lower level modules. Across hierarchies (horizontal data flow) modules interact through the World Model with modules in another hierarchy and at any level. The NASREM architecture accommodates growth by adding more levels at the top. For example, NASREM's Service Bay level accommodates one telerobot executing multiple tasks at different sites and the Service Mission level accommodates multiple telerobots operating at multiple jobs at multiple sites.

The TRTB project calls for developing and validating technology which ultimately will be integrated with GSFC's Space Station Flight Telerobotic Servicer (FTS) and Development Test Flight (DTF-1, DTF-2) arms. Since FTS has accepted a requirement to conform to the NASREM telerobot architecture, a mapping was established between the NASA/OAST Telerobot Testbed architecture and the NASREM architecture (see Figure 4, Ref. 6). Roughly, the NASA/OAST Telerobot functions described earlier are reproduced by the first four NASREM levels.

7.1 DIFFERENCES BETWEEN THE NASREM AND THE '89 NASA/OAST TELEROBOT ARCHITECTURES

Comparison of the NASREM and NASA/OAST architectures reveals subtle differences. However, the differences between the two architectures are deemed minor and do not preclude technology transfer from the NASA/OAST Telerobot Testbed to the NASREM FTS. A list of these differences follows:

1) NASA/OAST World Model vs NASREM Global Database

TPR, RTC, and MCM utilize separate, subsystem-specific but consistent data bases whereas NASREM uses one integrated data base. In the NASA/OAST design database, information flows directly and to some extent simultaneously between subsystems while in the NASREM architecture information must flow serially into and out of the Global Database. Thus TRTB subsystems need neither to interrupt other subsystems nor be time-coordinated when accessing database information. Dashed lines in Figure 3 depict data flow which is direct in the JPL Testbed but must pass through the GDB in the NASREM architecture.

2) Time Delay Between Local and Remote Sites

In future developments, the Telerobot Testbed expects to accommodate time delays between the local (Operator/TPR) and remote (RTC/MCM/S&P) subsystems whereas the NASREM architecture does not specifically address this issue. Tests in teleoperations show that deterioration in eye/arm coordination makes it impossible for a human to perform complex tasks with the Telerobot arms whenever the round trip time delay between the Operator and end-effectors is greater than two seconds. The NASA/OAST architecture expects to accommodate teleoperations under such conditions with a more robust TPR than is required without time delay and without a requirement for synchronization between the Operator and the remote manipulator arms. In this concept the Operator interacts with a TPR generated simulation of the manipulators and sensors, rather than with the actual Testbed manipulators. Forces on the end-effectors are predicted based on TPR/RTC's model of the world. The Operator's interactions with the simulation produce commands which are sent to the remote site for execution and the remote site asynchronously returns status messages.

3) Data Base Updates

The NASA/OAST Telerobot Testbed is concerned with paths and tasks in the vicinity of an object in the work-space while NASREM is concerned with activities in the whole workspace. Thus when updating the Testbed databases only subsystems with an interest in the ongoing activity are updated. The update is restricted to information about the local work space only and the Operator is required to be cognizant of activities in the rest of the workspace. In contrast, updating the NASREM Global Data Base requires an extensive run through the entire database, thereby introducing a potential delay in task execution.

4) Operation Modes

The NASA/OAST architecture follows the NASREM philosophy in that the Operator can interact with the telerobot at all levels in the hierarchy. However, NASA has implemented teleoperation, shared control, and traded control modes whereas NASREM is a yet undefined mix of teleoperation and autonomous control.

5) Kinematics

NASREM incorporates knowledge of robot kinematics at the E-move level and lower while the NASA/OAST architecture incorporates it at the TPR level and lower, thereby providing the TRTB with a more robust level-4 capable of increased task planning, task replanning, and path planning.

6) Dynamics

NASREM incorporates knowledge of robot dynamics at the primitive level and lower while the NASA/OAST architecture incorporates it at the RTC level and lower, thereby providing the TRTB with a more robust level-3 capable of increased local path planning and recovery.

8.0 SUMMARY AND CONCLUSIONS

NASA is embarking on a program dedicated to increasing productivity on-orbit while reducing mission costs and risk to astronauts. Its Jet Propulsion Laboratory has been designated as the lead center for identifying and developing flight robotics technologies. JPL is currently implementing a Telerobot Testbed project which seeks to 1) provide a testbed for robotics system integration and technology demonstrations, 2) provide a laboratory or prototype laboratory where flight operations can be evaluated, 3) transfer technology to NASA standard telerobotic arms used on Space Station and STS such as the GSFC FTS and DTF systems, 4) and, for the first time, identify system issues and performance criteria for flight telerobots.

This paper described the TRTB's system architecture (Figures 1, 2) as well as its five new capabilities. Criteria for testing telerobot system performance at the subsystem design level, at the integrated system level, and at the demonstration level, and evaluating its generic capabilities were discussed. The role of demonstrations in the Testbed and the demonstration chosen for the '89 Testbed were described.

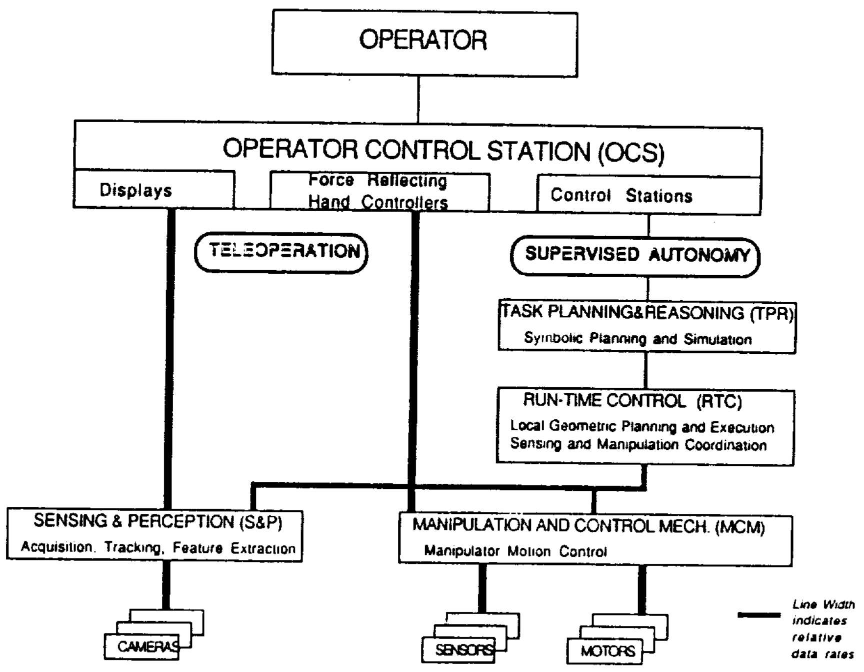
Technology developed and tested in the TRTB will be transferred to GSFC's FTS and DTF arms as candidate technology for implementation. The teleoperated FTS and DTF arms have accepted requirements to conform to the NASREM architectures. Differences between the NASREM and NASA/OAST architectures were identified and for the first time a mapping (Figure 4) between two telerobot architectures was established.

9.0 ACKNOWLEDGEMENTS

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the Office of Aeronautics and Space Technology, National Aeronautics and Space Administration.

REFERENCES

1. JPL D-5692, Final Draft "Telerobotics Project Plan," Aug. 19, 1988 (JPL Internal Document).
2. JPL D-3693 Revision 3, "Functional Requirements Telerobotic Testbed Project," November, 1988 (JPL Internal Document).
3. "Interface Specifications for the 1988 Telerobotic Testbed," W.F. Zimmerman, November, 1988 (JPL Internal Document).
4. "Real-Time Hierarchically Distributed Processing Network Interaction Simulation," W.F. Zimmerman and C. Wu, Proceedings of the 21st Annual Simulation Symposium, pp. 207-226.
5. JPL Interoffice Memorandum 347-88-837, "Results of SUN Network Interface Protocol Performance Study" (JPL Internal Document), W.F. Zimmerman, September 1988.
6. JPL Interoffice Memorandum 343-88-319, Rev. A, "NASREM and the JPL Telerobotics Testbed" (JPL Internal Document), W.O. Keksz, May 26, 1988.
7. "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", J.S. Albus, H.G. McCain, R. Lumia, National Bureau of Standards Robot Systems Division, 12/4/86.



OPERATOR
OPERATOR CONTROL STATION (OCS)
Displays
Force Reflecting Hand Controllers
Control Stations
TELEOPERATION
SUPERVISED AUTONOMY
TASK PLANNING&REASONING (TPR)
Symbolic Planning and Simulation
RUN-TIME CONTROL (RTC)
Local Geometric Planning and Execution
Sensing and Manipulation Coordination
SENSING & PERCEPTION (S&P)
Acquisition, Tracking, Feature Extraction
MANIPULATION AND CONTROL MECH. (MCM)
Manipulator Motion Control
CAMERAS
SENSORS
MOTORS
Line Width indicates relative data rates

FIGURE 1: THE NASA/OAST TELEROBOT TESTBED ARCHITECTURE - '89 CONFIGURATION

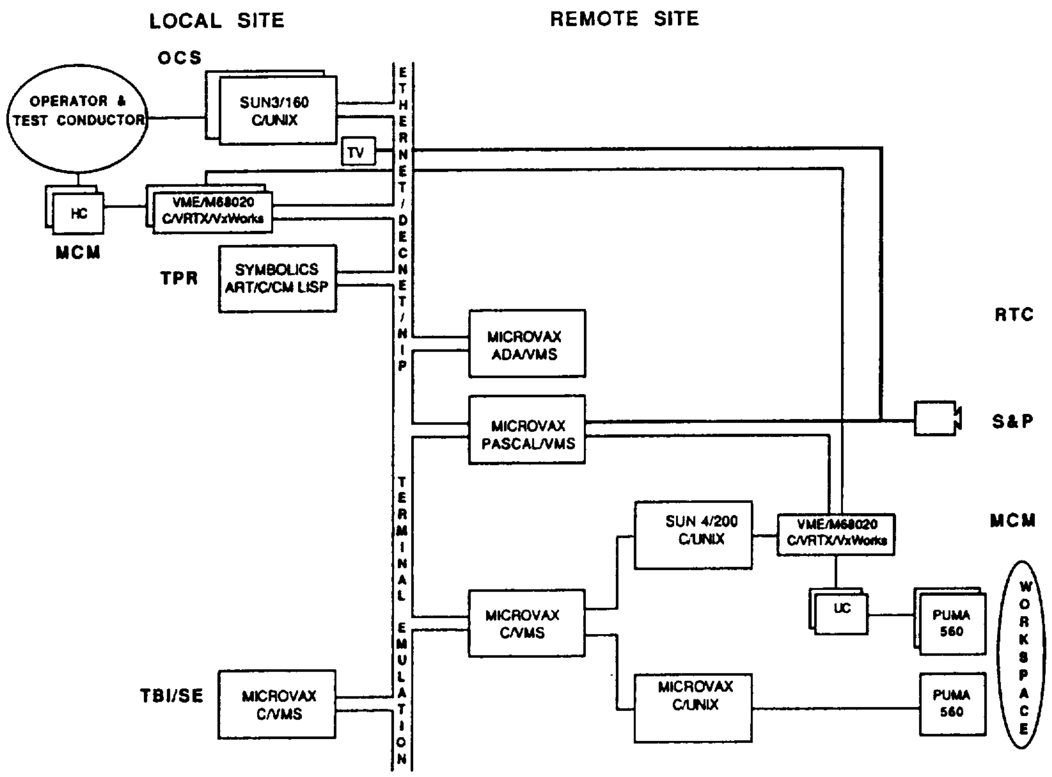


FIGURE 2: THE NASA/OAST TELEROBOT TESTBED IMPLEMENTATION ARCHITECTURE - '89 CONFIGURATION

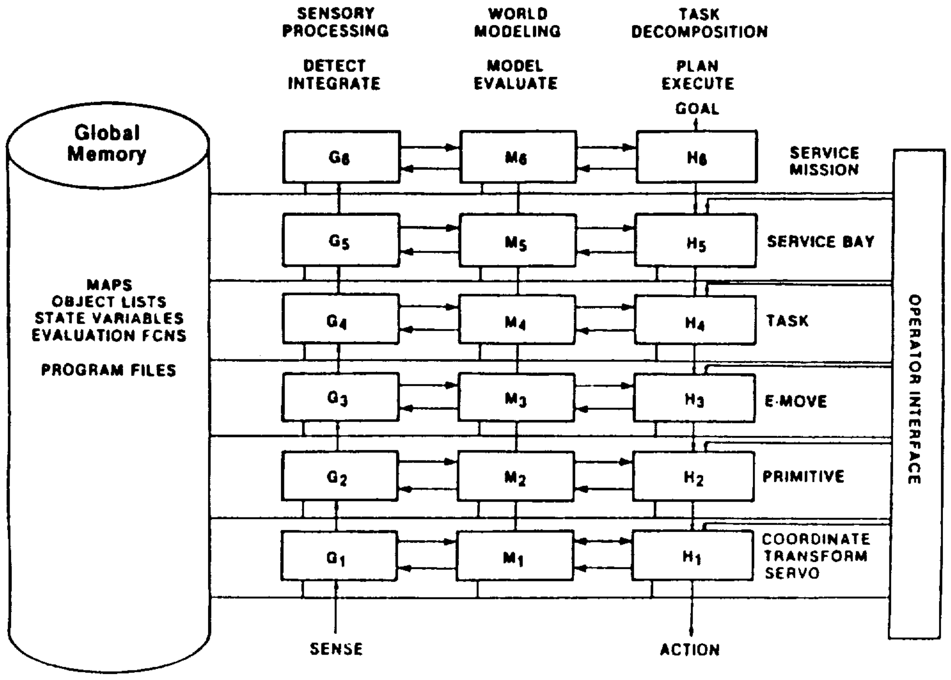


FIGURE 3: THE NBS NASREM STANDARD REFERENCE MODEL

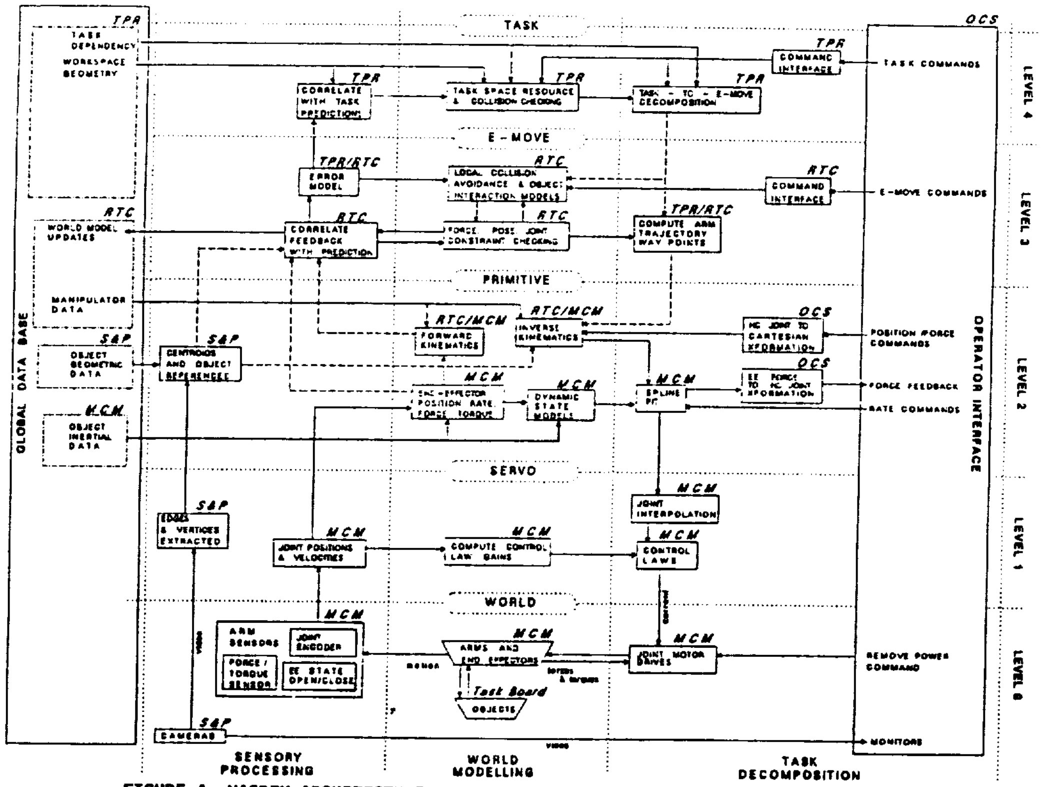


FIGURE 4: NASREM ARCHITECTURE OVERLAID ON THE NASA/OAST TELEROBOT ARCHITECTURE

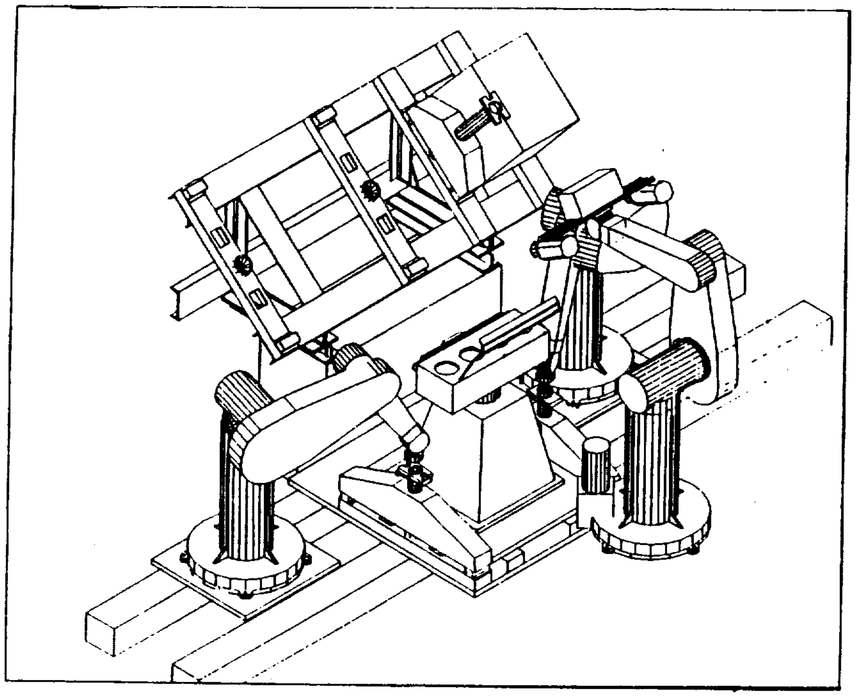


FIGURE 5 THE NASA/OAST TELEROBOT TESTBED EOS PLATFORM SERVICING - 1989 DEMONSTRATION

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE 1: THE '89 TELEROBOT TESTBED VALIDATION DEMONSTRATION

EOS TASKS	'89 CAPABILITIES VALIDATED	GENERIC CAPABILITIES VALIDATED
1 - OPERATOR LOCATES ARMS, TOOLCRIB, STOWBIN, OBJECTS IN WORKSPACE	CAMERA VISION FOR OPERATOR	OPERATOR VISUALLY IDENTIFIES OBJECTS/FEATURES, LOCATIONS/ORIENTATIONS IN THE WORKSPACE
2 - IDENTIFY FOR TELEROBOT ARMS, TOOLCRIB, STOWBIN, TASKBOARD, BOLTS ON ORU/INSTR.	OCS DESIGNATE FUNCTION; DATABASES UPDATED	TELEROBOT VISUALLY IDENTIFIES OBJECTS/FEATURES, LOCATIONS/ORIENTATIONS IN THE WORKSPACE; COMMUNICATIONS BETWEEN TELEROBOTIC SUBSYSTEMS AND OPERATOR; UPDATES DATABASE
3 - MOVE ARM-1 TO VICINITY OF TOOLCRIB	TELEOPERATIONS (SINGLE ARM, GROSS ARM MOTIONS)	TELEOPERATION (SINGLE ARM, GROSS MOTIONS)
4 - GET WRENCH	TRADED CONTROL - TRANSITION FROM/TO TELEOPERATIONS TO/FROM AUTONOMOUS CONTROL	AUTONOMOUS FINE MOTION PATH PLANNING, ATTACHING TOOL TO END-EFFECTOR
5 - MOVE ARM-1 TO VICINITY OF ORU	SAME AS (3)	SAME AS (3)
6 - REMOVE BOLT-1 ATTACHING ORU TO EOS PLATFORM	TRADED CONTROL;SELF-CALIBRATION	AUTONOMOUS FINE MOTION PATH PLANNING, ENGAGE BOLT, TWIST BOLT OFF, UPDATE DATABASE
7 - MOVE ARM-1 TO VICINITY TOOLCRIB	SAME AS (3)	SAME AS (3)
8 - RETURN TOOL TO TOOLCRIB	TRADED CONTROL	AUTONOMOUS FINE MOTION PATH PLANNING, RELEASE TOOL TO TOOLCRIB
9 - REPEAT PROCEDURE WITH ARM-2	SAME AS (3) THROUGH (8)	CAPABILITY TO OPERATE TWO ARMS INDIVIDUALLY
10 - MOVE ARM-1 AND ARM-2 TO VICINITY ORU	TELEOPERATIONS (TWO ARMS, GROSS ARM MOTIONS)	TELEOPERATION (DUAL ARM, GROSS MOTION)
11 - GRASP HANDLE-1 ON ORU	TRADED CONTROL; SELF-CALIBRATION	AUTONOMOUS FINE MOTION PATH PLANNING. RETURN TO OPERATOR IF UNRESOLVABLE ERROR OCCURS, GRASP HANDLE WITH FORCE CONSTRAINT, UPDATE DATABASE.
12 - GRASP HANDLE-2 ON ORU	TRADED CONTROL; SELF-CALIBRATION	SAME AS (11)
13 -DETACH ORU FROM EOS PLATFORM	DUAL-ARM SHARED CONTROL - REMOVE OBJECT ATTACHED BY TWO PINS. OPERATOR PROVIDES POSITION/ORIENTATION OF OBJECT AND TELEROBOT AUTONOMOUSLY PROVIDES POSITIONS/ORIENTATIONS AND FORCE/TORQUE COMPLIANCE APPLIED TO ORU BY BOTH ARMS.	TELEROBOT REMOVES OBJECT ATTACHED BY TWO PINS WITH COORDINATED DUAL-ARM ACTION. MAINTAINING POSITION/ORIENTATION. FORCE/TORQUE CONTROL. OPERATOR PROVIDES PATH PLANNING AND SENSES FORCE/TORQUES.
14 - MOVE ORU TO STOW RACK	SAME AS (13) - MANIPULATE OBJECT	TELEROBOT MANIPULATES OBJECT WITH COORDINATED DUAL-ARM ACTION. MAINTAINING POSITION/ORIENTATION, FORCE/TORQUE CONTROL. OPERATOR PROVIDES PATH-PLANNING.
15 - ATTACH ORU TO STOW RACK	SAME AS (13) - TWO PIN INSERTION	TELEROBOT INSERTS OBJECT (TWO-PIN INSERTION) WITH COORDINATED DUAL-ARM ACTION. MAINTAINING POSITION/ORIENTATION. FORCE/TORQUE CONTROL. UPDATES DATABASE. OPERATOR PROVIDES PATH PLANNING AND SENSES FORCE/TORQUES.
16 - UNGRASP ORU HANDLES	TELEOPERATIONS (UNGRASP)	TELEOPERATION (UNGRASP COMMAND)
17 - MOVE ONE ARM TO VICINITY OF SMALL INSTRUMENT	SAME AS (3)	SAME AS (3)
18 - GRASP HANDLE ON SMALL INSTRUMENT	SAME AS (11)	SAME AS (11)
19 - DETACH SMALL INSTRUMENT FROM STOW RACK	SINGLE-ARM SHARED CONTROL - REMOVE OBJECT ATTACHED BY ONE PIN. OPERATOR PROVIDES POSITIONS/ORIENTATIONS FOR ARM AND TELEROBOT PROVIDES FORCE/TORQUE COMPLIANCE APPLIED TO INSTRUMENT. FORCE REFLECTION ENABLES OPERATOR TO FEEL FORCES AT END-EFFECTOR.	TELEROBOT REMOVES OBJECT (ONE PIN) WITH SINGLE-ARM ACTION. MAINTAINING POSITION/ORIENTATION, FORCE/TORQUE CONTROL. OPERATOR PROVIDES PATH PLANNING AND SENSES FORCE/TORQUES INDUCED ON OBJECT.
20 - MOVE SMALL INSTRUMENT TO VICINITY OF EOS PLATFORM	SAME AS (19) - MANIPULATE OBJECT WITH FORCE FEEDBACK	SAME AS (3)
21 - ATTACH SMALL INSTRUMENT TO PLATFORM	SAME AS (19) - SINGLE PIN INSERTION	SAME AS (19) - SINGLE PIN INSERTION; SELF CALIBRATION
22 - MOVE ARM TO VICINITY OF TOOLCRIB	SAME AS (3)	SAME AS (3)
23 - GET WRENCH	SAME AS (4)	SAME AS (4)
24 - MOVE ARM TO VICINITY OF SMALL INSTRUMENT	SAME AS (3)	SAME AS (3)
25 - BOLT SMALL INSTRUMENT TO PLATFORM	TRADED CONTROL; SELF-CALIBRATION	AUTONOMOUS FINE MOTION PLANNING, ENGAGE BOLT, TWIST BOLT ON, UPDATE DATABASE.
26 - RETURN TOOL TO TOOLCRIB	SAME AS (8)	SAME AS (8)

N90-29017

FORMULATION OF DESIGN GUIDELINES FOR AUTOMATED ROBOTIC ASSEMBLY IN OUTERSPACE

by

Dr. Suren N. Dwivedi
Professor in Mechanical & Aerospace Engr.
West Virginia Univ.
Morgantown, WV 26506

Mr. Gary Jones
GSFC/NASA
Greenbelt, MD 20770

S. Banerjee,
Dr. S. Srivastava
Professor in Mathematics & Computer Science
Bowie State University
Bowie, MD

1. ABSTRACT

In this paper, we illustrate the approach for arriving at design guidelines for assembly by robots in outerspace. The use of robots in a zero gravity environment necessitates that extra factors over and above normal design guidelines be taken into account. Besides, many of the guidelines for assembly by robots on earth do not apply in space. However, considering the axioms for normal design and assembly as one set, guidelines for design and robotic assembly as another, and guidelines for design and assembly in space as the third set, unions and intersections of these sets can generate guidelines for two or more of these conditions taken together - say design and manual assembly in space. Therein lies the potential to develop expert systems in the future, which would use an exhaustive database and similar guidelines to arrive at those required by a superposition of these conditions.

2. INTRODUCTION

In view of the ambitious plans afoot in this country to launch the world's first permanently manned space station, automated robotic assembly in space takes on a whole new significance. Earlier, astronauts have successfully assembled and serviced critical components in space. However, this exposes human beings to the hazardous space environment. In the long-term, it is expected that full automation of the space station, will fullfil all possible objectives without the loss of human lives. At present however, it is intended to restrict, as long as possible, the activities of the astronauts to only those tasks which are impossible without human intelligence and decision-making ability.

Typically, the Permanently Manned Space Station (PMSS) would be involved in collecting data from remote heavenly bodies, servicing and maintenance of satellites, and experimental production of extremely pure pharmacological products. Progressive automation of these activities will probably eliminate all human involvement.

Current robotic technology is limited to conveying out repetitive tasks (for which the robotic systems have to be programmed beforehand) or having the system undertake a series of actions (for which an operator would have to control it at each step). Though research is underway at Goddard Space Flight Center (GSFC) to build capabilities into the robotic systems so that they could independently decide the course of actions once given a task, very little effort has been made to devise and formulate the special design requirements for components that such robotically operated tasks in space would demand (2).

The area of formulating design guidelines for robotic servicing/assembly in space is one of the key spheres that is expected to emerge as the focus of concentrated study and research efforts in the coming years. This is because of the peculiar demands that the absence of gravity and other factors (like the desire to avoid building complicated, unnecessary capabilities into the robot) would place on the design.

In this paper, we propose to formulate some of these ideas into a set of rules that could be used as guidelines while designing components to be assembled.

By implementing these rules, it is expected that it would be possible to achieve increased efficiency, lower costs and reduced human exposure to space hazards, requiring no EVA (Extra Vehicular Activity).

3. SET OF GENERAL GUIDELINES FOR DESIGN AND ASSEMBLY ON EARTH

Before we discuss the set of guidelines for assembly a) in space and b) by robotic system, we should review normal design and assembly guidelines as they are used in the various engineering industries.

It has been proposed by Suh, Bell et al. (3) that design should be guided by certain axioms which cannot be violated. It is expected that all design and assembly rules should be derivable from one or more of these axioms and thus a large data base would eventually enable the development of a mechanized design algorithm (4) or even a Design and Manufacturing Advisor (5) that would transform the perceived need into a numerically expressed set of functional requirements, evaluate alternative design concepts, and select final design and production process.

In view of these trends, it is advisable to formulate guidelines for all kinds of applications in designing for assembly. This is so that A.I techniques could be taken advantage of to analyze and synthesize these rules to suit the particular situation.

Some of the general axioms put forward by Suh, Bell et al. (3) are as follows:

1) All functional requirements and constraints should be kept at the barest minimum level.

2) Functional requirements should be satisfied in their order of importance.

3) Information content is to be minimized (Instructions regarding locating, processing etc should be kept to a minimum and tolerances, surface finish, etc. should be as relaxed as possible).

4) If some functional requirements in the proposed design can be satisfied independently of each other, they should be integrated in a single part.

5) As little material should be used as possible.

Many of the heuristics as well are well known, formalized techniques in manufacturing and assembly like group technology, value engineering etc. involve direct corollaries of these axioms. As far as assembly on earth (with or without robotic systems) are concerned, some more illustrated guidelines follow:

6) Symmetry should be maintained as far as possible.

7) Parts should be standardized whenever possible.

8) Commonality of parts facilitates assembly.

9) Mutual interference of parts is to be avoided. Springs must have closed loops, diameter of the spring wire should be greater than the spacing between coils.

10) Gravity should be taken advantage of for locating, feeding and damping whenever possible. Figure 1 shows how the front weight of a tractor (provided to ensure stability) is screwed to the front rigid axle without any mechanical fastener, using the force of gravity only.

11) Bottom-up assembly should be used - i.e., the heaviest part should be at the bottom, the next heaviest part on it, and so on.

GUIDELINES FOR ROBOTIC ASSEMBLY IN SPACE OR ON EARTH

It is very important to make sure that no unnecessary features are built into a design because this would stretch the capabilities of the robot, reduce the reliability of the system, increase cost and decrease efficiency. In other words, design should be such that even the simplest of robots using the minimum number and type of motor actions and sensory features can assemble or service the items in space. Other guidelines are as follows:

1) Minimize the number of parts.

2) Break up the assembly into modules that can be easily assembled or disassembled. This would minimize the effort involved in servicing.

3) Use a sequential or layered approach to assembly.

Gripping the items present difficulties on earth as well as in space. However, in space these difficulties must be resolved fully as the consequences of improper gripping can be extremely costly or even fatal. Any component that slips out of grip can become a projected missile and fatally injure the occupants of the spacecraft, or damage the spacecraft and its equipment. Therefore rule 4 is formulated as follows:

4) All movements of components must be secure, verifiable and failsafe. Handles or special areas are to be provided for fail safe gripping whenever possible.

5) Unidirectional assembly is to be attempted. This would reduce unnecessary motions on the part of the robot.

6) Maximize commonality of parts and minimize product variations.

7) Eliminate electrical cables.

Experience has shown that electrical cables are extremely difficult to assemble by robots unless they are in the most elementary form. In fact any flexible part creates the same problems. Absence of electrical cables and other flexible parts would alleinate a major problem area in assembly.

8) If the robotic system uses vision equipment, then shiny surfaces should be avoided that blind vision by causing problems with the camera.

9) Use gravity whenever possible to locate parts. For example, the heaviest part could be used as a fixture.

10) Use gravity-fastening methods.

SET OF GUIDELINES FOR DESIGN AND ASSEMBLY IN SPACE

As mentioned earlier, the absence of gravity is the single most important factor that calls for major changes in the design measurements vis-a-vis assembly on earth.

Gravity acts as a gripping, holding, or locating force on earth. This is something that is normally taken for granted, for in the absence of gravity, a major constraining force is lost. This means that bottom-up assembly is more preferable to any other orientation. Therefore;

1) Orientation of components can be in any direction. However, components should be so designed that during assembly (if they are non-symmetric) they should be capable of being assembled from any one particular side.

On earth gripper safety is considered more important than that of the object because of the high cost of grippers. In case of a slip, the object is dropped to save the gripper. However, in space any object that slips out of the robot's grip can become a projected missile and severely damage the spacecraft and equipment or even fatally injure the occupants, hence

2) All components should have handles or gripping facilities that prevent their slipping out of grip.

If a component is capable of reconfiguring itself the need to reassemble it is eliminated. The chance of the component slipping out of control, need for unnecessary gripping, extra motion, etc are deviated and therefore, safety and efficiency in space operations are enhanced. This translates itself into the guideline:

3) Components should be able to reconfigurate themselves whenever feasible. An

example of this would be a solar panel which can fold up, instruments that can realign themselves etc.

4) Use of symmetric parts that require no particular orientation.

5) Avoid parts with tangling tendencies.

6) Gravity fastening can not be used.

In figure 1 the component (front weight of a tractor for providing stability) uses gravity fastening. However, such fastening would not constrain the component in space.
Some other guidelines formulated by Gordon, S.A., (4) are as follows:

7) All components have to be constrained fully.

8) Servicing operations using two gripping/moving locations should be permissible.

9) Fasteners should be constrained from all sides (captive fasteners)

10) Hardware used must be the ones that are already in use and standardized.

Guideline number (7) is quite self explanatory. If the components are not constrained fully and securely, they would float away from control due to the absence of gravity. While the components are in the robot's grasp, they must not even be able to shift position as thus would disrupt assembly/servicing operations.

The eighth guideline has been formulated by GSFC in order to make the design of the end effector of the robot arm simpler and more compact as the task of gripping and operating the restraint is now shared by two separate arms. In the case of only one arm, the end effector has to perform both gripping and operating the restraint. Also, in this case the torque is reached back through the robot arm or end effector and this calls for the presence of gripping fixtures at each fastener location. In two arm operation, only one gripping fixture is needed to constrain the component.

Figure 2 (Gordon, S.A., [4]) shows the advantages of using a captive fastener. The only task that the robot has to carry out for assembly / disassembly is to apply torque to the fastener. If the fasteners are not captive, the robot has to perform the additional task of (a) restraining the fastener while operating it and (b) removing it and using a separate fixture to store the fastener.

Thus, a simple design modification considerably reduces the complexity of and the number of operations needed in the task.

The use of standardized hardware eliminates the need for research to develop special hardware and the whole host of design modifications that accommodating such special hardware would entail.

In the above referred research work, a mock-up of a unit of a spacecraft was used to study the effect of the special requirements of the robotic servicing in space on the design and the guidelines discussed (7 to 10) were found to be major influencing factors.

AN ELEMENTARY APPROACH FOR ARRIVING AT GUIDELINES THAT ARE SUBSETS / SUPERSETS OF THESE THREE SETS

Figure 3 explains the position of robotic assembly in space vis-a-vis assembly on earth (by robots or otherwise) using set theory.

Mathematically, S_E denotes the set of design assembly guidelines on earth (using robots or otherwise). S_R denotes the set of design assembly guidelines by robots (on earth or in space). S_S denotes the set of design assembly techniques in space (by robots or by astronauts). Then,

$S_E \cap S_R = S_{RE}$, set of guidelines for design and assembly by robots on earth.

$S_E \cap S_S = S_{ES}$, set of design and assembly guidelines suitable on earth and in space, using robots or otherwise.

$S_R \cap S_S = S_{RS}$, set of design techniques for assembly by robots in space.

$S_R \cap S_S \quad S_E = S_{RSE}$, set of guidelines for design and assembly by robots that can be used in space and on earth.

Thus, in order to arrive at S_{RS}, ie, the set of all guidelines for robotic design and assembly in space, we need only find $S_R \cap S_S$. In this case, it can be done by inspection. The only guidelines of S_R that do not apply are those pertaining to the use of gravity, ie,

9) Use gravity whenever possible to locate parts

10) Use gravity-fastening methods, by eliminating these two guidelines we arrive at S_{RS}.

CONCLUSION

Due to the energizing trends of using A-I techniques to design components and assemblies for different kinds of applications, it is important to formulate guidelines that could be used in the data base that such techniques would need. It must be emphasized that these guidelines are far from exhaustive and have been used for illustrative purposes only. It is hoped that at a later time it would be possible to develop expert systems that would be able to arrive at guidelines which can be formed from a combination (union or intersection) of these three sets from an exhaustive data base. Examples of such subsets would be guidelines for design and assembly on earth by robots, guidelines for design and assembly in space by astronauts (This would be important as a back up in case of a failure of the robotic systems), etc. With almost complete computerization, these tools would prove indespensible to several applications.

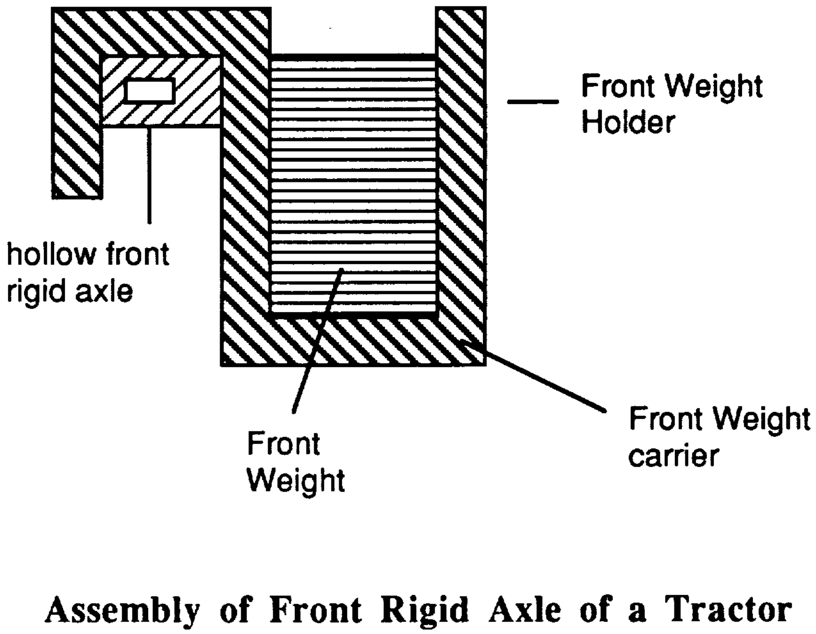


Fig 1. Assembly of Front Rigid Axle of a Tractor

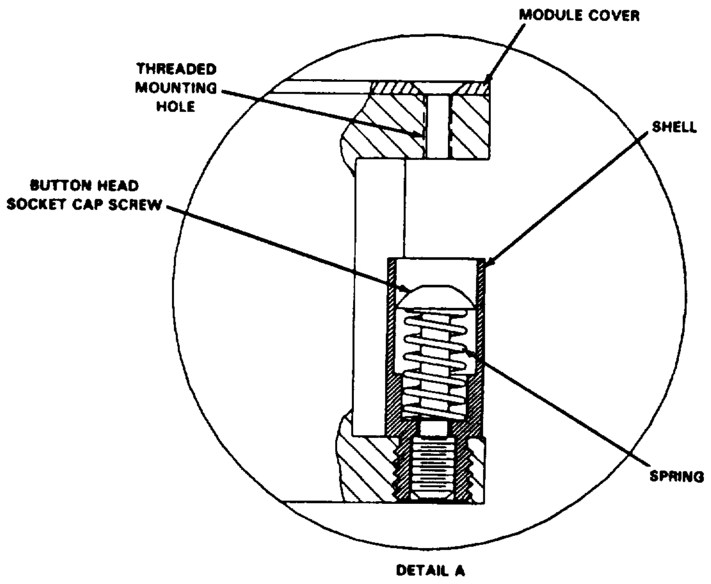


Fig. 2. Use of a Captive Fastener

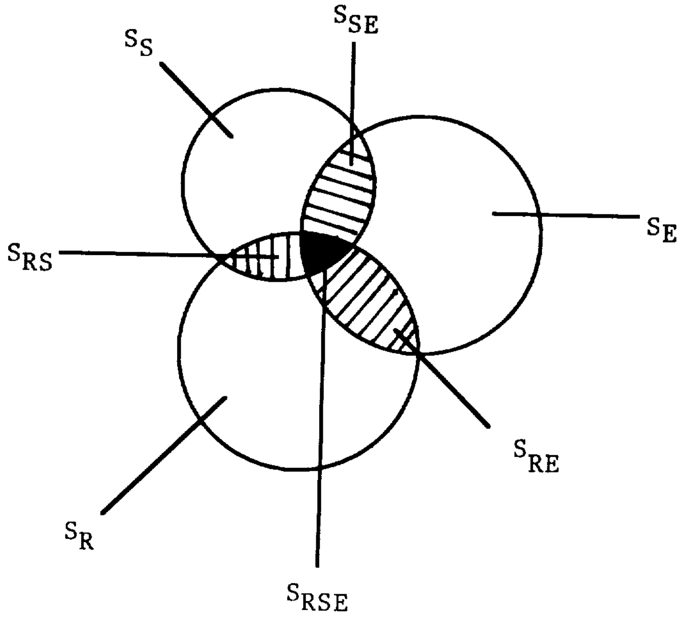


Fig. 3. Robotic Assembly in Space Vis-A-Vis Assembly on Earth.

REFERENCES

1. Boothroyd, G., Design for Assembly Handbook, University of Massachusetts.
2. Suh, N.P., Bell, A.C., Gosssard, D.C., "On an Axiomatic Approach to Manufacturing and Manufacturing Systems", Journal of Engineering for Industry, Vol. 100, May 1978.
3. Suh, N.P., Kim, H.S., "On a Consultive Expert System for Design Axiomatics", Presented at Intelligent Manufacturing Systems, An International Conference, Budapest, Hungary, June 1986.
4. Gordon, A. Scott, "Design Guidelines for Robotically Serviceable Hardware", NASA Technical Memorandum 100700, GSFG 1988.
5. Starkey, John M., and Florin, Gregory J., "Design for Manufacturability", Journal of the ASME, DET-121, 1986.
6. Ho, C., "On Comparison of New Methodologies Applied to Manufacturing Design", Presented at the SME Seventh North American Metalworking Research Conference, May 1979.
7. Dwivedi, S.N., "Guidelines and Rules for Design of Products to be Assembled by Robots", Proceedings of Japan-U.S.A. Symposium on Flexible Automation, Osaka, July 14-18, 1986.
8. Dwivedi, S.N. and Klein, B.R., "Design for Manufacturability-Makes Dollars and Sense," Journal of CIM Review- A Journal of Manufacturing and Management, Vol. 2, No. 3, pp. 53-59, 1986.
9. Dwivedi, S.N. and Mahalingam, S., "Guidelines for Design for Manufacturability and Automation," Proceedings of International Congress on Technology and Technology Exchange, Pittsburgh, October 6-8, 1986.
10. Dwivedi, S.N., "A Fully Automated System for Electronic Assembly and Testing," CIM Review - The Journal of Computer Integrated Manufacturing Management, Vol. 2, No. 4, 1986.

N90-29018

AUTOMATION AND ROBOTICS TECHNOLOGY FOR INTELLIGENT MINING SYSTEMS

JEFFREY H. WELSH

Pittsburgh Research Center
Bureau of Mines
U.S. Department of the Interior
Pittsburgh, PA, USA

ABSTRACT

The U.S. Bureau of Mines is approaching the problems of accidents and efficiency in the mining industry through the application of automation and robotics to mining systems. This technology can increase safety by removing workers from hazardous areas of the mines or from performing hazardous tasks. The efficiency of mining systems can increase through a reduction of machine downtime and through more efficient operation of mining equipment. The short-term goal of the Automation and Robotics program is to develop technology that can be implemented in the form of an autonomous mining machine using current continuous mining machine equipment. This requires technology that would allow a continuous mining machine to perform the same functions in coal extraction as a manually-operated continuous mining machine, only without human intervention. In the longer term, the goal is to conduct research that will lead to new intelligent mining systems that capitalize on the capabilities of robotics.

The Bureau of Mines Automation and Robotics program has been structured to produce the technology required for the short- and long-term goals. The short-term goal of application of automation and robotics to an existing mining machine, resulting in autonomous operation, is expected to be accomplished within five years. Key technology elements required for an autonomous continuous mining machine are well underway and include machine navigation systems, coal-rock interface detectors, machine condition monitoring, and intelligent computer systems.

A navigation scheme consisting of a laser scanning unit, a gyroscope, sonar, and clinometers has been designed, sensors procured, and lab testing initiated. For coal-rock interface detection, Bureau work is focusing on the techniques of machine vibration, in-seam seismic properties, and doppler radar. In-mine testing of these techniques is underway. The Bureau is approaching machine failures through the development of real-time sensor-based diagnostic expert systems. An expert system for the hydraulic and electrical subsystems of a continuous mining machine is being developed. The last key technology element, an intelligent computer system, provides the backbone for intelligent mining systems. The computer system must interface to a variety of sensor systems, and control the mining machine to mine coal according to a mining plan, while being able to react to any abnormal conditions encountered. An onboard computer system has been designed and machine control tests have been initiated.

In this paper, the Bureau of Mines program is described, including status of key technology elements for an autonomous continuous mining machine, the program schedule, and future work. Although the program is directed toward underground mining, much of the technology being developed may have applications for space systems or mining on the moon or other planets.

1. INTRODUCTION

Coal is the most abundant U.S. energy resource. There are enough coal reserves in the U.S. to provide energy for several hundred years. However, a number of factors have contributed to a situation where the U.S. coal industry is losing its competitiveness in the world energy market. Two contributing factors are the costs associated with accidents and system inefficiencies. If coal is to be the solution for U.S. energy needs in the future, something must be done to increase safety, health, and efficiency of mining operations.

Many new technologies have been introduced to mining over the years. Continuous mining machine and longwall coal extraction systems have had a large impact on coal productivity. However, these systems, even though productive, still operate far below their design capacity. Also, the number of accidents occurring annually in underground coal mines has basically leveled off, with no major reduction expected using current mining systems.

The application of high technology including artificial intelligence and robotics is one way for the U.S. coal industry to be competitive, and for coal to be the energy resource of the future. Robotics and automation technology are now commonly used in other industries to increase safety and productivity, and to reduce costs. The automotive industry is one example in which this technology is used with success. A similar success can be achieved in the mining industry. Productivity should be improved by increasing machine availability and efficiency of operation. Health and safety should be improved by removing humans from hazardous areas of the mine.

The Bureau of Mines is committed to a program of research in Robotics and Automation for the mining industry. This paper presents the Bureau's program, discussing key technology areas, current status, and plans for the future.

2. PROGRAM DIRECTION

Room-and-pillar mining using continuous mining machines is a major coal extraction method in the U.S. today and is expected to continue to be a major method. Continuous mining machines are also necessary for panel development for the longwall coal extraction method. Therefore, the Bureau of Mines has initially focused its Robotics and Automation program toward the technology required for an autonomous or robotic continuous mining machine--that is, toward technology that would allow a continuous mining machine to operate without human intervention in a room-and-pillar mining scenario.

This research involves integrating new technology with existing continuous mining machines to allow them to operate autonomously. The new technology consists of sensor systems and computer intelligence that would allow the mining machine to sense its own status and operation, sense its environment, make decisions, and extract coal according to a mining plan, while being able to react to changing conditions. While working toward the program goal of a completely autonomous continuous mining machine, much of this technology can also be utilized on existing manually-operated mining machines to have an immediate impact on mining safety and productivity.

3. TECHNOLOGY ELEMENTS

An autonomous continuous mining machine must be capable of performing the same functions in coal extraction as a manually-operated continuous miner, only without human intervention. With a manually-operated continuous mining machine, the operator relies heavily on his human sensory systems to control the machine according to a mining plan (fig. 1). The operator often uses vision to tell when the cutting drum is cutting coal or the overlying or underlying strata. The feel or vibration of the machine and the noise generated by the cutting are also indicators of the material being cut. As the machine goes from cutting coal to a harder or softer strata, vibration and noise change.

To keep the mining machine traveling in the desired heading, as specified in the mining plan, the machine operator must visually align the machine with the survey marks on the roof. Vision is also necessary to guide the machine from one location in the mine to another, to negotiate entries, and to avoid obstacles. Verbal communication with the face foreman is also necessary for machine guidance and to deal with abnormalities.

To detect machine problems or failures, the machine operator uses his senses of vision, hearing, and smell. For example, the operator can visually see a failure such as a ruptured hydraulic hose, hear changes in the sound a motor is making, or smell a hot motor.

All these sensory inputs are then passed to the operator's brain, where decisions are made and control of the machine initiated. The human operator must make decisions not only for normal conditions, but also for abnormal conditions. This sometimes requires the operator to consult the face foreman or other mine experts who are more knowledgeable of how to best deal with the situation.

An autonomous continuous mining machine must be able to sense, make decisions, and carry out machine control (fig. 2) as described above for a manually-operated machine. The key technology elements required for an autonomous continuous mining machine are shown in fig. 3.

Basic Mining Machine

As previously discussed, the drum-type continuous mining coal extraction machine is the first target for the robotics and automation technology. Research is being directed toward machines currently being used in the mining industry so that the technology developed can be easily implemented by mining machine manufacturers.

Computer Systems

The computer system provides the backbone for intelligent mining systems. In an intelligent system, the computer must interface to a variety of internal (machine world) and external (surrounding environment) sensors, gather data from the sensors, be able to make decisions based on the real-time sensor data, and initiate and carry out machine control. Essentially, the computer system has to play the role of a human to sense the environment, make decisions, and control the machine based on those decisions. The computer must know what to do not only under routine conditions, but also during abnormal conditions.

Machine Control

Establishing accurate, computer-based control of a continuous mining machine is the first step in the development of autonomous, robotic equipment. Machine control refers to the steps necessary to establish and maintain accurate control of the mining machine appendages (cutting boom, gathering head, conveyor elevation and swing, and stabilizer jack) and locomotion tracks.

Guidance Systems

An autonomous mining machine must be able to guide itself not only in the face area, but also throughout the entire mine. This requires sensors to provide information on machine position and heading, and the distance to walls and other obstacles. An intelligent computer system must take this real-time information, add it to already known information such as the mining plan, and decide where to cut coal or where to next navigate.

Coal-Rock Interface Detection

Not only must an intelligent mining machine be able to navigate horizontally throughout a mine, but it must also be able to keep its cutting within the coal seam or to some other specified vertical cutting pattern. Known as coal-rock interface detection, this ability is critical to any autonomous mining system. The machine must be able to tell when it is cutting coal, and when it crosses the coal-rock boundary to cutting overlying or underlying rock strata. Sensors to detect where the cutting head is vertically in the coal seam are necessary.

Diagnostics

Machine downtime is a significant cause of lost productivity from underground mining equipment. Being able to predict a machine failure in advance of its occurrence or to rapidly determine the cause of a failure would provide a significant increase in machine productivity. A machine diagnostic-predictive system is even more crucial when a human is not onboard the machine to see the failure occur.

Planning and Supervision

In a robotic application where a stationary robot is doing a routine, repetitive action, planning for the robot is greatly simplified. However, in a situation where the environment can change significantly, and where the robot is mobile, planning for the robot is complex. The robot must be able to update or change plans as conditions change. With an autonomous vehicle, a machine planner is required to make sure that a particular machine goal is achieved. For an autonomous continuous mining machine, the planner must be able to instruct the mining machine to mine coal according to a mining plan, while making adjustments as conditions warrant. This planner must deal with normal and abnormal conditions. The Bureau's Automation and Robotic program is addressing each of the key technology elements required for an autonomous continuous mining machine.

4. PROGRAM STATUS

A Bureau-owned Joy 16CM[1] is currently used as a testbed for evaluation and prototyping of systems for the Robotics and Automation program. The machine was available from previous mining research and met the needs at the time. However, because it was originally designed as a miner-bolter, it has certain peculiarities that make it unsuitable for use in future plans to take the machine underground. Therefore, a new machine will be acquired for evaluation of technology underground.

The Joy 16CM is currently under computer control [1]. An onboard computer has been designed, assembled, and programmed for collecting data from machine sensors and for controlling the machine. The computer is a real-time, multitasking, multiuser system consisting of off-the-shelf hardware. It is based on the Intel 80286 processor. Near-term expansion of the computer system will involve upgrading to a distributed processing network, Bitbus. This will allow each separate machine subsystem, such as a coal-rock interface detector, to have its own processor for carrying out its function. The Bitbus network will also permit other computers such as a PC, SUN workstation, or Symbolics computer to be interfaced to the onboard computer, through the network, for offboard tasks such as planning. It will significantly enhance system capabilities.

Accurate, closed-loop computer control of the Joy 16CM has been established. That is, through computer commands, each of the movable parts of the Joy 16CM can be controlled by sensor feedback with good accuracy. This has involved a series of open-loop and closed-loop tests of the Joy 16CM operating in free space at the Bureau's test facility. Sensors installed on the machine to provide the angular position of the movable parts provide the necessary feedback information for closed-loop control. Once accurate control of the mining machine was established, the ability to maintain that accuracy and stability under stress conditions was determined by cutting simulated coal known as coalcrete. Accurate computer control was maintained.

During the last year, a navigation scheme for an autonomous continuous mining machine operating in a room-and-pillar, two-pass mining scenario was defined [2]. It makes use of a suite of sensors that work in concert to enable the mining machine to navigate not only locally in the face area, but also throughout the mine. Sensors being evaluated for the sensor suite include sonar, a laser scanning system, a gyroscope, a fluxgate compass, and clinometers. The navigation scheme uses a reference frame, a mobile control structure, on which the laser-scanning units are mounted. The laser-scanning

[1]Use of manufacturer's names is for identification only and does not imply endorsement by the Bureau of Mines.

units send out a horizontal laser beam at a 90° field of view toward the mining machine on which retroreflector targets are mounted. The beam is reflected off the targets and returned to the scanning unit and detected by a photo detector. The other sensing systems are installed on the mining machine in the navigation scheme. These sensor systems are currently being evaluated on the mining machine testbed and on a locomotion emulator, a rubber-tired vehicle that can emulate the motion paths and control commands of various types of mining equipment.

Research on coal interface detector (CID) sensors involves several techniques. First, an investigation [3] of the fundamental physical properties of the coal-rock interface of the U.S. coal seams where the majority of the coal is produced and/or is expected to be produced in the future, is being conducted. Coal and rock samples from the major seams are being analyzed in the lab to determine if natural gamma radiation, which has been successfully used in Europe, or a new technique using optical, electrical, or mechanical properties of the coal-rock interface may be useful as a coal-rock interface detector.

Second, several specific techniques are being investigated for a coal-rock interface detector, including machine vibration and in-seam seismic [4,5]. In both techniques, sensors, in this case accelerometers, are attached to the mining machine for machine vibration, or to the coal, roof, and floor for in-seam seismic, to sense signals generated as the mining machine is extracting coal. Different signals are generated when the machine is cutting coal versus when it is cutting rock. Powerful, intelligent signal processing computer programs, referred to as "Adaptive Signal Discrimination Networks," are used to discriminate this difference. Once the system has been trained on known conditions, it can distinguish the difference on unknown conditions. The key to this technique is the intelligent signal processing program. In-mine data are being collected for lab analysis and training in the lab.

Lastly, another technique for CID being investigated [6] is a doppler radar system. Research into this unconventional radar technique is being pursued instead of the conventional brute force approaches of pulse, impulse, and FM/CW radar, which have reached their maximum limits of feasibility. The doppler radar concept is not dependent on the parameters that have limited the performance of previous coal interface detectors. The present concept is to move the sensor antenna through a small spatial cycle by electronically switching signals among four small stationary dipole antennas. The doppler history caused by the apparent antenna motion is stored in computer memory and is then correlated with a prestored template of all coal dielectric and depth combinations. A matrix array of dielectric and thickness probabilities is then obtained. A new parameter matrix is then picked, and a new doppler history array is generated. The process is repeated until one dielectric and one thickness correlate. An advantage of this technique is that the dielectric need not be known nor a dielectric value assumed, as in previous radar techniques, before the thickness can be determined. Only enough measurements need be made to obtain the desired statistical confidence level of correlation.

The Bureau is addressing the problem of machine failures and downtime through the application of expert system technology to the problem. An expert system is being developed to diagnose and/or predict continuous mining machine failures. It is a real-time, sensor-based system, using input from onboard sensors. The goal is to make it a predictive maintenance system which would be capable of monitoring sensor data over time, looking for degradation of machine components that would indicate a failure may occur in the future.

There are three main subsystems for the continuous mining machine maintenance expert system: electrical, hydraulic, and mechanical. Two parallel efforts are currently underway to develop the maintenance diagnostic expert system, one for the electrical subsystem [7], and one for the hydraulic subsystem [8]. In both cases, sensors have been defined and installed on the mining machine to accurately detect system failures, and to provide information to the expert system of them. The knowledge for each respective knowledge base is being developed in conjunction with experts in the field.

Research for machine planning is underway in several respects. Under contract [9] with West Virginia University, the Bureau is developing an expert system to assist a face foreman in decision-making. The same rules and thought process used in the expert system will be part of autonomous machine planning. Under another cooperative contract, Carnegie Mellon University is working on planning strategic level actions. They have produced a Small-talk-80 implementation for strategic planning, modeling a network of actions describing a continuous miner in entry and crosscut operations. Effort currently underway is to develop the merging of machine task planning with a geometric model of the domain. An object representation for face, ribs, floor, and roof will be developed for the local environment, as well as an object representation for the extended environment, typified as a mine map.

5. SPINOFF TECHNOLOGY

Although the Bureau's program is investigating technology for a completely autonomous mining machine, much of this technology can be applied to mining problems in current mining scenarios to provide an immediate benefit. Two mining scenarios in which this technology may be immediately beneficial are in deep cut mining and highwall mining. In deep cut mining, continuous mining machines, operated by radio or tethered remote control, are used to make cuts of 40 ft, going beyond the point of supported roof. Since the operator is not on the machine, this deeper cut is permitted. However, with this deeper cut, the operator may not always be able to see the face or, in certain cases, the mining machine. Both vertical coal-rock interface detectors and lateral guidance systems would be beneficial in these cases.

With highwall mining, operators are interested in penetrating into the highwall 500 to 1,000 ft. Since the miner is out of visual view, again CID is necessary. In highwall mining, a constant rib thickness is maintained for the length of the penetration. If too little rib is left, the roof may fall. If wider than necessary rib is left, coal is wasted. Therefore, an accurate lateral guidance system is needed. The radar system being researched for CID is also expected to be useful here.

CID technology would also be beneficial for vertical guidance of longwall shearers. Longwalls typically produce far less than their design capacity, and the operator is often exposed to respirable dust. Vertical guidance technology will both increase productivity and remove miners from unhealthful conditions.

6. PROGRAM SCHEDULE

Work under the Robotics and Automation program will develop technology, as described in this paper, for a completely autonomous mining system. A timeline showing major milestones (large circles) and the research (small circles) required to attain these milestones, is shown in fig. 4.[2] Machine control [1] was established in 1988. Machine presence, the next milestone, will be completed in 1990. For this milestone, position-heading navigation technology [2] and coal-rock seam detection [3,4,5,6] must be completed. In 1992, the machine guidance milestone will be completed, incorporating machine planning, which includes contingency reactions and machine action lists. Between 1992 and 1995, machine diagnostics [7,8] will be available, MSHA approvals will be addressed, in-mine evaluations will be conducted, and the equipment will be ruggedized. This all leads to technology for an autonomous mining machine being available in 1995.

Again, as technology pieces for an autonomous mining machine become available, they will be implemented, as appropriate, to current mining machine operations.

7. FUTURE RESEARCH

While the thrust of the Automation and Robotics program is directed toward the short-term objective of technology for an autonomous continuous mining machine, longer term objectives are taking a broader approach, looking at robotics technology and how it can lend itself to improving the coal extraction process. This may lead to new machine designs or new mining processes.

8. SUMMARY

The Bureau of Mines Robotics and Automation research is addressing the problems of mining inefficiency and accidents in underground coal mines through the development of technology for autonomous mining systems. Work on the key technology elements required for an autonomous continuous mining machine, including intelligent computer systems, machine control, guidance systems, coal-rock interface detectors, diagnostics systems, and planning systems is underway.

REFERENCES

[1] Sammarco, J. J., and W. H. Schiffbauer. Computer Control of a Continuous Mining Machine. Presented at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[2] Anderson, D. L. Position and Heading Determination of a Continuous Mining Machine Using an Angular Position Sensing System. Presented at The Ninth WVU International Conference on Mining Electrotechnology,

[2]The timeline only relates to the major milestones; most of the research for these major milestones is already underway.

Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[3] Dobroski, Jr., H. The Application of Coal Interface Detection (CID) Techniques for Roboticized Continuous Mining Machines. Presented at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[4] Mowrey, G. L. Applying Adaptive Signal Discrimination Systems to Mining Problems. Poster Session at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[5] Cislo, R. P., and M. J. Pazuchanics. Evaluation of a Machine Vibrational Coal Interface Detector. Poster Session at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[6] Chufo, R. L. A New Radar Detector for Coal Interface Detection. Poster Session at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[7] Berzonsky, B. E. A Knowledge-Based Electrical Diagnostic System for Mining Machine Maintenance. Poster Session at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[8] Mitchell, J. A Knowledge-Based System for Hydraulic Maintenance of a Continuous Miner. Poster Session at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

[9] Nutter, R. S., L. Grayson, F. Londono, R. Raman, M. Watts, N. Chatree, and R. Reddy. Face Decision Support System: An Expert System for the Section Foreman (contract H0358020, West Virginia Univ.). Presented at The Ninth WVU International Conference on Mining Electrotechnology, Morgantown, WV, July 26-29, 1988, and scheduled for publication in the Proceedings.

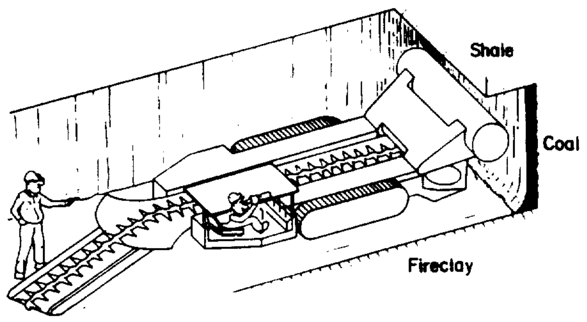


Fig. 1. - The operator of a continuous mining machine relies on his sensory systems to control the machine.

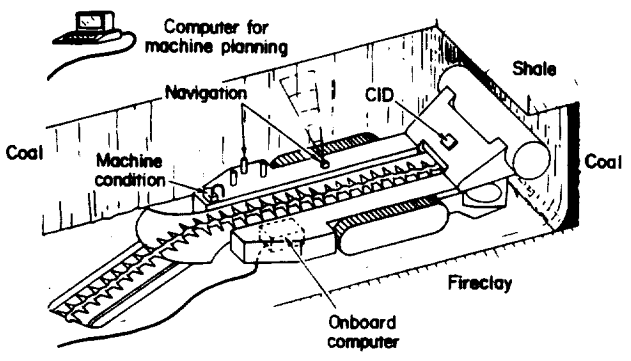


Fig. 2. - Key components for an autonomous continuous mining machine.

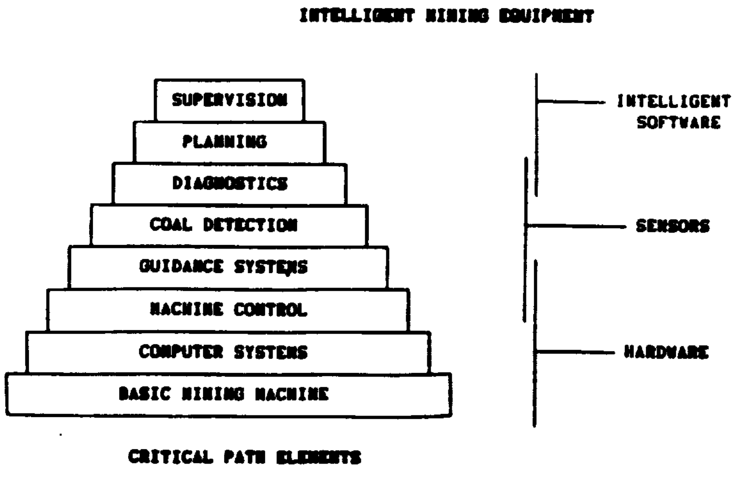


Fig. 3. - Key technology elements for an autonomous mining machine.

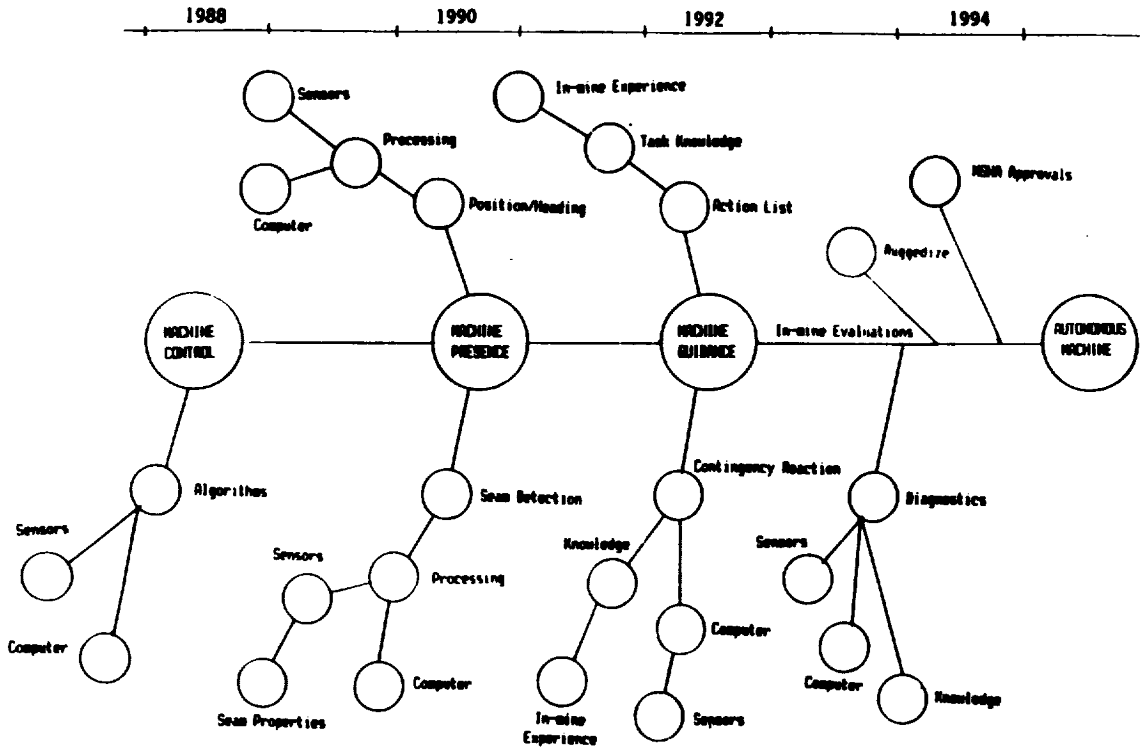


Fig. 4 - Technology timeline for an autonomous mining machine

ROBOT SENSING AND PLANNING

A Fast Lightstripe Rangefinding System with Smart VLSI Sensor

Andrew Gruss, L. Richard Carley, and Takeo Kanade

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Introduction

The focus of our research is to build a compact, high performance lightstripe rangefinder using a VLSI *smart* photosensor array.

Rangefinding, the measurement of the three-dimensional profile of an object or scene, is a critical component for many robotic applications, and therefore many techniques have been developed [2]. Of these, lightstripe rangefinding is one of the most widely used and reliable techniques available.

Though practical, the speed of sampling range data by the conventional light stripe technique is severely limited. A conventional light stripe rangefinder operates in a *step-and-repeat* manner. A stripe source is projected on an object, a video image is acquired, range data is extracted from the image, the stripe is stepped, and the process repeats. Range acquisition is limited by the time needed to grab the video images, increasing linearly with the desired horizontal resolution. During the acquisition of a range image, the objects in the scene being scanned must be stationary. Thus, the long scene sampling time of step-and-repeat rangefinders limits their application.

The fast range sensor we propose to build is based on the modification of this basic lightstripe ranging technique in a manner described by Sato [6] and Kida [3]. As will be seen, this technique does not require a sampling of images at various stripe positions to build a range map. Rather, an entire range image is acquired in parallel while the stripe source is swept continuously across the scene. Total time to acquire the range image data is independent of the range map resolution.

Our target rangefinding system will acquire 1,000 100×100 point range images per second with 0.5% range accuracy. It will be compact and rugged enough to be mounted on the end effector of a robot arm to aid in object manipulation and assembly tasks.

Integrated *Smart* Sensing

The search for an efficient implementation of the parallel algorithm leads one to the use of *smart* sensors. A smart sensor has cells which provide processing at the point of sensing. Our range sensor uses smart cells to independently acquire data for range map points in parallel. When a scan has completed, the collected data can be read sequentially from the sensor. The slight increase in cell functionality from sensing-only to sensing-and-storage makes a high performance rangefinder based on the parallel algorithm realizable.

Advances in VLSI technology make smart sensors possible and hold the promise for further integration of computation and sensing. The key to a VLSI implementation of the lightstripe sensor chip is the ability to integrate photoreceptors, analog circuitry, and digital logic on a single CMOS chip. Examples of this class of chip exist and include commercial *CCD* camera chips, the *Xerox Optical Mouse* [4], Mead's *Artificial Retina* [7], and an *Optical Position Encoder* done at the CSEM in Switzerland [1].

Parallel Rangefinder System Overview

Algorithm

Figure 1 shows the principle on which a lightstripe rangefinder operates. The scene is illuminated with a vertical plane of light. The light is intercepted by an object surface in the path of the beam and, when seen by a video camera placed left of the light source, appears as a stripe which follows the surface contour of objects in the scene.

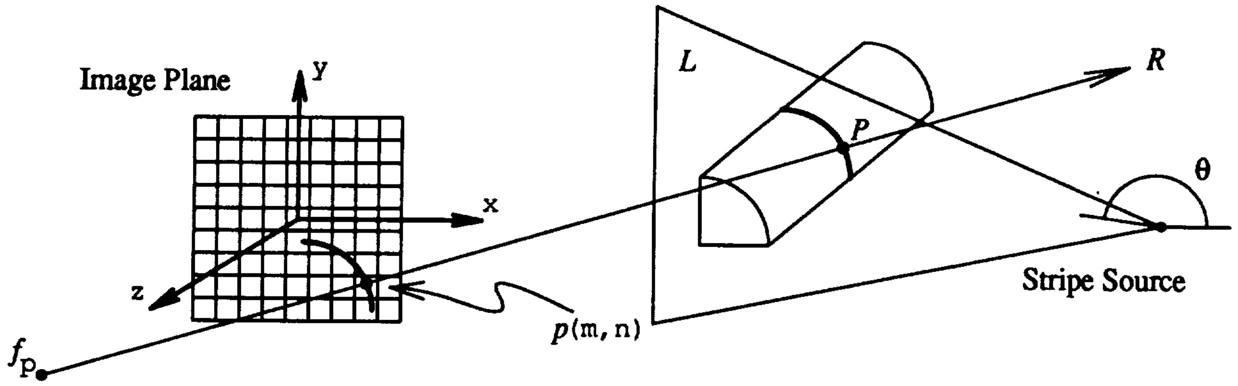


Figure 1: Lightstripe Rangefinder Geometry

Range data along the contour can be calculated easily using the principle of triangulation. In figure 1, the equation of the plane of light L is known because the projection angle θ is controlled. The line of sight R for each point p on the image of the stripe can be also determined by tracing a line from the image focal point f_p through p. The intersection of the ray R with the plane L uniquely determines the three-dimensional position of P on the surface corresponding to p. Range data of the whole scene is collected via a *step-and-repeat* procedure, that is, iterating the process of fixing the stripe on the scene, taking a picture, and processing the resultant image until the entire scene has been scanned.

Though practical, the speed of sampling range data by the conventional light stripe technique is severely limited. Assume that a video camera image has N rows. Since from one image at each step we can obtain up to N data points, the maximum speed of sampling is $S_{max} = \frac{N}{T_f}$ where T_f is the time required to acquire and process an image frame. Typically, N ranges between 256 and 512 samples and T_f ranges between one-thirtieth and one-tenth of a second. Thus, sampling speeds of camera based systems are limited to $S_{max} \approx 2.5K \sim 15K$ *samples/second.*

In the parallel rangefinding technique, the video camera is replaced by a two-dimensional array of smart photosensitive cells. In addition, range data is not acquired in a step-and-repeat manner. Instead, the plane of light is swept across the scene at a constant angular velocity once from left to right.

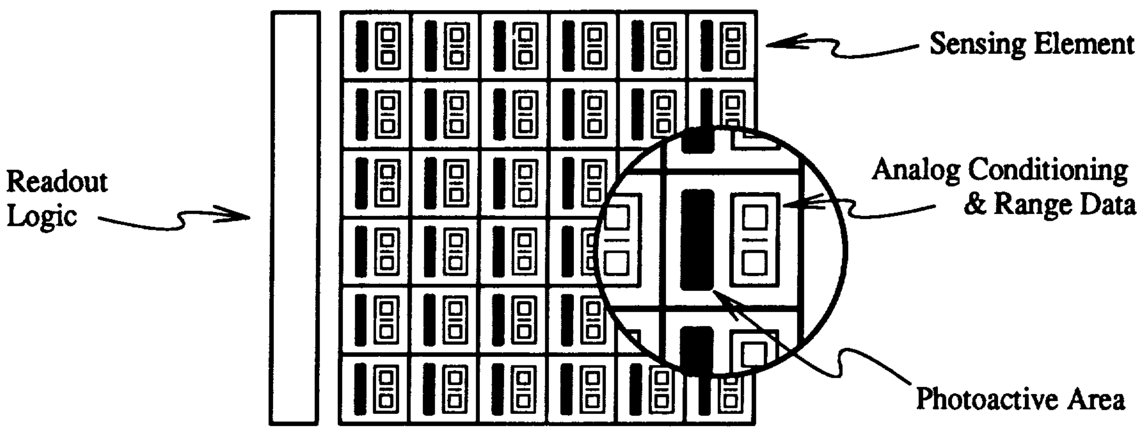


Figure 2: 2D Array of *Smart* Photosensors

The array of photosensitive cells, shown in figure 2, is smart for the following reason. Each has circuitry that can remember the *time* at which it observed the peak incident light intensity during a sweep of the stripe L. Observe that each cell predefines a unique line of sight R and that the information t_{cell} recorded by each cell defines a particular orientation of the stripe $L(t_{cell})$. Recalling the geometry in figure 1, one sees that this information is sufficient to calculate the three-dimensional position of the imaged object point P, again using triangulation. The data gathered during *one* pass of the stripe in an $M \times M$ array of these smart sensing elements is sufficient to calculate the $M \times M$ range map of an imaged scene.

For an $M \times M$ array of these cells, the sweep time T_s of the lightstripe and M determine the sampling speed. The sweep time T_s will be limited by photoreceptor sensitivity and M by integration technology. We are predicting values of T_s on the order of one to ten milliseconds, and values of M ranging between 40 and 100. The rates at which the smart sensor array generates range data will be $S_{max}^{\text{Smart}} = \frac{M^2}{T_s} \approx 0.2M \sim 10M$ *samples/second.*

This is a speedup of several orders of magnitude over that of a conventional camera-based system.

System Implementation Issues

The hardware necessary to construct a rangefinding system based on the parallel technique is comparable to that found in a conventional rangefinder. One needs a photosensitive array, stripe generation hardware, and system interface circuitry.

Functionally, each element of the smart photosensitive array converts light energy into an analog voltage, determines the time at which the voltage peaks, and remembers the time at which the peak occurred. The implementation of this functional specification requires that photoreceptor and signal conditioning circuitry be integrated into a unique hybrid sensor cell. The sensing element design must consider tradeoffs in cell size, power dissipation, bandwidth, sensitivity, and accuracy.

Special care must be taken with the photodiode amplification stages. Photocurrents induced by incident light from the stripe are on the order of a nanoamp and must be amplified to reasonable voltage levels. In addition, the high rate of range map acquisition supported by our system implies high bandwidth photodiode signals. A 100×100 element sensor gathering 1,000 range images per second requires an amplifier that can provide gain out to $100KHz$. On the other hand, we know that the frequency content of signals generated by a continuously moving stripe will be found above the base scanning frequency. Thus, the low pass nature of amplified photocurrents should be combined with a high pass filter stage to yield an amplifier with an overall bandpass frequency response. The cells that result will be most sensitive to frequencies generated by the image of a moving stripe. The amount of interference caused by ambient light and signal conditioning circuitry $1/f$ noise will be reduced.

The stripe generation hardware consists of a coherent light source, stripe optics, and sweep mechanics. It must project a continuously moving stripe whose geometry with respect to the sensor is known as a function of time. A *start-of-scan* (SOS) detector and its conditioning electronics must also be included in the stripe generation assembly. The SOS indication defines a reference point in time relative to which range data will be measured and recorded by the sensing elements.

The system interface circuitry bridges the gap between range sensor and host processor. Range image acquisition must be coordinated between the sensor and the stripe generation hardware. Sensing element data must be acquired and accurately converted into a form usable by the host. Finally, the system interface must provide a high bandwidth path to the host for acquired range data.

Photodiode Based Test System

Implementation

We have designed and built a prototype rangefinding system based on the parallel algorithm. Essential components in this system included stripe generation hardware, range sensor, range sensor optics, and host interface. This implementation is similar in spirit to systems built by others [6] [3] and served as groundwork for our VLSI sensor based system.

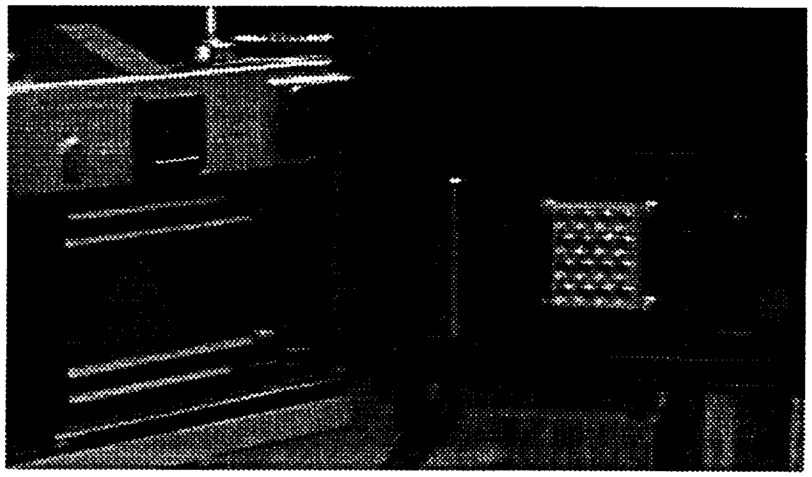


Figure 3: 4×4 Photodiode Array Mounted in a 35mm Camera

The sensor in our evaluation system has been constructed using a discrete 4×4 array of photodiodes as the sensing device as seen in figure 3. The photodiodes are mounted in a 35mm SLR camera body which provided a convenient mechanism for incorporating focusing optics and for sighting the rangefinder.

Photodiodes were chosen for the sensing elements because they possess bandwidth sufficient to meet our sweep rate specifications.

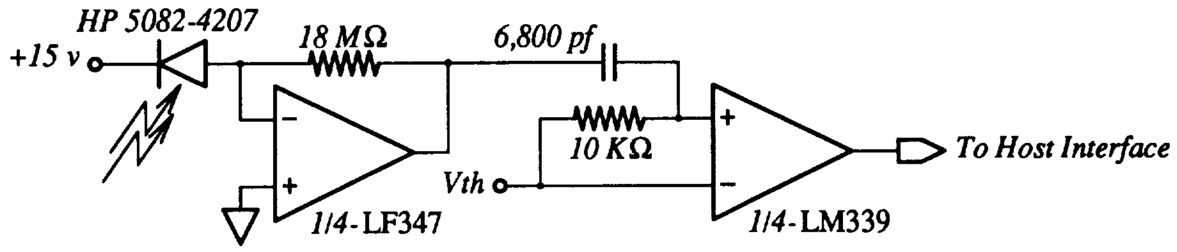


Figure 4: Photodiode Signal Conditioning Circuitry

Analog signal conditioning circuitry for each of the discrete photodiodes was designed to provide a digital transition when the reflection of the stripe passed through a diode's field of view. It consisted of a high gain transimpedance amplifier and simple thresholding stage as shown in figure 4.

The design of the photocurrent amplifier was crucial. For purposes of small signal bandwidth analysis, reversed biased photodiodes can be modeled as the parallel combination of a current source of a few nanoamps, a resistance, and a capacitance. Though the internal capacitance is moderate, on the order of a few picofarads, the internal resistance is ten gigaohms or more. The parallel combination of these creates an undesirable pole at a frequency of a few hertz which tends to low pass filter any output signal. The photocurrent amplification circuitry must provide a large photocurrent to voltage gain while presenting a small impedance to the diode. The amplifier shown in figure 4 provides $18M\Omega$ of photocurrent amplification and employs negative feedback to servo the photodiode anode to a constant voltage, increasing available bandwidth.

Amplified photodiode signals were high pass filtered before reaching the comparator. This was done to make the comparator threshold level independent of photodiode dark current and ambient light levels and to remove low frequency circuit noise.

The digital output from the comparator in each cell was passed directly to the host interface. This is practical when a sensor of only 16 elements is involved, but would not be practical for sensor densities much above 10×10. On the host interface, the 16 comparator outputs were sampled into a local dual-ported memory. Host access to the data was provided via a memory-mapped VME interface.

The stripe in this discrete implementation was generated using a $5mW$ helium-neon (HeNe) laser and half-cylindrical lens. Sweeping of the stripe was accomplished using a mirror mounted on a galvonometer. Use of a galvonometer to sweep the stripe meant that scans alternated in direction between left-to-right and right-to-left. The galvonometer was driven with a $500Hz$ triangle wave in order to generate our target 1,000 sweeps per second.

Two additional photodiodes were used to provide *start-of-scan* (SOS) and *end-of-scan* (EOS) indications necessary for determining the time origin, direction, and duration of a sweep. Conditioning circuitry used for these scan detectors is similar to that used by the sensing elements. The digital output from these detectors is used by host interface hardware to initialize the sample memory address counter and determine the direction of stripe scan.

Results

The photodiode based rangefinding system hardware was able to generate and record over 1,000 4×4 range images each second. System software, running on a SUN 3/160 workstation, slowed the rate of processed image data to about 100 frames a second. Range data, encoded in the time from a scan origin to when a sensing element sees the flash, was continuously displayed on the monitor of the host workstation.

VLSI Range Sensor Based System

System Overview

From a speed and sensitivity standpoint, our discrete photodiode based system is a successful implementation of the parallel rangefinding algorithm. However, a 4×4 array does not provide enough range points to be useful. As one considers building larger and larger arrays out of discrete photodiodes, implementation problems quickly become apparent. The cost of wiring individual photodiodes to interface circuitry becomes prohibitive for arrays much larger than 10×10. Support circuitry must be built out of off-the-shelf analog and digital IC components for each sensing element. Schemes which time multiplex wires from the photodiodes and sensing element conditioning electronics are not practical because one cannot predict when a given cell will see the stripe.

A VLSI implementation of the range sensor shows the greatest promise for increasing the range image density of the system. The essential thing that VLSI provides is the ability to integrate the sensor, conditioning circuitry, and range memory into a single smart cell. Wiring costs from sensor to amplifier to memory circuitry are virtually eliminated. Time multiplexed readout of range data is practical once data for a scan has been recorded within the sensor cells. Thus, the range map resolution of a VLSI based sensor is not limited by the number of connections which can be made to the sensing elements.

Range Sensor IC Interface

The range chip provides data in the form of two time multiplexed *analog* outputs. The first transfers sensed stripe arrival time values from the chip. The second provides the intensity seen at a cell when the incident stripe intensity was at a maximum. This intensity output will give a rough idea of scene reflectance and will be useful in determining the level of confidence one can attach to the corresponding time sample. Storage of chip data as analog values might at first seem to be inherently noisier than storing acquired data digitally. Justification for this decision is outlined in the description of the sensor chip.

The system interface will control the range image acquisition process, drive the sensor chip, retrieve raw range data, and make this data available to the host over a high bandwidth path. Two analog-to-digital converters will be necessary. The first will be used to convert the analog time values, the second to convert the maximum sensing element intensity values. These analog-to-digital converters will need to have a $2MHz$ conversion rate and better than ten bits of accuracy. Stripe sweep control circuitry will generate the analog time ramp. The period of the ramp will be phased locked to index pulses generated by the SOS detector.

IC Based Rangefinder Stripe Generation

HeNe laser tubes are not well suited for use in a compact rangefinding system. We plan to construct an infrared (IR) laser diode based stripe source. Silicon photodiodes have good sensitivity to light in the near infrared[1] area of the spectrum, in the range wavelengths emitted by typical IR laser diodes. A good match of spectral characteristics between stripe source and photodiode based detector will aid in stripe detection. A cylindrical lens will serve to fan the collimated beam into a stripe.

In order to sweep a 60° field of view in one millisecond, the stripe will have to rotate at 10,000*RPM*. A multifaceted mirror attached to the shaft of a motor spinning at 5,000*RPM* is one candidate for sweeping the stripe. The stripe sweep hardware and range sensor will be assembled as one unit to insure an accurate and steady baseline for range calculations.

The Sensor Chip

A block diagram of our sensor chip, showing sensing element layout, can be seen in figure 2. Photodiode areas are arranged in vertical stripes which are to be aligned in the direction of the imaged light stripe. Sensing element support circuitry is sandwiched between the photodiodes. As will be described, the design of sensing elements on the IC differs from the design of sensing elements in the discrete photodiode implementation in several important ways.

Sensor chips and test structures are being fabricated in a 2.0μ CMOS[2] P-well double-metal, double-poly process. Fabrication is provided through the MOSIS [5] system. Global chip busses required for power and ground runs, timestamp input, and multiplexed data readout can easily be realized with this two layer metal process. Total sensing element area is projected to be $200\mu \times 200\mu$. We expect to be able to integrate a 40×40 cell sensor in a roughly one square centimeter die. In order to keep chip power dissipation to a few watts, cell current is budgeted at around 200μ*amps*.

If our sensor chip design is to be successful, attention must be paid to the following issues:

- design of the integrated photoreceptors,
- photocurrent amplification bandwidth and noise floor,
- representation of the global timestamp signal on the chip,
- data storage capability, and
- design of the sensor chip interface.

Refer to the block diagram for one sensing element, shown in figure 5, as we discuss the manner in which these issues are addressed in our sensing element design.

Integrated Photodiodes as Sensing Devices

The photodiodes are critical to the sensitivity and bandwidth of the sensor cells. Current output at a given incident light intensity is directly proportional to the photodiode area. The more area devoted to photodiode structures the better the optical sensitivity of the sensing elements will be. Our photodiodes are 20,000μ^2 in area, one half the total area budgeted for a cell.

[1]Near infrared includes wavelengths between 700*nm* and 1,000*nm*. IR laser diodes emit light at wavelengths ranging between 820*nm* and 880*nm*.

[2]A *micron* (μ) is 10^{-6} *meters*.

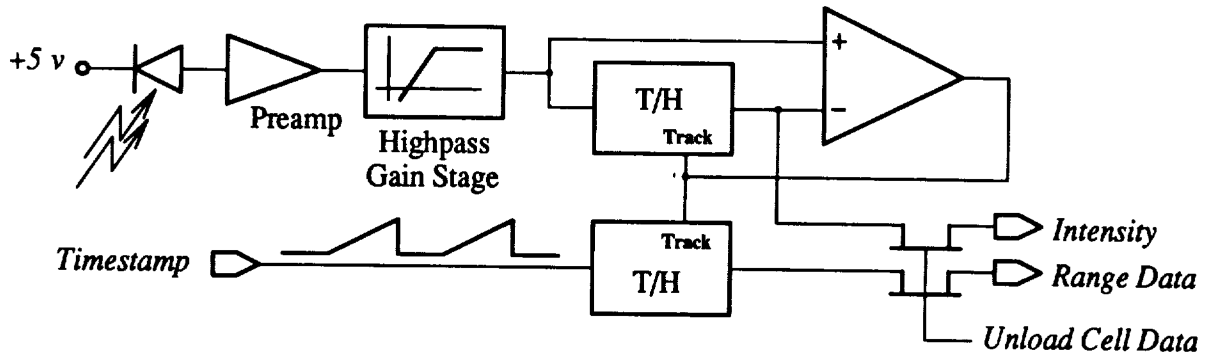


Figure 5: Sensing Element Circuitry

In a CMOS process, maximum sensitivity photodiodes are build using the well-substrate junction [1]. This vertical photodiode structure is constructed using the n-type substrate as the cathode and the p-type well as the anode. An additional p^+ implant is driven into the well to reduce the surface resistivity of the anode to which contact is made. Finally, the photodiode structures are surrounded with guard rings to minimize the chance of photocurrent induced latchup. Only the anode of the photodiode is accessible for reversed biased operation. The cathode of the diode will be at the substrate voltage.

Photocurrent Amplification

A candidate photocurrent preamplifier is based on one developed for use in an optical position encoder [1]. This amplifier is logarithmic in that its output voltage is proportional to the log of the induced photocurrent. A negative feedback loop consisting of a p-channel FET common source amplifier and common base lateral NPN transistor servos the photodiode anode to a constant voltage. This reduces the effective capacitance of the photodiode by a factor equal to the loop gain and thus extends operational bandwidth. The lateral bipolar transistor used in this circuit is fabricated in a P-well area into which a p^+ base contact has been implanted. A small area of n^+ diffusion, which becomes the emitter, is surrounded by a ring of n^+ diffusion to form the collector. A parasitic vertical NPN structure is also formed by the substrate, the P-well base, and the emitter. The parasitic lowers the α of the lateral device but does not adversely affect its operation as a common base stage.

The output of the transimpedance amplifier feeds a second stage of amplification through a high pass filter. As was the case in the discrete photodiode implementation of this sensor, the high pass section nulls out the effects of dark current and ambient light. In addition, $1/f$ noise inherent in MOSFET based amplifiers is also filtered out. The low pass nature of the photodiode combined with the high pass filter of the second gain stage yields an overall bandpass transfer characteristic for the photodiode amplifier stages. Thus, sensing elements on the chip will be most sensitive to those frequencies generated by the image of the stripe moving across the sensor.

Representing Time as an Analog Voltage

Representing time as an analog voltage has several advantages over the digital equivalent of latching the value of a continuously running counter. The analog only scheme avoids noise problems associated with mixing sensitive analog circuits with digital logic within the cell. A digital timestamp bussed over an entire chip, combined with transients associated with the latching of timestamp values by sensing elements, are sources of noise with the potential to corrupt the measurement of small levels of photocurrent. Photodiode anode points are high impedance nodes and will be susceptible to noise coupling from other chip circuitry.

The chip area needed for timestamp broadcast and latching will be smaller for the analog scheme when compared with the circuit area required by the digital scheme. An analog timestamp can be broadcast over the entire chip on a single wire and the circuitry to record an analog time value consists of a holding capacitor and a switch. The eight bits of digital time necessary for 0.5% resolution would have to be broadcast over a bus and each sensing element would need an eight bit latch.

Raw Range Data Storage with Track-and-Hold Circuitry

The system analog timestamp voltage is switched on to the holding capacitor in a track-and-hold (T/H) circuit until incident intensity has peaked. A second T/H follows the sensed light intensity until the point when its held voltage exceeds the input voltage. At that time, the comparator stage changes state, disabling the T/H circuits and recording a range time value.

In addition to the structures that can be built using a standard P-well CMOS process, the double-poly process we are using provides high quality linear capacitors. These capacitors exhibit good matching across a die and are needed by sensing elements on the range sensor IC to store analog voltages. The matching of these capacitors across the sensor chip will in large part determine the variance in voltage reported by individual sensors for given time values. At capacitive densities of $0.5 ff/\mu^2$, a $1 pf$ T/H capacitor will be $45\,\mu$ on a side.

Host Interface Considerations

Charge accumulated in the sensing elements on the holding capacitors is passed out of the chip on a bus and integrated to produce a voltage. Both the range data and maximum intensity values will be read from the sensor in this way. We plan to offload sensing element range data in raster fashion, much like a CCD camera chip. For the initial 1,600 element sensor, we can spend 500 *ns* on each cell if range acquisition time and offloading time are to remain comparable. Future versions could certainly take advantage of multiple data pathways to reduce the time necessary to dump stored range data. Data could also be pipelined with range acquisition if two sets of T/H circuitry were built into each cell. Initially we have decided not to do this for two reasons. First, cell area would grow mainly due to the size of the two additional T/H capacitors needed. Second, time multiplexing of the on chip busses is essentially a digital process which has the same noise pitfalls as any other digital circuitry on the chip. By separating acquisition and offloading phases, we can insure that no digital switching will be occurring while range measurements are taking place.

Future Work

Our decision to represent range data on the sensor chip as an analog voltage will enable us to apply analog signal processing techniques for on chip computation. Simple computations can be done with analog circuitry in less area than possible with corresponding digital computations. For example, switched capacitor technology can reduce the circuitry needed to compute a weighted average to a few capacitors and transistors. We hope to explore these possibilities further.

We also plan to explore the use of layered or *three-dimensional* (3D) VLSI to assist in increasing sensor density. A 3D VLSI process is an ideal one for building a dense parallel range sensor. Photosensitive elements can cover the surface of the chip without gaps in a 3D process, maximizing sensitivity and range image density.

The research leading to the development of our lightstripe chip has great potential for advancing smart sensor technology in general. The sensor will need to acquire, amplify, and process information derived from a weak incident power source. We will need to incorporate ideas from a variety of disciplines to achieve this goal. In addition, decisions on the kinds of signal processing operations to be performed on the sensor itself must be made. A *smarter* chip has the potential to operate on its data directly, saving the considerable time and bandwidth spent shuffling data between sensors and processing in a typical system.

References

[1] P. Aubert and H. Oguey.
An Application Specific Integrated Circuit (ASIC) with CMOS-Compatible Light Sensors for an Optical Position Encoder.
In *IEEE 1987 Custom Integrated Circuits Conference*, pages 712-716. IEEE, May, 1987.

[2] P.J. Besl.
Range Imaging Sensors.
Research Publication GMR-6090, General Motors Research Laboratories, March, 1988.

[3] T. Kida, K. Sato, and S. Inokuchi.
Realtime Range Imaging Sensor.
In *Proceedings 5th Sensing Forum*, pages 91-95. April, 1988.
In Japanese.

[4] R.F. Lyon.
The Optical Mouse, and an Architectural Methodology for Smart Digital Sensors.
Technical Report VLSI-81-1, Xerox Palo Alto Research Center, August, 1981.

[5] G. Lewicki, et. al.
MOSIS User's Manual
USC Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90292-6695, 1988.

[6] Y. Sato, K. Araki, and S. Parthasarathy.
High Speed Rangefinder.
SPIE , 1987.

[7] M.A. Sivilotti, M.A. Mahowald, and C.A. Mead.
Real-Time Visual Computations Using Analog CMOS Processing Arrays.
In P. Losleben (editor), *Advanced Research in VLSI -- Proceedings of the 1987 Stanford Conference*, pages 295-312. The MIT Press, 1987.

Methods and Strategies of Object Localization

Lejun Shao[1] and Richard A. Volz[2]

[1]Robot Systems Division, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan 48109.

[2]Department of Computer Science, Texas A&M University, College Station, Texas 77843.

Abstract

An important property of an intelligent robot is to be able to determine the location of an object in 3-D space. This paper proposes a general object localization system structure, discusses some important issues on localization and gives an overview of current available object localization algorithms and systems. The algorithms reviewed in the paper are characterized by their feature extracting & matching strategies, the range finding methods, the types of locatable objects and the mathematical formulating methods.

1 Introduction

Since the early 1980's, the use of robots in industry has become increasingly popular or even crucial in some areas. Through practice, researchers have realized that an important property of an intelligent robot is the ability to determine the location of stationary and moving objects. For example, space station construction, repairing, maintenance, satellite refueling, etc. have been identified as the potential areas of applications of telerobotics systems. The availability of an efficient means of locating objects is one of the key factors to the success of developing such systems.

Object localization has long been defined as a part of object recognition process in computer vision research [4]. But in most instances, the emphasis of the research is on object recognition. Object localization is only a by-product. In robotic applications, however, object localization usually is the ultimate goal. And, it has many of its own problems to be solved such as real-time considerations, accuracy issues, types of locatable objects, working conditions *etc.*, which object recognition research generally does not address. In some systems, "**locate**" has been defined as one of the basic independent operations the telerobot system is to perform [27]. As a result, object localization research has attracted increasing attention recently.

This paper will give an overview of the three-dimensional object localization problem. First, it provides a closer examination of the problem and then proposes a general object localization system structure. Some important issues of object localization and the possible implementations of key components of the proposed object localization system are discussed and compared. A summary is presented in the final section.

2 The Object Localization Problem

As we have mentioned, object localization is the determination of the location of an object. What must be solved when a robot vision system is trying to locate an object? A necessary component of every intelligent robot system is the world modeling system which stores, among other things, a representation of all the object models that are relevant to the robot's operation and a definition of the sensing coordinate system. To specify an object in the world modeling system, there must exist either an implicit or explicit coordinate system which is associated with that object.

The real problem to be solved when locating an object, therefore, is to determine the relative location between the sensing coordinate system and the coordinate system for the object, which is somewhat different from what the object localization means from the point of view of object recognition. In object recognition, localization can mean a description of the location relationships among the objects to be recognized.

The relative location of two coordinate systems can be specified in any one of the following methods:

1. *Position and orientation:*
   The position can be specified by three parameters, e.g., the (x,y,z) coordinates of the origin of the object coordinate system relative to the sensing coordinate system. There are three different representations of orientation:
   - Three Eular angles α, β, γ, or angles about the coordinate axes.
   - A unit vector $\mathbf{r}$ and an angle θ.
   - A quaternion. [22, 26]
2. *A* 4×4 *homogeneous transformation matrix.*
3. *Dual number quaternion:* [37]
   This is an extension of quaternion representation in which each quantity is changed to a dual quantity [11]. The dual quaternion has a similar interpretation as the real quaternion:
   $$\hat{q} = \begin{bmatrix} \sin(\hat{\theta}/2)\hat{n} \\ \cos(\hat{\theta}/2) \end{bmatrix}$$
   where the vector $\hat{n}$ is a unit line vector about which the coordinate system has rotated and translated and $\hat{\theta}$ is the dual angle of rotation and translation.

The objective of the object localization algorithm is to compute the parameters which specify the corresponding representation.

Because there are six degrees of freedom a rigid object could have in three dimensional space (three for position, three for orientation), six independent parameters is the minimum number to be determined. The advantage of position/orientation representation is that it has minimum or near minimum number of variables. But there are disadvantages. The angular representations use trigonometric functions which are of infinite order and lead to a nonpolynomial criterion. Vectorial representations also have singularities; when the rotation angle θ is zero, the axis of rotation is arbitrary. The matrix representation is linear and has no singularity problem. But the inherent redundancy for rotation leads to a high-dimensional space constraints and will make the computation a little harder. The dual number quaternion representation has a dimension of eight, which is a little bit higher than the minimum but is still quite simple to compute.

In some applications, the localization problem can be simplified due to extra constraints imposed on the object. For example, if an object is so constrained that a planar surface of the object is always lying on a plane, the degrees of freedom of the object are reduced to three: one for the rotation and two for the translation.

How does one solve for these position/orientation parameters if sensor data and object models are given? Usually the computation is carried out by a matching process. That is, the object localization algorithm will try to find a "best" transformation which will put sensed features into its corresponding model features.

From the above description, it is not difficult to imagine that a general object localization system should contain the following components: (1) **sensing system**: to provide necessary measurements;

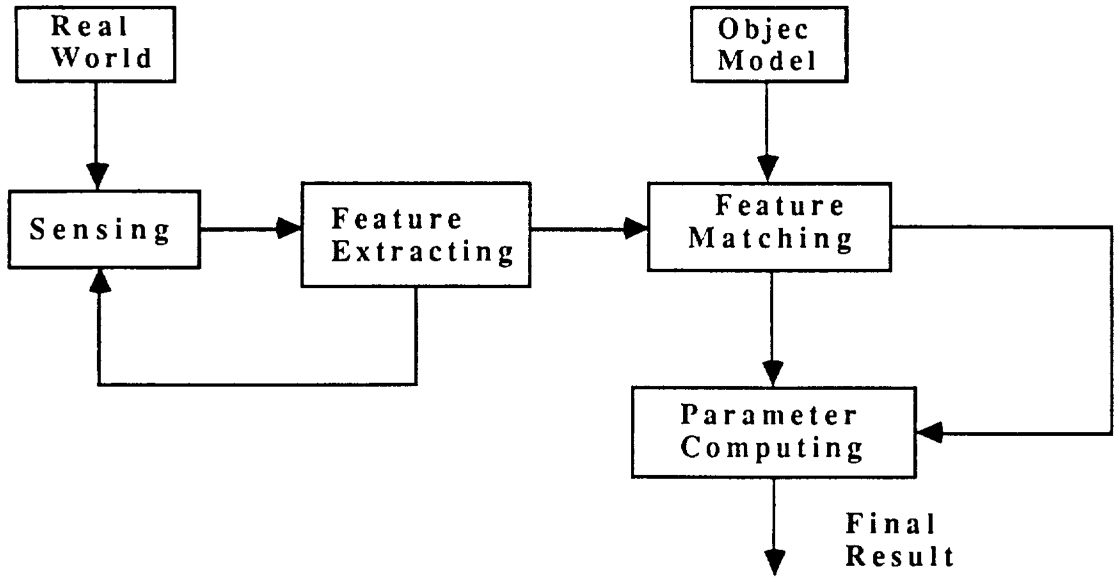


Figure 1: Object localization system organization

(2) **world model**: to represent all the objects including robot and sensors and their relationships; (3) **feature extraction**: to retrieve features which is to be used in the matching process; (4) **matching**: to try to pair the sensed features with corresponding model features; and (5) **computing**: to calculate the transformation parameters. See Fig. 1 for the configuration.

Based on the types of sensors used in the sensing system, the sensing method can be divided into serial sensing and parallel sensing. If the necessary sensor data is obtained through a series of measurements, such as the case when a spot range sensor is used, it is called as serial sensing; if the sensor data can be obtained by a single measurement, it is called as parallel sensing. In the case of serial sensing, the whole feature extraction might go through a repeated sensing-extraction process. The geometric features to be extracted and to be matched can be classified as low-level features and higher-level features. Possible low-level features include points, vectors, line segments, axes, surface patches, edges, boundaries and *etc.*. Possible high-level features include straight dihedrals, circular dihedrals [9], principle directions of surface curves, minimum, maximum and mean curvatures of surfaces, Gaussian curvatures, and *etc.*. Usually the lower the level of features is, the greater the number of features to be extracted.

The matching process is the process of finding the pairings of sensed features and the model features. Depending on the level of intelligence of the system, the matching could be done in different ways. On the lowest level, there is no matching process at all in the system. Whenever a measurement is taking place, either a default matching is assumed or a man-assisted matching is provided. On telerobotics systems, for example, the teleoperator might interactively assist the model matching by indicating with a light pen which features in the image (e.g. edges, corners) correspond to those in a stored model [1]. On higher levels, the system will be able to paring the features automatically. Table 1 shows some known feature matchings which have been used in literature to derive the location of an object. Sometimes, a combination of feature matchings are necessary to completely specify a rigid

Measured features	Matched to
Point	Point
	Planar surface
	Surface patch
Surface normal	Surface normal
Line segment	Line segment
	Planar surface
	Surface patch
Edge	Edge
Planar surface	Planar surface
Quadric surface	Quadric surface
Gaussian curvature	Gaussian curvature

Table 1: Known Matching Strategies in Object Localization

transformation.

We have just showed and discussed a general object localization mechanism. There is no common solution for the implementation now. Each component could be implemented in many different ways. Some components or relations in this mechanism may be unnecessary in certain implementations. In the next two sections , a further discussion about sensing system and feature extracting & matching strategies will be given.

3 Some Issues

We just showed a general structure of object localization systems. In practice, there are some important issues which must be considered when a real localization system is to be designed.

1. *Real-time execution*: **Hierarchical** control structure has been defined as a standard for telerobot control system architecture [1] and has been adopted by researchers to develop individual telerobot systems such as systems developed at Goddard [27], University of Michigan [36] and *etc.*. The functions of vision system are different at each level. So are the requirements for the object localization algorithms. Usually the higher the level, the slower the completion rate. See table 2 for typical completion rates at each level of telerobot control.

   At the object task planning level, for example, one of the functions of vision system is to recognize the environment. The object localization system, as a part of the vision system, is used to give an approximate measurements of the locations of the objects in the environment. The execution time is in the minute range. At the E-move level, however, the rate of completion is in the range of seconds. If a visual-feedback control strategy is used here, the localization system has to generate updated measurements for the control system to adjust the robot's movement in the same time frame. Real-time issue will become important. Based on different timing requirements, the strategies of localization might be also different.

2. *Accuracy*: Accuracy is another important issue in object localization. There are two definitions of accuracy, e.g., **absolute accuracy** ϵ and **relative accuracy** $\Delta\epsilon$.

   - *Absolute accuracy* is defined as the difference between a measured value m and it's real value s. That is, $\epsilon = m - s$;

	Average rate of change in output	Average replanning interval	Planning horizon
Servo	1 KHz	1 millisec.	15 msec.
Primitive	62Hz	16 millisec.	300 msec.
E-Move	8Hz	128 millisec.	2 sec.
Object/task	1 Hz	1 second	30 sec.
Service Bay	.1 Hz	10 second	> 10 min.
Mission	0.01 Hz	1.7 minutes	> 1 hour

Table 2: The rate of subtask completion at each level of hierarchy. ([1])

- *Relative accuracy* is defined as the difference between a measured difference and it's real difference. For example, if the real difference between two points is $\Delta\rho$ and the two measurements on the two points are s_1, s_2, then the measured difference is a function of s_1 and s_2, e.g., $\Delta\rho' = f(s_1, s_2)$, and the relative accuracy of the measurements is $\Delta\epsilon = \Delta\rho - \Delta\rho'$.

High accuracy, especially high absolute accuracy is not always required. For example, accuracy is not crucial at the beginning of an assembly task, but will be a determining factor in the final stage of operation. Even at that time, the determining factor is relative accuracy rather then absolute accuracy.

Absolute accuracy to a large extent depends on the accuracy of the sensing system. But this does not mean that we do not need a good localization algorithm. A good algorithm should be insensitive to measurement noises, object distortions and other factors which could influence the accuracy of the localization.

The achievement of high relative accuracy, on the other hand, does not necessarily depend on high accurate sensing systems. Human eyes, for example, are not good at locating objects in the absolute sense, but human have no difficulty picking up an object. Research is shown also proved this point of view [25]. Therefore, when designing an algorithm, one must evaluate its performance according to both it's absolute accuracy and relative accuracy, which has been neglected by some researchers.

3. *The type of locatable objects*: It is best if the system can locate arbitrary-shaped objects. If this is difficult, an alternative method is try to find specific detectable features for each object and store these features in that object model for feature-extraction and matching in localization process. If such features do not exist for some object, then one should try to make special marks on the object. Therefore, some guidelines should be given in the component design stage so that the design is favorable to part grasping and localization by the telerobot system. Sometimes, very simple modifications made on the part design can greatly improve part localization process.

4. *Sensing system*: What types of sensing techniques should be used in a localization system? Where should one install the sensing system? How is the dynamic range of sensing system determined? These are just some of the issues when one needs to design a sensing system. Javis [20] has presented an early overview of range finding techniques. Each technique has its advantages and disadvantages. Image-based sensing provides complete information about the environment but takes time to process it. Sparse data can be used directly for fast localization

purpose but needs a good sensing system for fast data acquisition. Multi-spot sensing is an example of such system [21]. For the installation, perhaps some sensors should be installed in fixed locations while others can be put into the robot's moving parts. Newly developed technologies should be used in localization systems. For example, motorized-zoom and auto-focus techniques can improve the dynamic range of measurements; VILS techniques can reduce the size of the whole sensing system. The use of the advanced techniques will have great impact on the design of object localization system.

4 Feature Extraction & Matching Strategies

As we said, the object localization process basically is a feature matching process. That is, finding a best estimate of transformation parameters which will align some modeled object features with certain (perhaps different type of) measured object features. Based on how feature matching is realized, the object localization algorithms can be broadly divided into two categories: the algorithms which do not involve any recognition process, and those which have more or less recognition process involved. We call these two types of algorithms as direct-localization algorithms and recognition-localization algorithms respectively.

Obviously the second type of algorithms has a higher intelligent level than that of the first ones. Even within the second type of algorithms, the intelligent levels could be different. Some of them can establish the matchings within one object, some of them can do it within a group of the same type of objects, others can match the features within a group of different types of objects. At the highest level, the algorithm could locate unmodeled objects. To do this, a set of primitive features should be specified in a database, which will form the basic frames of any object to be constructed. Before localizing the unknown object, the algorithm must explore the object and establish a model for the object using the set of primitives.

Each type of algorithm can be further classified according to their sensing methods, the types of features used for matching, mathematical formulating methods, the types of locatable objects and so on.

4.1 Direct-Localization

Direct-localization algorithms are mostly used in the situations where either the working environment is a highly-structured or the position relationships among the objects in the environment have previously been established proximately or the human beings could provide the assistance as where to take the required measurements. The telerobotics applications in most space programs meet these requirements.

Because no recognition is involved, the localization process is quite simple. The extracted features and model features can be used as inputs for direct computation. The time of localization depends on the time spent on measurements and feature extraction.

One method proposed by Gunnarsson and Prinz [18, 19] is based on their observation that if a set of points are measured and these measurements are distributed on the object surfaces, the best transformation is the one which will make the sum of distances between each measured point and it's corresponding transformed surface minimal. Their idea leads to a point-surface matching strategy. Their algorithm, when formulated in mathematical terms, becomes a least squares minimization problem and can be used to locate arbitrarily-shaped objects. Usually an iterative numerical procedure is needed to solve for the problem. The numerical procedure they used is a modified Lagrange multiplier and Newton-Raphson method. Because a good initial guess can be provided due to the fact that the object's approximate location is supposed known, the convergence of the algorithm is guaranteed in most cases.

Gordon and Seering [17] developed a system which uses striped-light and camera sensing to gather necessary range data. The system can only locate planar objects. Line-surface matching is used in their algorithm. The striped-light when projected on the planar surfaces of the object generates straight-line segments. The scene is then viewed by a camera. The equation of each line segment can be obtained by analyzing the corresponding image of that line segment viewed by the camera. Three independent line segments are needed to compute the rotation and translation parameters. The fact that the line vector is perpendicular to the rotated modeled surface normal vector can be used to derive the rotation. The algorithm uses quaternions to represent rotation and uses a numerical method to compute it. They also give a closed-form solution for the rotation when three surfaces sensed are perpendicular with each other. The calculated rotation is then used to compute the translation.

The same striped-light and camera sensing system is also used by Rutkowski, Benton and *etc.* [3, 28]. But their matching strategy is point-surface matching. In their algorithm, the measured points are from extracted line-segments, either straight or curved. Their method imposes no particular constraints on the shapes of the object surfaces, as long as the object surfaces can be partitioned into a collection of primitive surfaces, such as planes, cylinders, or spheres. The computation is carried out by a repeated location adjustment. The location adjustment is expressed by three quantities: the rotation center, rotation axis and translation vector. To guarantee a fast convergence of their algorithm, the center of mass data points is chosen as the rotation center instead of the origin of the model's coordinate system.

In above methods, if the object is a polygon, at least three surfaces need to be accessed in order to take enough measurements. Shao, Volz and *etc.* [29] have implemented an algorithm based on line-segment line-segment matching, which needs to access only one surface when localizing a planar object. Their algorithm can locate object which has planar surfaces, quadric surfaces and revolutionary surfaces. A line range sensor is used to extract line-segment parameters. The line-segments are either boundary edges or axes. The extraction of only two line-segments are enough to locate an object. Closed-form formulas are used to compute the position and orientation parameters.

When comparing with these methods, we can find out that all of them have very high measurement accuracy and fast execution speed. For example, Gordon's system has 2.5 seconds of execution time and a relative accuracy of 0.002 inches in translation and 0.1 degrees in rotation when a two inch cube is being located, and is capable of reliably assembling components with little clearance without using force controlled motion. In Gunnarsson's algorithm, the measurement error is on the same order of magnitude as the sensor error. These algorithms also have some problems. The problem associated with stripped-light sensing is that it requires extra light source with special pattern, which sometimes is inconvenient. The use of spot sensor or line sensor has the problem of multi-measurement, e.g., the sensor has to be installed on the robot's moving part and be moved together with the robot in order to take multi-measurement. This will slow down the localization process.

High-level features can also be used to locate objects. For example, Thorne and *etc.* [35] described an algorithm which uses features such as the radii or curvatures of a space curve along the curve to locate an object. The curvatures k or radii ρ of a space curve can be expressed as a function of the length s of the curve, e.g., $k = \kappa(s)$ (or $\rho = p(s)$), which is independent of the coordinates of the curve and is thus invariant under rotation and translation. The algorithm assumes that there exists a particular feature line or *fingerprint* for each object. The feature line could be a certain portion of the curved edge(s) of the object or a curve on the object surface. A curvature plot along the feature line can be drawn. In the database, the feature line is specified by a set of discrete points with each point associates with the information about its coordinate (x, y, z), radius of curvature, curvature, delta length, and total length. The total length is zero for the first point. The localization is proceeded through point-point matching. The method first measures a set of discrete points along the feature line and then finds a corresponding point for each measured point and a least squares optimization algorithm is used to find the location parameters. A similar algorithm which uses iso-gaussian (a curve connecting points of constant gaussian curvature) matching to localize an object was described by Gunnarsson [18].

4.2 Recognition-Localization

In many applications, the objects could be placed anywhere in the environment. Therefore, if a measurement is made and some sensed features are extracted, the localization system has no prior knowledge about which object or which part of the object the sensed features should belong to. In this case, in order to compute the location of an object, a recognition process is needed, which will establish the matchings between a set of sensed features and the model features.

There are two popular matching strategies: *tree searching* and *clustering.*

In tree searching strategy, if there are k sensed features $S_i, i = 1 \cdots k$ and l_m model features $M_j, j = 1 \cdots l_m$ for object $O_m, m = 1 \cdots w$, a searching tree can be constructed for each known object O_m such that the tree has l_m levels, and each intermediate node has k branches. Each path from root to leaf represents a potential matching. The total number of possible matchings, or the searching space for object O_m is l_m^k, which is very huge. To reduce the searching space, several methods have been proposed.

One algorithm proposed by Grimson, Lozano-Perez and *etc.* [15] [16] is to use the local geometrical constrains such as distance constraint, angle constraint, direction constraint, triple-product constraint and so on to reduce the searching space. Beginning from the root of the tree down, at each node, local constraint test is made to see if the sensed features up to that level are consistent with these constraints. If it is not, the entire subtree is discarded for consideration.

A similar tree searching method is used in Faugeras and Hebert's work [12, 13, 14]. Instead of local constraints, rigidity is used as the basic constraint during tree search process. Every path from the root to an intermediate node (level k for instance) represents a partial matching. The algorithm computes a best rigid transformation $\mathbf{T}_k$ up to that level (k). Then $\mathbf{T}_k$ is applied to the next unmatched model primitive M_{k+1} and only those sensed primitives that are sufficiently close to $\mathbf{T}_k M_{k+1}$ are considered. The computations are carried out by least squares optimization techniques. As each new pair of primitives adds to the partial matching list, the new estimation of transformation has to be started over again. The algorithm's underlying paradigm is "*locating while recognizing*" which is different from the paradigm of "*locating after recognizing*" used in Grimson and *etc.*'s algorithm.

Reducing the number of sensed and model features is another important method to speed up the tree searching process. The use of higher level features can effectively reduce the size of the searching tree because fewer features are usually adequate. The system developed by Bolles, Horaud and *etc.* [9, 10] is such an example. Three different types of edges are used as the primitive features. They are: straight dihedrals, circular dihedrals, and straight tangentials. They are higher level features: one pair of matched features can determine all but one of the object's six degrees of freedom.

Clustering is another technique used in recognition-localization algorithms. The principle of clustering is very simple:

```
For each element in the sensed feature list
    for each element in the model feature list
        if they are compatible, compute a transformation candidate
        put it into cluster space.
```

The cells with the largest counts are expected to represent the location.

While the principle is simple, the implementation is not so easy. The high dimensions (six) and huge space of clustering are just two difficulties. Different methods have been proposed to accommodate these problems. Three dimensional clustering, the use of proper size of cells and hierarchical clustering are some of them [2]. Several systems have been proposed by using the clustering technique. Linnainmaa *etc.* [23, 24, 25], Silberberg, Harwood, *etc.* [31], and Stockman *etc.* [32, 33, 34] are typical examples. One property of clustering is the algorithm's parallel structure, which will have an important impact on the future development of object localization algorithms.

In most algorithms, the least squares optimization is the mathematical tool to estimate the best transformation if many feature-pairs are found. But in many situations, this method is not the only tool. Bolle, Cooper [5] [6, 7, 8] presented a statistics approach of combining pieces of information to estimate 3-D complex-object position. They formulate the optimal object localization as a Bayesian probability estimation problem. The objective is to find the most likely transformation $\mathbf{T}$ that maps the model primitives onto the measured range data. The likelihood $p(Y|\mathbf{T})$ should be maximized with respect to $\mathbf{T}$, where Y is the measurement data. If κ primitives have been extracted from range data and matched to model primitives, then $p(Y|\mathbf{T}) = \prod_{k=1}^{\kappa} p(Y_k|\mathbf{T})$. That means, to arrive a global optimal solution, the maximum likelihood estimation has to be applied locally. Based on this analysis, they arrived at a different formula for minimizing the estimation error from the traditional least squares optimization formula. To arrive an optimal solution, a through analysis of measurement errors and having a good error model are needed.

5 Summary

We have discussed several object localization methods and strategies. Different levels of telerobot control have different requirements on the localization system. At the low level, the consideration of real-time execution and high accuracy is important. At higher level, the use of AI (artificial intelligence) technology becomes crucial. It seems that a lot of work has to be done in order to develop a real practical localization system. The issues discussed in section 3 are just few of those which need to be addressed by future research.

References

[1] Jamas S.Albus, Harry G. Mccain, and Ronald Lumia, *"NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)"*, National Bureau of Standards, Robot System Division, March 13, 1987, pp.12.

[2] D.H. Ballard *"Parameter Networks: Towards a Theory of Low-Level Vision"*, Int'l Joint Conf. on Artificial Intelligence, Vancouver, B.C., August 1981, pp.1068-1078.

[3] R. Benton and D. Waters, *" Intelligent Task Automation Interim Technical Reports"*, AFWAL/MLTC Wright Patterson AFB, Ohio, Reps. 1-7 Oct. 1985.

[4] Paul Besl and Ramesh Jain, *"An Overview of Three-Dimensional Object Recognition"*, Univ. of Michigan, December 1984, Tech. Rep., RSD-TR-19-84.

[5] Ruud M. Bolle, *"Information Extraction About Complex Three-Dimensional Objects From Visual Data"*, Ph. D. Dissertation, Brown Univ., May 1984, Brown Univ. Tech. Rep., LEMS-6.

[6] Ruud M. Bolle and David B. Cooper, *"On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data"*, IBM Tech. Rep. RC 11037, Brown Univ. Tech. Rep. LEMS-8, Feb. 1985.

[7] -, *"On Parallel Bayesian Estimation and Recognition for Large Data Sets with Application to Estimating 3-D Complex-Object Position from Range Data"*, Proc. SPIE Conf. Vision for Robots, Cannes, France, Dec. 1985.

[8] -, *"On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-8 No.5, Sept. 1986, pp. 619-638.

[9] R. B. Booles, P. Horaud, and M. J. Hannah, *"3DPO: Three Dimensional Part Orientation System"*, Robotics Research: The First Symposium, Cambridge, MA: MIT Press 1984.

[10] P. Horaud, and R. B. Booles, *"3DPO's Strategy for Matching three-Dimensional Objects in Range Data"*, Proc. IEEE 1984 Int'l Conf. Robotics, Atlanta, GA, March, 1984, pp. 78-85.

[11] W.K.Cliford, *"Preliminary Sketch of Bi-Quaternions"*, London Math. Soc., 4:381-395, 1873.

[12] Faugeras, O.D. and Herbert, M., *"A 3-D Recognition and Positioning Algorithm Using Geometrical Matching Between Primitive Surfaces"*, Proc. Eighth Int'l Joint Conf. on Artificial Intell., Aug. 1983, pp.996-1002.

[13] Faugeras, O.D., *"New Steps Toward a Flexible 3-D Vision System for Robotics"*, Proc. 8th Int'l Joint Conf. Pattern Recognition, Montreal, Canada, July-Aug., 1984, pp. 796-805.

[14] Faugeras, O.D. and Herbert, M., *"The Representation, Recognition, and, Locating of 3-D Objects"*, The Int'l Journal of Robotics Research, Fall 1986, pp.27-52.

[15] Peter C. Gaston and Tomas Lozano-Perez, *"Tactile Recognition and Localization Using Object Models: The Case of Polyhedra on a Plane"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No.3 May 1984.

[16] W.Eric L. Grimson and Tomas Lozano-Perez, *"Model-Based Recognition and Localization from Sparse Range or Tactile Data"*, The Int'l Journal of Robotics Research, Fall 1984 pp.3-35.

[17] Steven J. Gordon and Warren P. Seering, *"Real-Time Part Position Sensing"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI 10, No.3 May 1988.

[18] Kristjan T.Gunnarsson, *"Optimal Part Localization by Database Matching with Sparse and Dense Data"*, Ph.D. dissertation, Dept. of Mechanical Engineering, Carnegie Mellon Univ., May 1987.

[19] Kristjan T. Gunnarsson and Friedrich B. Prinz, *"CAD Model-Based Localization of Parts in Manufacturing"*, Computer, August 1987, pp.66-74.

[20] R. A. Jarvis *"A Perspective on Range Finding Techniques for Computer Vision"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.2, March 1983, pp. 122-139.

[21] Takeo Kanade, *"Three-Dimensional Machine Vision"*, Kluwer Academic Publishers, 1987.

[22] P. Kelland and P.G. Tait, *"Introduction to Quaternions"*, Macmillan and Co., 1882.

[23] S. Linnainmaa, D. Harwood, and L.S. Davis, *"Pose Determination of a Three-Dimensional Object Using Triangle Pairs"*, CAR-TR-95, Center for Automation Research, University of Maryland, 1985.

[24] -, *"Triangle-Based Pose Determination of 3-D Objects"*, Int'l Conf. on Pattern Recognition, 1986, pp. 116-118.

[25] -, *"Pose Determination of a Three-Dimensional Object Using Triangle Pairs"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Volume 10, No. 2, September, 1988, pp. 634-648.

[26] Edward Pervin and Jon A. Webb, *"Quaternions in Computer Vision and Robotics"*, Tech. Rep., Dept. of Computer Science, Carnegie-Mellon University, CMU-CS-82-150, 1982.

[27] *"Robotic Assessment Test Sets: Level 1, 2, & 3"*, NASA Gorddard, SS-GSFC-0029.

[28] W.S. Rutkowski, R. Benton, and E.W. Kent, *"Model-Driven Determination of Object Pose for a Visually Served Robot"*, IEEE Int'l Conf. Robotics and Automation, Raleigh, NC, 1987, pp.1419-1428.

[29] L. Shao, R.A. Volz, and M.W. Walker, *"3-D Object Location Determination Using Line-Segments Matching"*, IEEE Int'l Conf. Industrial Application of Machine Vision, Tokyo, Japan, 1989.

[30] S. Shekhar, O. Khatib and M. Shimojo, *"Sensor Fusion and Object Localization"*, IEEE Int'l Conf. Robotics and Automation, San Francisco, CA, 1986, pp.1623-1628.

[31] T.M.Silberberg, D. Harwood, and L.S. Davis, *"Object Recognition Using Oriented Model Points"*, Computer Vision and Image Processing, Vol. 35, 1986, pp. 47-71.

[32] G. Stockman and J.C.Esteva, *" Use of Geometrical Constrains to Determine 3-D object Pose"*, Proc. 7th Int'l Conf. on Pattern Recognition, Montreal, Canada, 1984, pp. 742-744.

[33] G. Stockman and J.C.Esteva, *" 3-D Object Pose from Clustering with Multiple Views"*, Pattern Recognition 3, 1985, pp. 279-286.

[34] George Stockman, *"Object Recognition and Localization via Pose Clustering"*, Computer Vision, Graphics and Image Processing Vol. 40, 1987, pp.261-387

[35] H. Thorne, F.B. Prinz, and O.K.Kirchner, *"Robotic Inspection by Database Matching"*, Tech. Rep., CMU-RI-TR-85-4, The Robotics Institute, Carnegie Mellon University, March, 1985.

[36] M.M. Walker, J. Dionise *"On the Simulation of Space Based Manipulators with Contact"*, NASA conf. on Space Telerobotics, January 1989.

[37] M.W. Walker, L. Shao and R. Volz *"Optimal Object Localization Using Dual Number Quaternions"*, IEEE Int'l Conf. on Applications of Artificial Intelligence, March 1989.

N90-29021

A LASER TRACKING DYNAMIC ROBOT METROLOGY INSTRUMENT

G.A. Parker, J.R.R. Mayer

Department of Mechanical Engineering
University of Surrey
Guildford, Surrey, GU2 5XH. U.K.

Abstract

Research work over several years has resulted in the development of a laser tracking instrument capable of dynamic 3D measurements of robot end-effector trajectories. We describe the instrument characteristics and experiments to measure the static and dynamic performance of a robot in an industrial manufacturing environment. We speculate on the use of this technology for space applications.

1. Introduction

Robots developed for space applications are likely to be significantly different from their earth-bound counterparts. Structural weight will be as low as possible, consistent with adequate stiffness, with arms designed for a wide variety of working volumes. At one extreme we have loading/unloading arms for use with the space shuttle and at the other we have delicate, miniature arms for small-scale space laboratory experiments. In addition, the dynamic speed requirements of space robots are likely to remain much lower than industrial robots to cope with inertia re-action problems.

To achieve overall improvements in such robots the designer will address himself to such characteristics as machine structure, drive characteristics, adaptive control of the servomechanism for each articulation, software limitations, and so on. All of these areas to a greater or lesser extent require sophisticated measurement techniques to validate the design approach and provide insight into the deficiencies of present robots. A particularly good example of this is the significant variation in the dynamic characteristics of most robots within the working volume as a result of both load and positional changes. Good instrumentation is vital to achieve an accurate experimental description of robots under these conditions.

Many different non-contacting techniques have been investigated for 3D dynamic metrology related to robot technology [1-10]. Of these, the three most promising approaches for industrial applications are the camera based lateral effect photodetector [5], the laser interferometer [10] and the laser triangulation tracking system [9]. For space applications, the latter method has particular attractions due to its large static and dynamic measurement range together with its robust measurement and control aspects. The next section describes the characteristics of an instrument developed at Surrey University based on tracking laser triangulation principles.

2. The Laser Tracking Instrument Concept

The measurement instrument is based on a two-dimension measuring unit or sub-system which can be combined with up to three other units to provide three-dimensional positional information. A sub-system has an opto-mechanical laser beam stearing mechanism which is electronically controlled and is linked to a general purpose micro-computer. Each sub-system can track at high speed the position of an optically passive retroreflective target, attached to the robot arm, by aiming a beam of collimated coherent light at its optical centre. The result is a line-of-sight along which the target is known to lie but which does not inherently provide range information. Figure 1 illustrates the use of a single sub-system to collect information from movements which are constrained to a defined surface, generally a plane perpendicular to the line-of-sight. Possible applications of a single unit could include modal analysis where the presence of defects in the robot structure or controller are identified through monitoring the arm response to white noise input commands and analysing the arm movement data using coherence techniques. [11]

Figure 2 shows a more general arrangement which aims at measuring the x, y, z position co-ordinates of the robot end effector. The addition of a second sub-system which tracks the same point as the first sub-system provides a further line of sight definition that can be used to provide range information through a triangulation calculation that results in an absolute measurement. An interruption of one or both beams only requires the re-establishment of tracking without loss of calibration. With the present equipment this requires approximately 1 ms to implement. A superfluous 4th datum is available and is used to assess the quality of the triangulation calculation. The tracking instrument and data collection are controlled by a micro-computer which has interactive graphics to provide the results of the calculation in an understandable format. Mass storage units, such as a tape streamer, a flexible disk drive and a hard disk are also available for data storage so that data averaging, filtering, and so on may be carried out on several tests.

For certain applications it might be necessary to provide not only information about the end effector's position but also about its orientation. For this purpose two more sub-systems are used. Two configurations can be adopted. One configuration consists of using two sub-system pairs, with each pair monitoring the co-ordinates of one target of a pair. This provides 5 degrees of freedom i.e. x, y, z and horizontal and vertical angles of the target pair using a linear calculation. The other configuration consists of each sub-system monitoring one target of a cluster of four. A non-linear calculation results in a measure of all 6 degrees of freedom of the robot end effector.

3. System Calibration

The instrument can only provide high accuracy and repeatability if the various constituent components of the system are modelled and calibrated. This must be performed at three levels; at the individual measuring component level, at the sub-system level, and at test site level. Of these, the sub-system is the most complex and will be considered first. Figure 3 shows the optical components of a sub-system which consist of a linearly polarised laser, a quarter-wave plate to optically isolate it, and a collimator to expand it to a convenient diameter. To deflect the transmitted beam to the target, there are two orthogonal plane mirrors attached to moving-iron galvanometer scanners. The reflected beam from the optical target returns parallel to the transmitted beam to impinge on the photodiode quadrant detector. The deflection of this beam from the centre of the detector is a measure of the tracking error which is used to drive the scanners to reduce this error in a closed control loop.

To calibrate a sub-system requires the calculation of the line-of-sight equation to the target based on the knowledge of the two scanning mirror angular positions as well as the lateral distance between the outgoing laser beam and target centre point. These numerical values are used with a geometric model of the sub-system. The model accounts for any possible departure of the real tracking head from the nominal design dimensions and is in parametric form. This makes it suitable for least square optimisation, in the case of dimensions not easily measured using direct methods.

The basic system components, such as the scanner transducer and quadrant detector photocell, are calibrated individually under conditions reflecting their actual use to accuracies in excess of that expected for the system components.

The determination of the relative position and orientation of the two sub-systems at a test site uses an indirect method. It involves, prior to performing actual tests, the measurement of two targets attached to a calibrated bar as well as the measurement of a set of random and unknown 3D positions. All six degrees of freedom are determined this way. The bar is made of carbonfibre which has a temperature expansion coefficient of $-0.7 \times 10^{-6}/°C$, thus providing good position stability.

In addition, the overall measurement uncertainty due to basic component errors has been assessed on the assumption that the sub-system and site calibration have been adequately performed. As triangulation involves an angular measurement, any angular error of the scanner system is crucial. Figure 4 shows the contribution of the scanner angular position measurement error to the measurement uncertainty based on 1 arc sec of scanner error with the sub-systems separated by 1 metre. Uncertainties are calculated in metres and increase linearly with separation distance and scanner error.

Overall calibration of the present system reveals a repeatability of better than ± 0.1mm in x, y, z directions for one standard deviation based on 30 tests. The tests were repeated at twenty nominal positions along a straight-line precision slideway equipped with a linear optical encoder. The total distance between the positions was 0.8m.

4. Some instrument results

Figure 5 shows the results produced by the laser measuring instrument for the accuracy and repeatability testing of an industrial robot. The test consists of a cycle of 5 points, repeated 30 times. These results are for one of the points. All data is referred to the robot reference frame to ease its interpretation. Numerical data is also provided with information on the statistical spread of the repeated positioning as well as the cartesian difference between the demand and mean attained positions. Figure 6 shows the results of a trajectory test where the robot must describe a rectangular path three times. The measured trajectory is projected on to the three cartesian planes of the robot reference frame. Figure 7 shows the results for the same dynamic test but as a function of time.

Typically for a measurement volume of 1 m^3 the measurement is currently achieving tracking speeds in excess of 3 m/sec with a measurement accuracy of 0.5 mm and a repeatability of $\pm$ 0.1 mm. The repeatability is reduced to 20 μm for a stationary target by taking the mean of 30 readings at a sampling rate of 200 Hz. The target can be tracked from 0.5 to 6 metres away from the sub-systems in the laboratory giving a variable measurement volume of approximately 0.01 to 27 m^3. The rangeability (the measurement range/resolution) is typically 10,000:1. Current work shows that improving rangeability by a factor of 5 can be achieved if required.

5. Potential space-station applications for laser triangulation

5.1 General

The permanently manned Space Station project being developed by NASA, with the participation of Canada, Europe and Japan, provides a unique opportunity to develop a wide range of automatic and robotic concepts in space. This should improve productivity, reliability, safety and give greater system flexibility. As far as the U.K. is concerned, the Department of Trade and Industry is sponsoring an Advanced Robotics Initiative in Space Applications as part of the European Programme. A proposal is being considered for the development of an Internal Experiment Manipulator (IEM) as a space laboratory work-cell demonstrator by a consortium headed by Logica.

The development of robots in space will initially use tele-operation under direct astronaut control with force reflected master-slave control. These systems will be capable of performing such tasks as removing and installing fasteners and umbilical cords, routine maintenance, space station construction and so on. The NASA/Johnson Space Center approach using the Shuttle Remote Manipulator System for Space Station assembly is a good illustration of tele-operated robot development.

As robots develop further there will be an evolutionary change towards autonomous robots. A range of sophisticated sensors will provide the robot with environmental and task information while the astronaut acting as a supervisor defines the task, monitors the robot and resumes control after the task is completed. Part of the NASA program foresees a Space Station Mobile Remote Manipulator System which can undertake autonomously Station assembly, Station/satellite maintenance and repair, and routine inspection.

Most of the external tasks required for robotic operations at the Space Station may be grouped under the headings of assembly of space structures, maintenance and repair, inspection. In addition, there will be internal tasks as the Space Station will support a variety of laboratories operating under microgravity conditions. Many of these laboratories will use low-reaction robots [12] operating at relatively high speed in an ordered environment not unlike industrial situations on earth.

All these tasks will require a wide variety of sensory information in which vision techniques will be predominant [13], particularly as robots become more autonomous. However, there is an important requirement for a range of non-contact position measurement and control which may be met optically without recourse to the complexity of full vision information processing. Thus applications such as docking, automatic and manual steering of remote manipulators, robot calibration in space laboratories are very suitable for laser tracking and triangulation technology. This approach provides absolute measurement, is robust from a control point of view, has a wide range of static and dynamic characteristics and has acceptable accuracy. Some indications of the approach to be used in these application areas are now considered.

5.2 Docking

Krishen [13] provides information on laser docking system performance goals (Table 1) and suggests that in docking and berthing applications a robotic vision/sensing system may be needed within a cone of 30 deg. to a distance of 50 m. Beyond this zone it is envisaged that a radar system may be used for tracking and monitoring the object motion. From the work on the laser tracking system described in Sections 2 to 4 the range and range rate accuracy requirements are well within the capability of the present electro-optic technology. The angular resolution requirements for bearing and attitude are also not very stringent but the angular rate resolution of 0.002 deg/s is quite demanding but achievable. Depending on the complexity of the docking configuration, a 2 or 4 laser beam configuration would provide all the measurement information required and could probably work satisfactorily at distances further away than 50 m.

5.3 Automatic and manual steering of remote manipulators

Despite astronaut tele-operator control of remote manipulators at the present time, there is probably scope for improvements in the speed and accuracy of task performance. This is mainly due to the complexity and flexibility of the manipulator arm structures used together with the range of loads carried by the arms. Improvements can be made to the arm dynamics by detailed mathematical modelling of its characteristics and the use of complex control laws. However, a more straightforward approach is possible by the direct positional control of the end-effector using on-line laser tracking and triangulation. The static and dynamic characteristics are well within the capability of such systems (see Section 4) and on-line control at the trajectory velocities required can be implemented with conventional microprocessor technology. Some consideration would need to be given to determine the optimal control strategy for each joint to follow the overall position demand. For tele-operator control of the position loop the demand information would be generated directly from the joy-stick.

5.4 Robot calibration in space laboratories

It is envisaged that a robot arm might have to move between several work-cells within a laboratory as well as the manipulation tasks within each cell. Many experiments will require a very high degree of isolation from reactive forces generated by the robot, Space Station, etc. thus requiring correction between the manipulator's reference co-ordinate frame and that of the work-cell. Ideally the calibration sensor system should be based on the work-cell so that it measures the position of the manipulator end-effector relative to the experiment. The advantage of this method is that it compensates for any possible errors in the robot structure and controller together with any relative movement between the robot and experiment base plants. Again, for the distances and accuracies involved, triangulation devices can be in-built to each work-cell to provide on-line calibration.

6. Conclusions

An industrial laser tracking instrument working on triangulation principles has been described together with some of its characteristics. It can provide absolute positional and orientation information, its rangeability is good, it has robust tracking control and has the necessary resolution to meet a significant number of space sensory and control requirements. A number of applications have been discussed where static and dynamic metrology can compliment the more sophisticated vision sensing developments required for space tasks.

Acknowledgements

The authors would like to acknowledge the financial support of the Bureau of Community Reference, Commission of the European Community, Brussels, directed by Dr D Gould; also of the Department of Trade & Industry, U.K., through Mr R Eggington

References

1. Warnecke, J.J. and Brodbeck, B., "Test Stand for Industrial Robots", 7th International Symposium on Industrial Robots, Tokyo (Japan), Oct 1977, pp. 443-451 .

2. McEntyre, R.H., "Three Dimension Accuracy Measurement Methods for Robots", The Industrial Robot, September 1976, pp. 105-112 .

3. Desmaret, J.P., "Methods de mesures tridimensionnelles a l'aide de deux theodolites", Direction des Techniques Avancees en Automatisme, Groupe Mesure, October 1981.

4. Fohanno, T., "Assessment of the Mechanical Performance of Industrial Robots", 12th International Symposium on Industrial Robots, 1982, pp. 349-358.

5. Selspot II - a complete system for sophisticated motion analysis, Sweden, SELCOM AB, Box 250, S433 25, Partille.

6. CODA 3 Movement Monitoring System, Movement Techniques Ltd., Unit 5, The Technology Centre, Epinal Way, Loughborough, Leicestershire, LEll OQE, England.

7. 3-Dimensional Sonic Digitiser, Model GP-8-3D, Science Accessories Corporation SAC, 970 Kings Highway West, Connecticut 06490, USA.

8. Burton, R.P. and Sutherland I.E., "Twinkle Box - A Three dimensional computer input device", National Computer Conference, AFIPS Press, 1974, pp. 513-520.

9. Gilby, J.H. and Parker, G.A., "Robot Arm Position Measurement Using Laser Tracking Techniques", 7th Annual British Robot Association Conference, 1984.

10. Lau, K.; Hocken, R. and Haight, W., "An automatic laser tracking interferometer system for robot metrology", 3rd Int. Precision Engineering Seminar, Interlaken, Switzerland, May 1985, pp. 100-102.

11. Dagalakis, N.G., "Analysis of Robot Performance Operation", 13th International Symposium on Industrial Robots, 1983, pp. 7-73 to 7-95.

12. Rohn, D.A.; Lawrence, C. and Brush, A.S., "Microgravity robotics technology programme", ISA 88, Houston, U.S.A., October 1988, Paper 88-1642.

13. Krishen, K., "Space Robotic Vision System Technology", ISA 88, Houston, U.S.A., October 1988, Paper 88-1615.

PARAMETER	LIMITS	ACCURACY (σ)	
Range (R)	0-1 km (3280 ft)	.01 R; 2.5 mm ≤ 10 m	
Range Rate	±3 m/s (±10 ft/s)	.0001 R/s; 3 mm/s ≤ 30 m	
Pointing	±π/2 rad (±90°)		
Bearing Angle	± .2 rad (±10°)	3 mrad (.2°)	
Bearing Angle Rate	±20 mrad/s (±1°/s)	.03 mrad/s (.002°/s)	
Attitude (P.Y)	± .5 rad (±28°)	7 mrad (.3°)	at R≤100 ft
Attitude (R)	±πrad (±180°)	7 mrad (.3°)	at R≤100 ft
Attitude Rate	±20 mrad/s (±1°/s)	.03 mrad/s (.002°/s)	at R≤100 ft
R, $\dot{R}$ Output Data Rate	1 Hz		
Angle Output Data Rate	3.125 Hz		

Table 1: Laser Docking System Specification
(Ref 13)

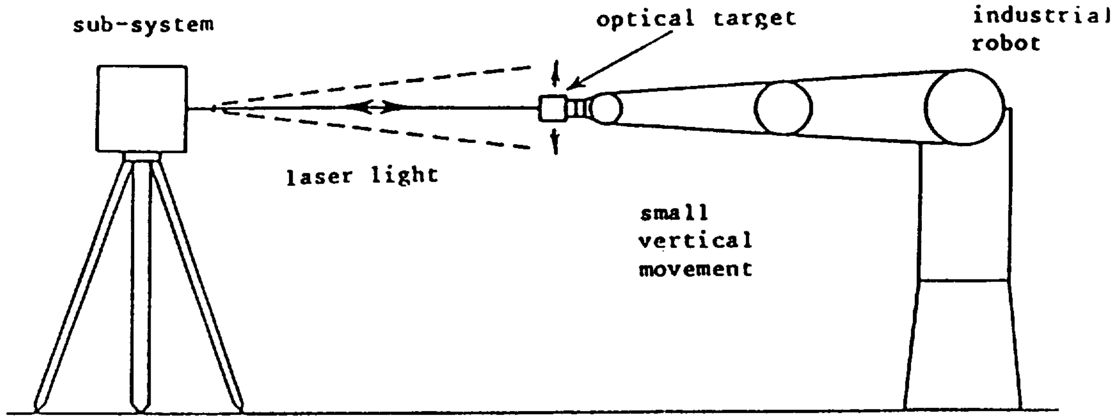


Figure 1 - Use of a single sub-system for measurements in a plane

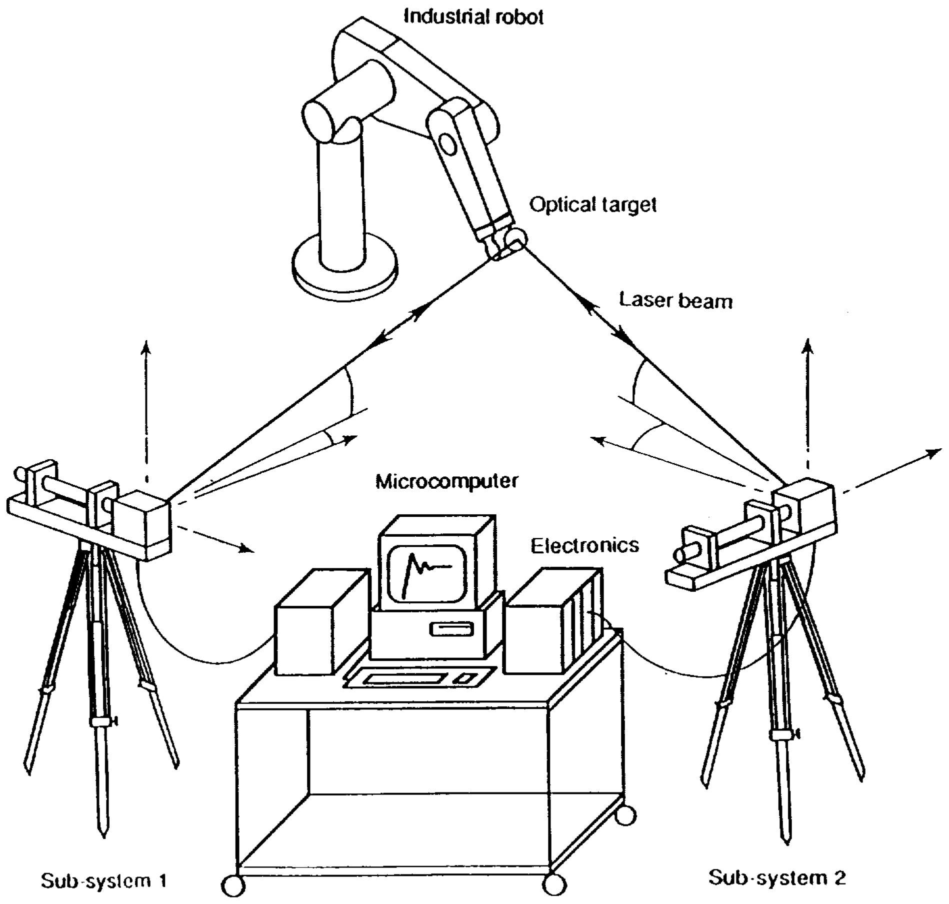


Figure 2 Diagram of the Laser Tracking Triangulation Method

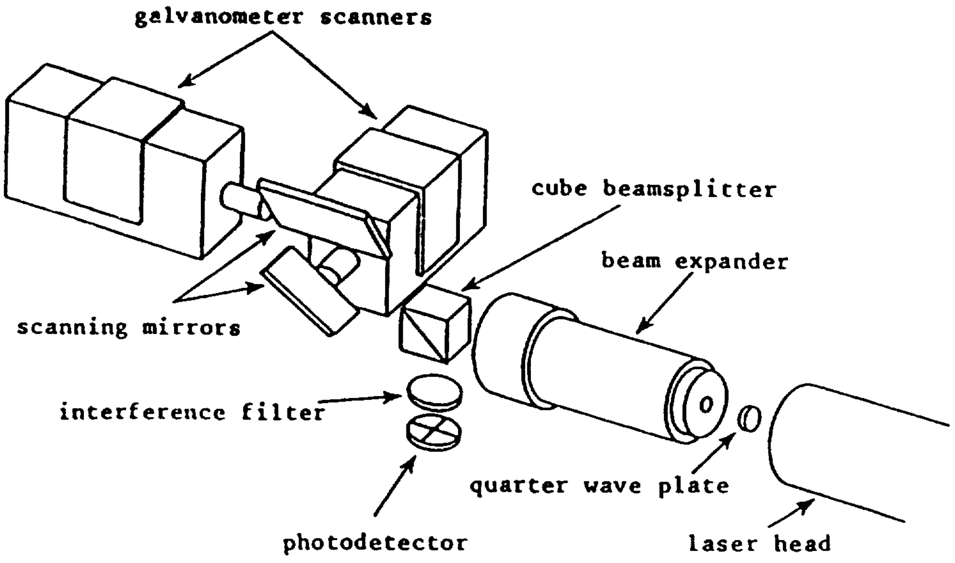
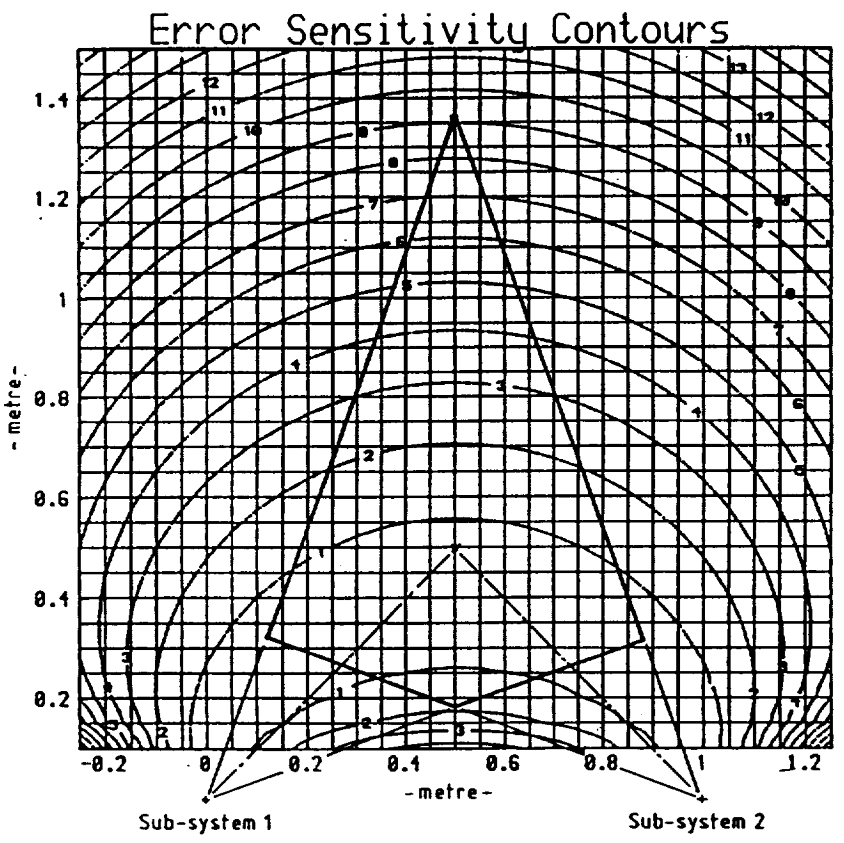


Figure 3 - The Optical Hardware of a Sub-System



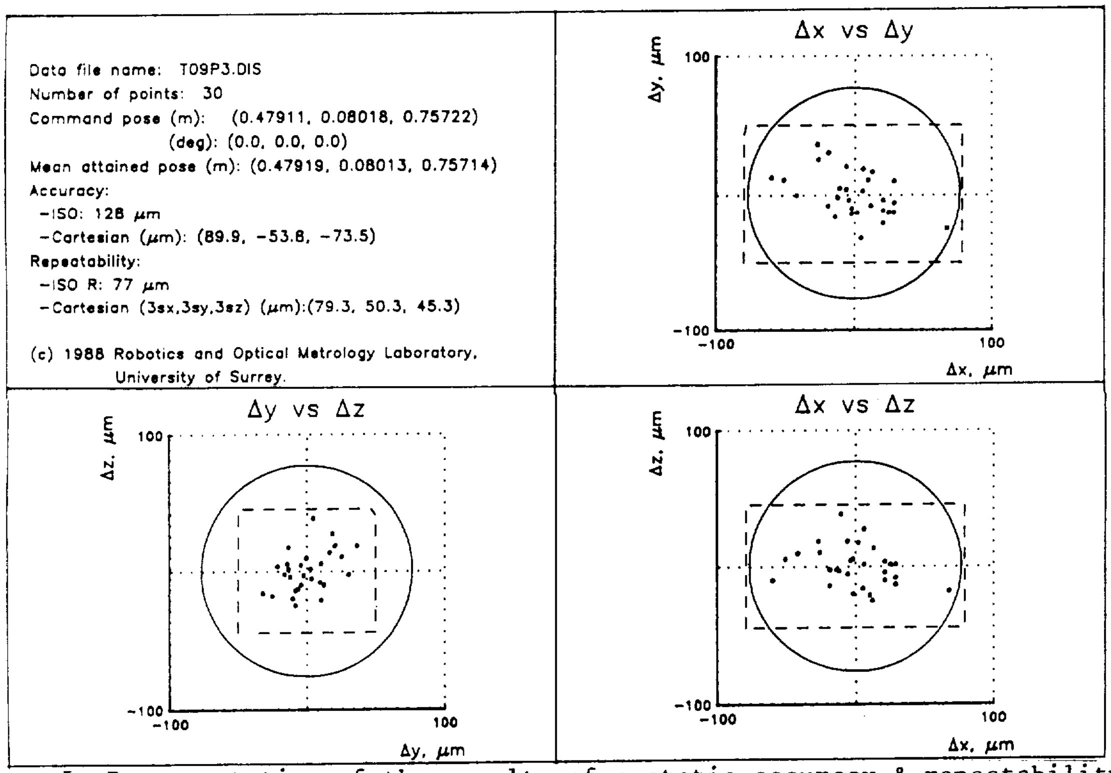
Position resolution (Metres)

CONTOUR KEY	
1	0.131E-04
2	0.168E-04
3	0.204E-04
4	0.241E-04
5	0.278E-04
6	0.315E-04
7	0.352E-04
8	0.389E-04
9	0.426E-04
10	0.463E-04
11	0.499E-04
12	0.536E-04
13	0.573E-04
14	0.610E-04
15	0.647E-04

Scanner accuracy = 1 sec.

Baseline length = 1 m.

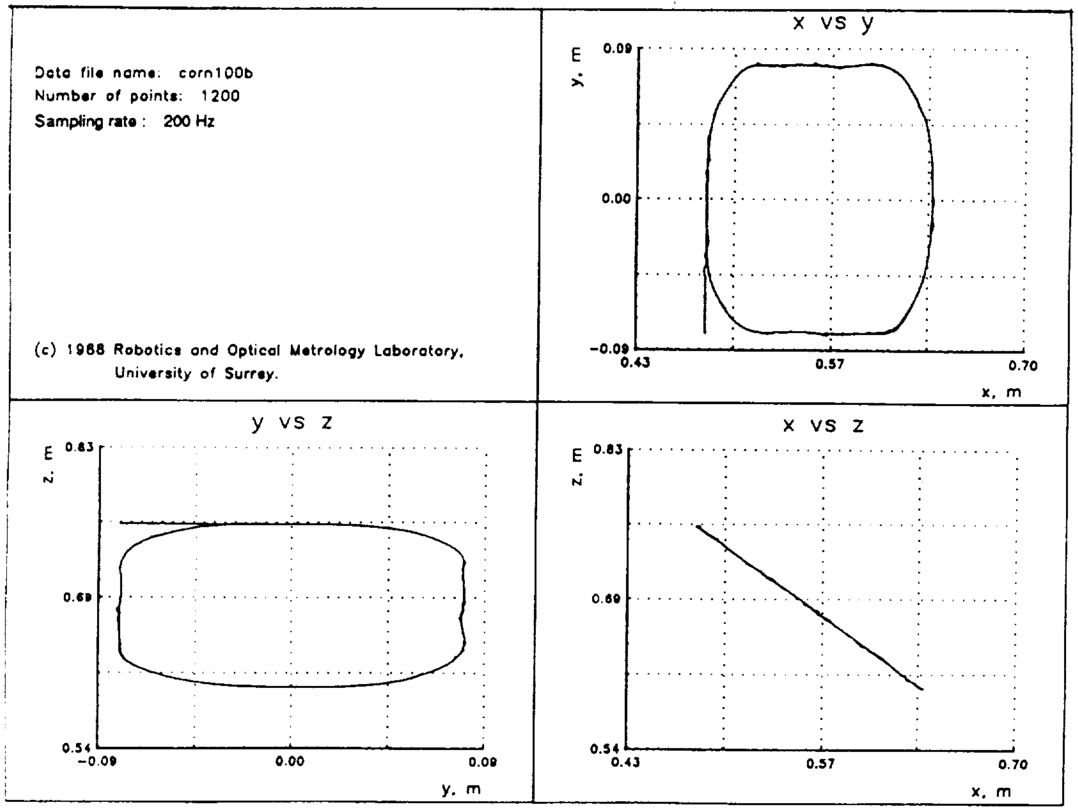
Figure 4 Effect of scanner angular uncertainty on the final co-ordinate measurements



Data file name: T09P3.DIS
Number of points: 30
Command pose (m): (0.47911, 0.08018, 0.75722)
(deg): (0.0, 0.0, 0.0)
Mean attained pose (m): (0.47919, 0.08013, 0.75714)
Accuracy:
-ISO: 128 μm
-Cartesian (μm): (89.9, -53.8, -73.5)
Repeatability:
-ISO R: 77 μm
-Cartesian (3sx,3sy,3sz) (μm):(79.3, 50.3, 45.3)

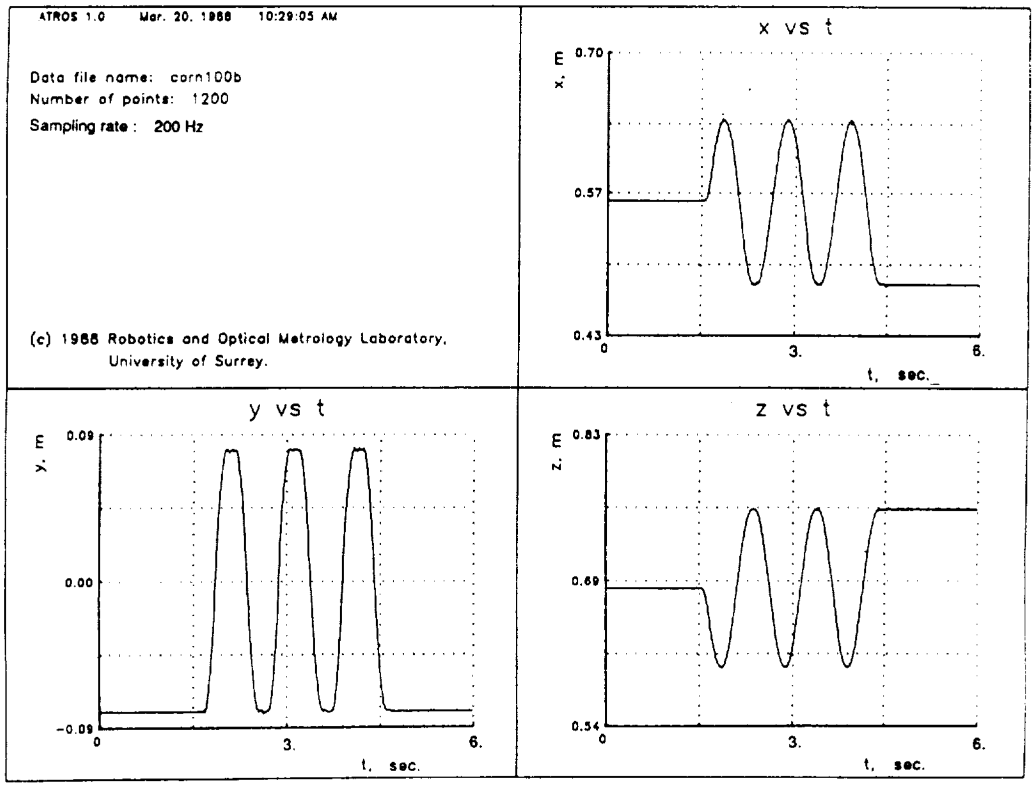
(c) 1988 Robotics and Optical Metrology Laboratory, University of Surrey.

Figure 5 Representation of the results of a static accuracy & repeatability test


Data file name: corn100b
Number of points: 1200
Sampling rate : 200 Hz

(c) 1988 Robotics and Optical Metrology Laboratory,
University of Surrey.

Figure 6 Spatial representation of the results of a dynamic cornering test


ATROS 1.0 Mar. 20. 1988 10:29:05 AM

Data file name: corn100b
Number of points: 1200
Sampling rate : 200 Hz

(c) 1988 Robotics and Optical Metrology Laboratory,
University of Surrey.

Figure 7 Temporal representation of the results of a dynamic cornering test

N90-29022

Robot Acting on Moving Bodies (RAMBO): Interaction with Tumbling Objects

Larry S. Davis, Daniel DeMenthon, Thor Bestul, Sotirios Ziavras, H.V.Srinivasan, Madhu Siddalingaiah, and David Harwood

Computer Vision Laboratory
Center for Automation Research
University of Maryland, College Park, MD 20742

Abstract

Interaction with tumbling objects will become more common as human activities in space expand. Attempting to interact with a large complex object translating and rotating in space, a human operator using only his visual and mental capacities may not be able to estimate the object motion, plan actions or control those actions.

We are developing a robot system (RAMBO) equipped with a camera, which, given a sequence of simple tasks, can perform these tasks on a tumbling object. RAMBO is given a complete geometric model of the object. A low level vision module extracts and groups characteristic features in images of the object. The positions of the object are determined in a sequence of images, and a motion estimate of the object is obtained. This motion estimate is used to plan trajectories of the robot tool to relative locations nearby the object sufficient for achieving the tasks.

More specifically, low level vision uses parallel algorithms for image enhancement by symmetric nearest neighbor filtering, edge detection by local gradient operators, and corner extraction by "sector filtering". The object pose estimation is a Hough transform method accumulating position hypotheses obtained by matching triples of image features (corners) to triples of model features. To maximize computing speed, the estimate of the position in space of a triple of features is obtained by decomposing its perspective view into a product of rotations and a scaled orthographic projection. This allows us to make use of 2D lookup tables at each stage of the decomposition. The position hypotheses for each possible match of model feature triples and image feature triples are calculated in parallel. Trajectory planning combines heuristic and dynamic programming techniques. Then trajectories are created using dynamic interpolations between initial and goal trajectories. All the parallel algorithms run on a Connection Machine CM-2 with 16K processors.

1 Introduction

The problem of robotic visual navigation has received considerable attention in recent years, but research has mostly concentrated on operations in static environments [1–11]. The area of robotics in the presence of moving bodies has seen little activity so far [12–16]. We are developing a control system which should allow a robot with a camera to accomplish a sequence of actions on a moving object.

One primary application for this type of research could be the development of an autonomous vehicle able to develop strategies for intercepting a moving target on the ground such as another vehicle. But our approach seems general enough to be applied to other domains such as robotics in space. For example, a robotic arm building a structure in the absence of gravity might require the capability of interacting autonomously with moving objects. During a teleoperated assembling process one of the building elements could break loose from the gripper. Then the natural motion of this element would be a translating tumbling motion, and there might be very little time to react before the structural element is out of reach of the robot and lost in space. A human operator would have little chance to estimate the object motion, plan the actions required to bring the robot gripper to the right gripping spot and orientation along the moving element, and complete these actions. However, with the proper equipment, the operator could immediately switch to a mode in which the robot is on its own for recovering the tumbling element in the short time available. To be able to handle such situations without human intervention, the robot could be equipped with a video camera, and could have a database describing the geometry of all the types of structural elements being assembled, with the various goal points which could be reached for a proper grip. While in teleoperating mode, the robot could keep track of which structural element is being handled, so that in case of an emergency the robot would already have retrieved all the relevant information from its database. Analyzing a sequence of images, the robot could find the trajectory of the element in location and orientation. Extrapolating the trajectory to the immediate future, it would plan its own motion to bring its gripper along the trajectory of a goal point of the element. Along this goal trajectory the moving element appears fixed with respect to the gripper, so that the gripping action can be accomplished as if in a static environment.

We have set up an experimental facility which has the necessary components for testing various vision-based control algorithms for intercepting moving objects. These algorithms could be incorporated into the navigation system of a vehicle able to intercept other vehicles, or in a robotic system able to recover objects which are tumbling freely in space . This facility is described in the following section.

2 Experimental set-up

A large *American Cimflex* robot arm, RAMBO, is equipped with a CCD camera and a laser pointer (Figure 1, top left). Images from the camera are digitized and sent to the Connection Machine for processing. A smaller robot arm (*Mitsubishi RM-501*) translates and rotates an object (called the *target* in this paper) through space. Several light-sensitive diodes with focusing optics are mounted on the surface of the object. RAMBO's goal is to hit a sequence of diodes on the moving object with its laser beam for given durations, possibly subject to overall time constraints. Electronics inside the object signal success by turning on an indicator light. Simultaneously, we are developing a full computer simulation in which the camera inputs are replaced by synthetic images (Figure 1, top right).

3 Summary of Operations

The vision-based control loop for RAMBO is shown in Figure 1. We briefly describe the functions of the different modules of this system from data collection to robot motion control, and refer to the sections of this paper which give more details.

1. The digitizer of the video camera mounted on the robot arm can grab video frames when new visual information is needed. A database contains a list of positions of feature points on the target, in the local coordinate system of the target.

2. A low-level vision module extracts locations of feature points from the digitized image (Section 4).

3. An intermediate vision module finds the location/orientation of the target in the camera coordinate system (Section 5).

4. Since the past camera trajectory is known, the position of the camera in the robot base coordinate system when the frame was grabbed is known. The location/orientation of the target is transformed to the robot base coordinate system (Section 6).

5. This most recent target pose at a specific time is added to the list of target poses at previous times. In the Target Motion Predictor, a target trajectory in location/orientation space is fitted to these past target poses

and extrapolated to the future to form a predicted target trajectory. We also obtain the predicted trajectories of *goal points* around the target. A goal point is a location –which is fixed in the frame of reference of the target, thus moving in the frame of the robot base– that one of the joints of the robot has to follow for the accomplishment of one of the subtasks of the total action (Section 6).

6. From the predicted goal point trajectories, the Robot Motion Planner calculates the robot motions necessary for following the goal points, and the resulting camera trajectories (Sections 7, 8, 9). If the subtasks are not ordered, the Motion Planner finds an optimal order (Section 10). The camera trajectories are used for transforming subsequent target pose estimates from a camera coordinate system to an absolute coordinate system (Section 6).

4 Low-level Vision

The model-based pose estimation described in the next section requires that feature points be extracted from each of the images of the target. This is the task of the low-level vision module. Feature points in an image could be images of small holes in the target structure, corners of letters, vertices, etc, ... Conversely, the geometric description of the target should contain the 3D locations of the feature points which are easily detected in images.

In our experiments, the target is a polyhedra, and the feature points that we use are the vertices of the polyhedra, so that our image analysis is partly specific to this type of feature detection. Our image processing algorithms involve a sequence of basic local operations [17, 18] implemented on the Connection Machine –enhancement [19], edge detection, edge thinning and vertex detection– followed by some simple processing to determine which pairs of vertices are connected by edges in the image. This last operation proceeds as follows:

Given k vertices, we use the processing cells in the upper-triangle of a $k \times k$ array of cells in the Connection Machine, each assigned to a possible edge between vertices.

1. Enable a grid of $k \times k$ cells. Disable cells in the lower triangle of the array.

2. Copy the addresses of corner points and the incident angles of their edges to the cells in the diagonal of the grid

3. By horizontal grid-scan and vertical grid-scan, spread the incident angles and the addresses of the vertices from the diagonal cells along rows and columns to all the cells in the array.

4. Each non-diagonal active cells (i, j) now has all the information about the i-th and j-th vertices; these cells can determine whether they have a pair of collinear incident edges. If they do, then that vertex pair is marked as being connected (Note that we could also count the number of edge pixels along the line joining the vertices, but this would be much more costly on the Connection Machine and not worthwhile for our purposes).

From this algorithm we obtain a list of all the vertices with for each vertex a sublist of the vertices connected to it. We then produce a list of all the image triples consisting of one vertex with two vertices connected to it. This list of image triples is input to the intermediate-level vision module described in the next section. The other input is a similar list for the triples of world vertices of the target, from the geometric database describing the target.

5 Intermediate-level Vision: Pose Estimation of the Target

The pose estimation algorithm combines three ideas:

1. Pose estimation by matching triples of image features to triples of target features[20].

2. Standard camera rotations [21].

3. Paraperspective approximation to perspective projection [22, 23].

This combination allows the extensive use of 2D look-up tables to replace the costly numerical computations used by similar previous methods [20, 24–26]. The algorithm is implemented on the Connection Machine. Details are given in [27].

The feature points detected in an image are grouped into triples (the *image triangles*). Each image triangle can be described by one of its vertices (the *reference vertex*), the length of the two adjacent sides, and the angle between them (the *reference angle*). These adjacent sides do not necessarily have to correspond to actual edges in the image, and all distinct triples of points could be considered, with each triple of points producing three such image triangles. However, if the feature points are vertices, it is useful to only consider image triangles in which the adjacent edges of the reference vertex are actual edges, and to only match these image triangles to world triangles with similar characteristics. This increases the proportion of good matches over the total number of possible matches.

The main algorithm steps are as follows:

1. Each image triangle is transformed by a *standard rotation*. The standard rotation corresponds to the camera rotation around the center of projection which brings the reference vertex of the triangle to the image center. For each reference vertex in the image, rotation parameters are read in a 2D lookup table.

2. Image triangles are then rotated in the image plane around the reference vertex (located at the image center) to bring one edge into coincidence with the image x-axis.

3. Once in this position, an image triangle can be described by three parameters only, the reference angle, the edge ratio (ratio of the lengths of the two edges adjacent to the reference vertex), and a size factor.

4. For each image triangle/target triangle pair, a 2D lookup table can be used to determine the orientation of the target triangle in space, if we approximate the true perspective with a paraperspective approximation [22,23]. There is one 2D lookup table per target triangle, which gives two possible orientations of this target triangle when the reference angle and edge ratio of its image are entered.

5. Comparing the size of the image triangle to the size of the target triangle of known orientation, we can then find the distance of the target triangle from the camera lens center.

6. The preliminary transformations of the image triangle can be reversed to obtain the actual 3D pose of the target triangle, the corresponding 3D position of the target center, and the image of the target center.

7. The target center projections are clustered to identify the pose of the whole target.

When RAMBO analyzes its first image, it does not have any *a priori* knowledge of which feature triangles are visible. In this case, the system uses all the possible combinations of target triangles and image triangles. However, clustering gives better results if most improper matches are removed, and it is possible to do so after a few consistent pose estimates of the target have been obtained. The system can also avoid considering matches for target triangles which are at a nearly grazing angle with the lines of sight, since for these triangles image analysis is likely to perform poorly and paraperspective does not approximate true perspective well.

For a target producing less than 16K image triangle/target triangle combinations, each pose calculation takes around one second on a CM-2 with 16K processors but without floating point processors.

6 Motion Prediction

The computation of a target position from an image gives the translation vector and rotation matrix of the target coordinate system in the camera coordinate system. However, the camera itself is set in motion by the robot arm. The trajectory of the camera in an absolute coordinate system is known, and it is straightforward to get the position of the camera coordinate system at the time the image was taken and to find the target position at this time in an absolute coordinate system.

From a sequence of target positions, the robot must be able to predict future positions of the target in order to construct plans of actions. These target positions are points in six-dimensional space (three translation parameters and three rotation angles), each with a time label. We can fit a parametric function of time, such as a polynomial, to each of these sequences of coordinates. The target trajectory is then described parametrically by six functions of time. Calculating these functions for a future value t of the time parameter will give a predicted target position at this future time.

If RAMBO is used in space and the axes of the frame of reference of the target coincide with the principal axes of inertia, then in the absence of external forces the translation of this coordinate system should be uniform, as well as the rotations around the three axes. But more complex cases could occur (for example, a structural element could be tethered at one end). The data base describing the target could specify what ranges and types of motions are possible, and this data could be used to determine the best way to parameterize the target trajectory.

7 Task and Trajectory Planning

In order to perform task and trajectory planning, RAMBO currently makes the simplifying assumption that a complex goal can be decomposed into a sequence of simple subgoals, and that each subgoal can be performed with one joint of the robot in a fixed position with respect to the target. This joint has to "tag along" with the target, thus we call this joint the *tagging joint* of the robot. The fixed position with respect to the target that the tagging joint must follow to complete a subgoal will be called a *goal point*. All goal points required for each complex action on a target can be predefined in a data base of actions specific to each target. Each goal point is defined by six coordinates, three for the location and three for the orientation of the tagging joint, in the coordinate system of the target.

Once the tagging joint is moving along the target so that it does not move with respect to the target, the more distal joints can be used to perform the finer details required by the subgoal. The programming of these distal joints will not be considered here, since it is equivalent to programming a robot to perform a task on a fixed object. For example a subgoal for a robot arm on a space shuttle might be grabbing a handle on a tumbling satellite. A database containing the geometry of the satellite would also specify in what position — fixed in the satellite frame of reference — the wrist of the robot arm should be in order for the end effector to grab the handle. Here the tagging joint is the wrist, and the goal point is a position above the handle given in the satellite frame of reference. Once the wrist is positioned at the goal point — which requires constant motion control of the robot arm during the subgoal completion, since the satellite is tumbling — the joints of the end effector require the same grabbing motion with respect to the wrist as would be needed if the satellite were not moving.

In our experimental setup, we have concentrated on reaching the goal points. Each subgoal consists of illuminating a light-sensitive diode mounted on the surface of the target for a given duration. The source of light is a laser pointer mounted on the tool plate of the robot arm. Each diode is mounted inside a tube at the focal point of a lens which closes that tube, so that the laser beam must be roughly aligned with the optical axis of the lens to trigger the electronic circuits which control the output of the diodes. Thus a goal point for the laser tool is defined by the positions at a short distance from the lens of a diode along the optical axis, and by the orientation of this axis.

8 Bringing a Tagging Joint to a Goal Point of the Target

Suppose the original trajectory of the tagging joint in location/orientation space is the vector $\vec{p}_0(t)$ (Figure 2). The goal trajectory in location/direction space is given by the vector $\vec{p}_g(t)$. At time t_0 we want the tagging joint to "launch" from its original trajectory $\vec{p}_0(t)$, and to "land" at time $t_g = t_0 + T$, on the goal trajectory $\vec{p}_g(t)$. The reaching trajectory $\vec{p}_r(t)$ should be equal to trajectory $\vec{p}_0(t)$ at time t_0 and to trajectory $\vec{p}_g(t)$ at time t_g. The operation will last for the reaching duration T. Furthermore, the first derivatives should also be equal at these times, so that the velocities change smoothly when the robot departs from its original trajectory and reaches the goal trajectory. Once a launching time t_0 and a reaching duration T are chosen, the end points of the reaching trajectory $\vec{p}_r(t)$, as well as the first derivatives of the reaching trajectory at these points are known. These boundary conditions are enough to define $\vec{p}_r(t)$ in terms of a parametric cubic spline, a curve in which all the coefficients of the six cubic polynomials of time can be calculated.

In our experiments we have also explored an alternative method which uses a scalar piecewise quadratic interpolation function $f_T(t)$ which is 0 at time t_0, 1 at time t_g, with horizontal derivatives at these times, and continuous derivatives in the time interval. This function is expressed by

$$f_T(t) = 2\left(\frac{t}{T}\right)^2, \qquad 0 \le t \le T/2$$

$$f_T(t) = -2\left(\frac{t-T}{T}\right)^2 + 1, \quad T/2 < t \le T$$

and the reaching trajectory is the interpolated trajectory given by

$$\vec{p}_r(t) = f_T(t-t_0)\vec{p}_g(t) + (1 - f_T(t-t_0))\vec{p}_0(t)$$

Note that the predicted motion of the target and the predicted goal point trajectories should be updated every time a new target pose is found for the target. After each of these updates, the reaching trajectory of a tagging joint should be recomputed based on the new goal trajectory, and the present joint trajectory, which may itself be a reaching trajectory started after a previous update.

9 Optimizing Reaching Trajectories

With either method of estimating reaching trajectories, one difficult problem is the preliminary choice of T, the duration of the reaching trajectory. Duration T should be chosen so that the resulting linear and angular velocities and accelerations are within the limits imposed by the robot design. Also, the reaching trajectory should not cross an obstacle or the target itself, and should not require the robot to take impossible configurations. Usually, time is the rarest commodity, and the shortest time T compatible with the above constraints should be chosen.

In our present simulations on a serial machine, the duration T is simply calculated as the time which would be necessary if the reaching trajectory followed a linear path, at a constant velocity chosen to be a safe fraction of

C4

the maximum linear velocity of the robot. Finally, we check whether this trajectory crosses robot limits or causes a collision with the target. If it does, the reaching trajectory is recalculated with a safe intermediary goal instead of the final goal point.

A better optimization of the reaching trajectory would require a choice of duration T which would set the velocities and accelerations along the reaching trajectory close to the limit capabilities of the robot. This can be done by calculating the reaching trajectories for a series of durations T, finding the maximum of the second derivatives along the trajectories, and identifying the trajectory with the smallest duration T which does not require positions, velocities and accelerations beyond the robot capabilities. On the Connection Machine, this operation can be done in only a few steps. We set up a 2D array of processing cells with time as the vertical dimension. Every column of the array contains a copy of the predicted goal trajectory, with the first cell containing the position of the goal at the present time in location/direction space, the next cell the position at a time increment in the future, and so on. Every column also contains a copy of the trajectory of the tagging joint, sampled with the same time increments as the goal trajectory. The difference between columns is that they use different durations T of the reaching trajectory, increasing from one column to the next.

Each cell computes a point of the reaching trajectory for the time t corresponding to its row and for duration T corresponding to its column, and then computes estimates of appropriate derivatives at its reaching trajectory point by communicating with its neighbors in the column. The maxima of the derivatives are computed for each column. The column that has the smallest duration T and for which the maxima of the positions and derivatives do not violate robot limits is the column which contains the desired reaching trajectory. The near-term future motion of the robot should be controlled based on this selected trajectory.

10 Higher Level Planning

In a complex action we have a set of tasks A, B, C, D, each of which requires a tagging joint to move smoothly to a specific goal trajectory. In some actions the order of the tasks is not specified, and we have to choose a good order in which to carry out the tasks. "Good" order here means one which minimizes the total time spent moving between goal trajectories. Notice that this is not equivalent to a travelling salesman problem, since the time required to move from performing, say, task B to task D, depends on when we move from B to D, therefore depends on the sequence of tasks which have been performed before B.

10.1 Greedy Approach

At any given time, we can compute all the reaching trajectories to all the goal trajectories of the remaining n tasks. From among these n reaching trajectories, we can choose the one with the shortest duration, and pursue the corresponding task. This could possibly be repeated in real time after each task is accomplished, but does not guarantee the best overall sequence of tasks.

However, given that our model of the anticipated motion of the target is being updated as new information arrives, this procedure may make the most sense, in that there may be no point in computing a global optimal sequence of tasks based on a model of anticipated target motion which will not remain correct in the future.

10.2 Exhaustive Search and Dynamic Programming

Assuming, however, that our model of anticipated motion is accurate enough to allow a meaningful computation of an overall optimal sequence of tasks, one (unattractive) possibility is to compute the total time required to complete all the tasks for all possible orderings of the tasks. For n tasks this will involve computing

$$\sum_{i=1}^{n} \frac{n!}{(i-1)!}$$

best reaching trajectories.

A dynamic programming method has been developed which can precompute the best overall sequence of n tasks using computation proportional to

$$\sum_{i=1}^{n} \frac{n!}{(n-i)!(i-1)!}$$

For four tasks, for example, this approach would require computing 32 reaching trajectories rather than the 64 required for the exhaustive approach.

11 Conclusions

We have described research on robots acting in dynamic environments. We discussed the use of vision to assess the motion of objects relevant to the robot's goals, and the use of particular motion prediction and planning techniques to accomplish these goals. We described a set of experiments currently under way involving a robot arm equipped with a camera and laser operating on a single moving target object equipped with light sensors. Finally, we detailed the parallel implementation of many of the tasks involved in the accomplishment of goals in dynamic environments, including parallel image processing, parallel pose estimation, and parallel planning.

12 Acknowledgements

The support of the Defense Advanced Research Projects Agency and the U.S. Army Engineer Topographic Laboratories under Contract DACA-76-88-C-0008 is gratefully acknowledged.

REFERENCES

[1] M. Asada, Y. Fukui, and S. Tsuji, "Representing a Global Map for a Mobile Robot with Relational Local Maps from Sensory Data", International Conference on Pattern Recognition, 1988, 520-524.
[2] R.A. Brooks, "Solving the Find-Path Problem by a Good Representation of Free-Space", *IEEE Trans. Systems, Man, and Cybernetics*,13, no.3, March-April 1983, 190-197.
[3] J.L. Crowley, "Navigation for an Intelligent Mobile Robot", *International Journal of Robotics Research* ,1, March 1985, 31-41.
[4] S.J. Dickinson and L.S. Davis, "An Expert Vision System for Autonomous Land Vehicle Road Following", Computer Vision and Pattern Recognition, 1988, 826-831.
[5] B. Faverjon and P. Tournassoud, " A Local-Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom", Proc. IEEE Robotics and Automation, 1987, 1152-1159.
[6] M. Kabuka and A.E. Arenas, "Position Verification of a Mobile Robot using Standard Pattern", IEEE Journal of Robotics and Automation, 3, 1987, 505-516.
[7] T. Lozano-Perez, "A Simple Motion-Planning Algorithm for General Robot Manipulators", *IEEE Journal of Robotics and Automation*, 3, June 1987, no.3, 224-238.
[8] H.P. Moravec, "Sensor Fusion in Certainty Grids for Mobile Robots", *AI Magazine*, 9 (2), Summer 1988, 23-104.
[9] K. Sugihara, "Some Location Properties for Robot Navigation using a Single Camera", *Computer Vision, Graphics and Image Processing*, 42, 1988, 112-129.
[10] C. Thorpe, M.H. Hebert, T. Kanade, and S.A. Shafer, "Vision and Navigation for the Carnegie-Mellon Navlab", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 362-373.
[11] M.A. Turk, D.G. Morgenthaler, K.D. Gremban, and M. Marra, "VITS-A Vision System for Autonomous Land Vehicle Navigation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 342-361.
[12] K. Fujimura and H. Samet, "A Hierarchical Strategy for Path Planning among Moving Obstacles", Center for Automation Research CAR-TR-237, University of Maryland, College Park, November 1986.
[13] K. Fujimura and H. Samet, "Accessibility: A New Approach to Path Planning among Moving Obstacles", Computer Vision and Pattern Recognition, 1988, 803-807.
[14] Q. Zhu, "Structural Pyramids for Representing and Locating Moving Obstacles in Visual Guidance of Navigation", Computer Vision and Pattern Recognition, 1988, 832-837.
[15] N. Kehtanmavaz and S. Li, "A Collision-Free navigation Scheme in the Presence of Moving Obstacles", Computer Vision and Pattern Recognition, 1988, 808-813.
[16] P.S. Schenker, R.L. French, D.B. Smith, "NASA telerobot testbed development and core technology demonstration", SPIE Conference on Space Station Automation (IV), vol.1006, Cambridge, MA, November 1988.
[17] D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice-Hall, 1982.
[18] A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, second edition, Academic Press, New-York, 1982.
[19] D. Harwood, M. Subbarao, H. Haklahti, H. and L.S. Davis, "A New Class of Edge Preserving Filters", *Pattern Recognition Letters*, 6, 1987, 155-162.
[20] S. Linnainmaa, D. Harwood, D. and L.S. Davis, "Pose Determination of a Three-Dimensional Object using Triangle Pairs", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 634-647.
[21] K. Kanatani, "Constraints on Length and Angle", *Computer Vision, Graphics and Image Processing*, 41, 1988, 28-42.
[22] Y. Ohta, K. Maenobu and T. Sakai, "Obtaining Surface Orientation of Plane from Texels under Perspective Projection", Proc. IJCAI, 1981, 746-751.
[23] J. Aloimonos and M. Swain, "Paraperspective Projection: Between Orthography and Perspective", Center for Automation Research CAR-TR-320, University of Maryland, College Park, May 1987.
[24] D.W. Thompson and J.L. Mundy, "Three-Dimensional Model Matching from an Unconstrained Viewpoint", Proc. IEEE Robotics and Automation, 1987, 208-220.
[25] D.W. Thompson and J.L. Mundy, "Model-Directed Object Recognition on the Connection Machine", Proc. DARPA Image Understanding Workshop, 1987, 98-106.
[26] R. Horaud, "New Methods for Matching 3-D Objects with Single Perspective Views", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9, no.3, May 1987.
[27] D. DeMenthon, S. Ziavras and L.S. Davis, "Fast Pose Determination of a Three-Dimensional Object on the Connection Machine", Center for Automation Research Technical Report, to be published.

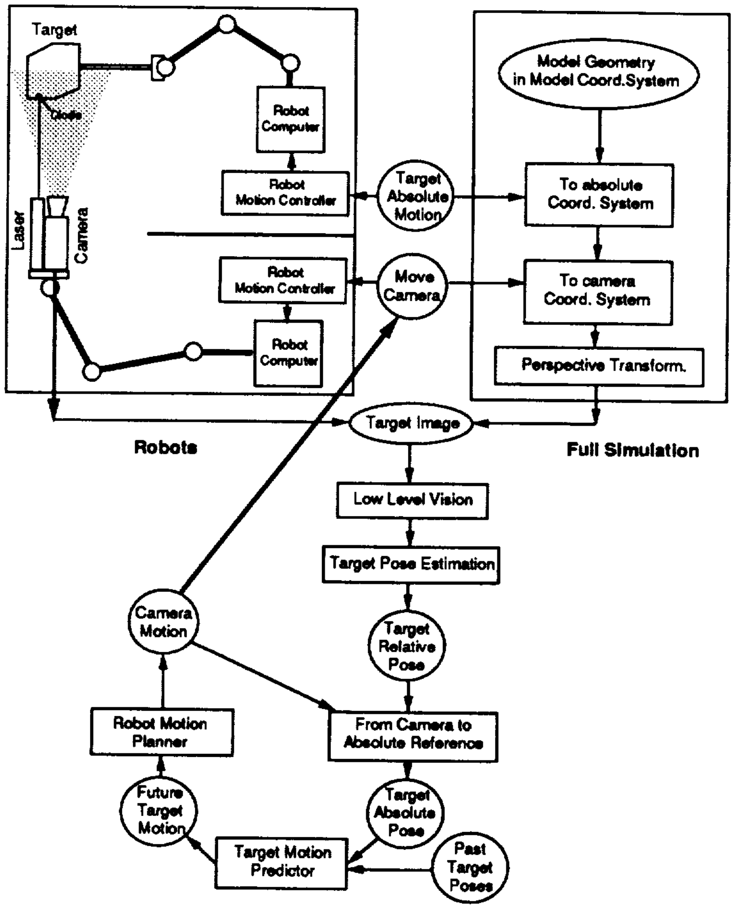


Figure 1. Vision-based control loop for a robot acting on a moving body

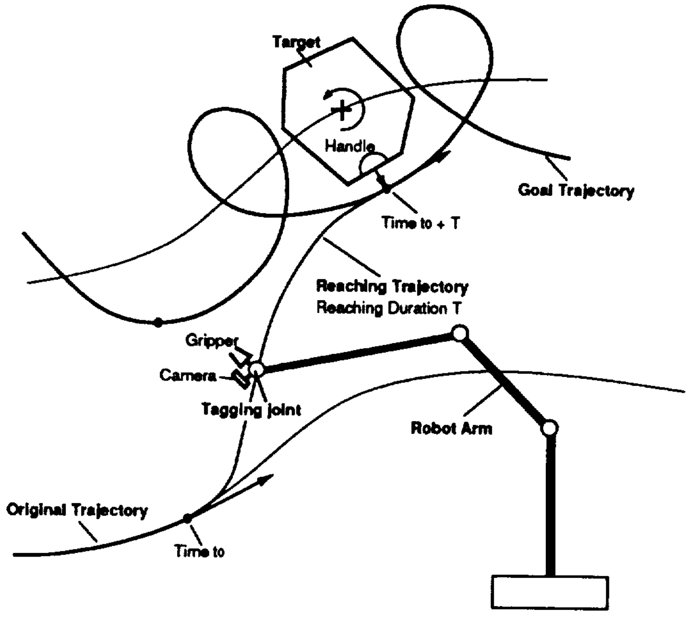


Figure 2. A tagging joint takes a reaching trajectory during time T to reach a goal trajectory required to grip a handle on the target

ORIGINAL PAGE IS
OF POOR QUALITY

N90-29023

Real-Time Edge Tracking Using a Tactile Sensor

Alan D. Berger, Richard Volpe and Pradeep K. Khosla
Department of Electrical and Computer Engineering
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

Object recognition through the use of input from multiple sensors is an important aspect of an autonomous manipulation system. In tactile object recognition, it is necessary to determine the location and orientation of object edges and surfaces. We propose a controller that utilizes a tactile sensor in the feedback loop of a manipulator to track along edges. In our control system, the data from the tactile sensor is first processed to find edges. The parameters of these edges are then used to generate a control signal to a hybrid controller. In this paper, we present theory for tactile edge detection, and an edge tracking controller. In addition, experimental verification of the edge tracking controller is presented.

1. Introduction

Object recognition is an important problem in robotics [18], particularly for autonomous manipulation systems. In the most general form, it is the problem of determining the environment from sensory data. The long-term goal of our research is to address the issue of object recognition using tactile data through the process of exploring the environment by moving the sensor. We call this approach dynamic object exploration.

Dynamic object exploration involves scheduling moves of the manipulator based on previously acquired data in order to create a more complete description of the object that is being explored. Thus, there is an interaction between manipulation and sensing. In dynamic exploration, the scheduled move affects the data obtained from the sensor, which in turn affects the next move of the manipulator. The two main steps in dynamic object exploration are: first to create strategies for scheduling manipulator moves; and second, to develop processing algorithms that will extract features of interest from the currently available data.

Researchers have actively addressed issues in both of the above mentioned components of dynamic object exploration and especially so in the context of using tactile data for exploration. Early work in edge and surface tracking was done by Bajcsy [2]. In this work, the utility of using a tactile sensor to move about an object to detect features is discussed. Work in object recognition has been done by Allen [1], Dario, et al [7], Ellis [8], Grimson [10], Klatzky, et al [12], Schneiter [19], and Stansfield [20]. Some of these groups [7, 12] take the approach of creating tactile subroutines to find particular features of an object. In this approach, a feature is extracted by calling a specific subroutine that moves and takes the appropriate measurements with the sensor. Other groups have taken a completely different approach to object recognition [8, 10, 19]. They have devised algorithms that determine the best path to approach a planar polygonal object such that it can be identified in a small number of discrete moves of the sensor.

The area of tactile image processing has received less attention than object exploration. Work has proceeded in both pattern recognition [14], and edge finding [16, 9]. Muthukrishnan, et al [16] developed a vision-like algorithm to detect edges in a tactile image. In contrast, Fearing and Binford [9] use the impulse response of their sensor to process the signals to measure the curvature of an object.

At Carnegie Mellon, our research group is addressing multi-sensor based manipulation. The goal of our research is to incorporate position, velocity, force, vision, and tactile sensors in the real-time feedback loop to create an autonomous manipulator system. The focus of this paper is to describe the use of a tactile sensor in the real-time feedback loop for edge tracking. We call this system a dynamic edge extractor. Our methodology utilizes a tactile sensor mounted on the end-effector of a manipulator to obtain data about objects. This system consists of both signal processing and control aspects. The role of the signal processing module is to find edges in the data from the tactile sensor, while the control module generates signals to servo the center of the tactile sensor along the edge. In this paper, we present the theory behind our signal processing and control modules in addition to the results of an experimental verification of the dynamic edge extractor using the CMU Direct Drive Arm II and a Lord LTS-210 Tactile Array Sensor.

2. Signal Processing

In this section, we present a brief description of the signal processing required to detect edges in a tactile image. Further details are presented in [3]. We propose algorithms that are based on the physical properties of the tactile sensor. The important charactersitics of our sensor, a Lord LTS-210, are that it has low spatial resolution and exhibits mechanical cross-talk noise. The noise is due to mechanical coupling generated by the rubber covering on the sensor. In addition, the background tactile elements (taxels) have non-zero force readings due only to mechanical cross-talk. Thus, assuming there is no cross-talk, edges are present at the locations where measured force goes from non-zero to zero. Taking these properties into account, we have devised an edge detecting algorithm that consists of two steps. The first step is an adaptive thresholder to remove cross-talk noise, and the second consists of an edge detector.

2.1. Adaptive Thresholder

The purpose of this filtering stage in our algorithm is to remove the effects of cross-talk noise from the tactile image. This operation simplifies the process of detecting edges because with no cross-talk noise, the locations where the force goes from a non-zero value to zero indicate the edges of planar surfaces. As will be discussed in the following section, the edge detector does not utilize the magnitudes of the taxels. It only uses the state of each taxel, whether it is zero or non-zero. Thus the filter may distort magnitude without adverse side effects. In the ensuing discussion of the thresholding algorithm, we show how this property is utilized.

Tactile images are very noisy. However, the noise of concern exists only at the edges of objects. In particular, the noise causes taxels that should read a force of zero to have a non-zero value. These taxels always have values that are less then their neighbors which are directly beneath the object. Hence, a thresholder that can choose the appropriate threshold at each taxel may be used to remove the noise. The threshold value is determined by the neighbors of the current taxel, thus making the thresholding an adaptive procedure. The proposed algorithm consists of three basic steps:

1. At each pixel, the force value at each of the four-connected neighbors is checked.
2. If any of these neighbors are large enough to have caused the current pixel to be noise (greater than threshold), the current pixel is set to 0 (no force).
3. Otherwise the pixel is set to a constant.

The threshold for a given taxel value is the minimum value that a neighbor must have in order for the original taxel to be cross-talk. Thus, if all neighbors of a taxel are below threshold, the taxel is considered to be part of the signal. Threshold values are determined through an experimental procedure which is described in [3]. Thresholds obtained with our sensor are summarized in Table 2-1. In this table, the first column is the cross-talk value, and the second column is the smallest value that will cause that cross-talk value.

Cross Talk	Minimum Neighbor
2	4
4	10
6	20

Table 2-1: Filter Threshold Values

2.2. Edge Detector

Edge detection in the thresholded tactile image is accomplished very efficiently. This is largely due to the assumption that the measured force goes to zero on one side of an edge, and is some non-zero value on the other side of the edge. Since the thresholding step filters out the taxels that have non-zero readings purely due to cross-talk, all that remains for the edge detector to do is to find those taxels that are neighbors of taxels with zero values.

Our edge detection algorithm consists of the following steps:

1. For each taxel, the eight-connected neighbors are checked.
2. If at least one of these neighbors is 0, the current taxel is copied to the edge image.
3. Otherwise the corresponding taxel in the edge image is set to 0.

This algorithm is very fast and minimally distorts the size, shape and position of the object. What does not come out of the algorithm is an estimate of the slope of the edges. Vision researchers have recognized that slope provides a considerable amount of information about the edge [6, 16]. However, since tactile images are small, they are simple in structure, and simply finding the position of edges appears to be sufficient for higher-level processing. In addition, standard vision edge operators that do provide this information have a number of undesirable characteristics for taction, such as edge spreading and high computational requirements. The slope of object edges may be obtained by combining the tactile and position information as the sensor tracks along the edge of an object.

3. Control

In this section, we discuss the control aspects of dynamic edge extraction [4]. The edge tracker starts on an edge and uses the extracted parameters of the edge to generate control signals to move along that edge. The control scheme is hierarchical, with the tactile controller wrapped around a cartesian space hybrid controller. In the ensuing paragraphs, we describe both the hybrid controller used in our scheme and the tactile controller.

3.1. Hybrid Controller

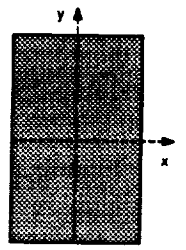


Figure 3-1: Sensor Coordinate Frame

Hybrid force and position control provides the ability to control both forces on the end effector and position of the sensor, [17]. Figure 3-1 depicts the sensor frame coordinate axis. The shaded box shows the face of the sensor. The *x* and *y* axis lie in the plane of the sensor, and the *z* axis (not shown) points out of the page. For tactile sensing, we control the normal force, and torques about the *x* and *y* axis of the sensor. Position is controlled in the *xy* plane, and about the *z* axis of the sensor. Normal force control is necessary to ensure that the tactile data is within the middle of the operating range [3]. High forces change the sensor cross-talk characteristics, and low forces result in a very low signal to noise ratio. Controlling torques about the *x* and *y* axis of the sensor allows tracking of surfaces that are not flat. Specifically, the desired torques are set to zero in order to place the sensor as flush as possible against the surface. Position control in the plane of the sensor is used because the processed sensor data provides information about the surface in the *xy* plane of the sensor. Thus, it is in this plane that we generate position control signals. Further, we control rotation about the *z* axis of the sensor. In summary, the hybrid controller commands position/orientation in three degrees of freedom, and commands force/torque in the other three. The *x* and *y* positions, and the rotation about the *z* axis of the end effector are controlled. Torques about the *x* and *y* axis, and force along the *z* axis are controlled.

3.2. Edge Tracking Controller

The edge tracking controller utilizes the edges extracted from tactile images to generate new reference signals for the hybrid arm controller. Edge tracking is initiated by positioning the tactile sensor on an edge. Through the edge detection technique discussed in the preceding section and the Modified Adaptive Hough Transform (MAHT) [5], our implementation of the Hough Transform, the tracker finds the parameters of the edge. The tracker queries a higher level process to determine which direction to travel, and begins to move the end effector in that direction. After this startup, the edge tracker functions independently of higher level input, utilizing a weighted least squares line fit to the data to determine the current parameters of the line. The Hough Transform is also performed every cycle to determine if any

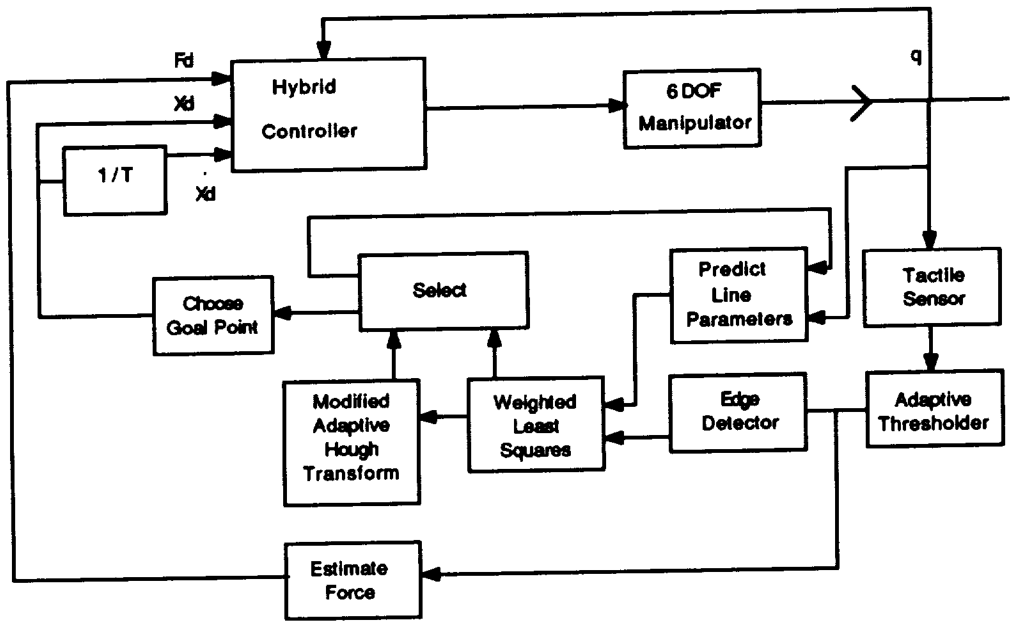


Figure 3-2: Block Diagram of Edge Tracking Controller

new edges have become visible. Each time through the loop, the robot's reference position is set to be the end point of the line segment on the sensor. Thus, if the edge extends past the end of the sensor, the point where the line intersects the edge of the sensor is selected as the goal point. As the end of a edge becomes visible to the sensor, the reference position is set to the actual end of the edge. In addition, a reference velocity is set such that the end effector should arrive at the reference position at the same time that a new reference position is generated.

Now we consider the controller in detail. Figure 3-2 is a block diagram of the edge tracker. Starting at the upper right corner of the diagram, the tactile sensor is mounted at the end effector of the manipulator. The touch image is first thresholded, with the adaptive thresholder algorithm discussed in Section 2. The thresholded image is then sent to both the edge detector and the force estimator.

The *Estimate Force* box computes a reference force such that the taxels operate in the middle of their range. Specifically, it takes the thresholded image and counts the number of taxels that are non-zero. The number of non-zero taxels multiplied by the area of each taxel is an estimate of the area of the sensor that is covered by objects. A desired normal force to the sensor may then be generated by dividing the full scale force by the area in contact with the surface. Full scale force is the total force to drive all taxels to mid-range when the entire sensor is on a flat surface.

Now, we return to the output of the adaptive thresholder. The thresholded image is passed through the edge detector (discussed in Section 2) and the result is sent to a weighted least squares line parameter estimator. This algorithm is used to estimate the slope and intercept of the edge based on the slope and intercept computed in the previous cycle. All data points in the image are weighted with a gaussian function, with $\sigma = 0.75$. A standard deviation of 0.75 was determined from our experimental work to be the best compromise for both accurate line fitting and adapting of line parameters. The weighting function is oriented such that data points located on the predicted location of the line have the highest weight. As the perpendicular distance of a point to the predicted line increases, the weight of that point decreases.

Use of this weighting function allows us to pass all of the data points to the line fitting algorithm without pre-processing to remove points that don't appear to be on the line. After the slope and intercept parameters for the edge are determined, the data points in the image corresponding to that line are removed. Also, the end points of the line are determined at this stage. These computations are the same as those performed by the MAHT, the details of which are discussed in [5]. The point removal and end point computation are part of the *Weighted Least Squares* box in the block diagram.

The weighted least squares computation requires an estimate of the parameters of the previous line segment in the current frame. The *Predict Line Parameters* box in the diagram performs this operation. The end effector will have translated and possibly rotated since the previous set of line parameters were determined. Thus the slope and intercept stored from the previous cycle must be updated to reflect this change. The predictor calculates the parameters of the current line based on the parameters of the previous line, the position of the end effector in the previous cycle, and the current position.

The remaining image is passed on to the Modified Adaptive Hough Transform. The MAHT extracts multiple lines of arbitrary slope from low signal to noise input data. Any line segments other than the one being currently tracked will be detected by this algorithm. If there are no edges remaining in the image, the transform exits, and the parameters and end points determined by weighted least squares are passed through the *Selector*. If there are new line segments, the higher level process will be informed. At this point a new line segment may be selected for tracking. When a new segment is selected, the *Selector* passes the parameters determined by MAHT to the predictor, and the end points determined by MAHT to the *Choose Goal Point* process.

Finally, *Choose Goal Point* determines which of the two end points of the segment should be set as the new reference position for the robot. The choice is made such that the robot continues to move in the same direction that it has been moving. The reference velocity is set to the distance to the new goal position divided by the edge tracking sampling period.

3.3. Discussion

The design of the edge tracking controller has several desirable properties. Specifically, it handles the of ends of segments smoothly, it can track curves in addition to straight lines, and the design is tolerant of any size sensor and data rate. In the following paragraphs, we discuss each of these points in some detail.

As the tactile sensor approaches the end of a line segment, the controller slows the arm down. When the center of the sensor reaches the end point, the arm stops. This action is a natural consequence of the way that new reference points for the hybrid controller are generated. In each cycle, the visible end of the line segment is chosen as the new reference point. Hence, before the end of the line is under the sensor, the point where the line leaves the sensor is the reference point. However, as the end point becomes visible, the controller chooses that point as the goal. This new goal point is closer to the center of the sensor than the edge of the sensor, and as a result, the velocity of the arm decreases. As the center of the sensor gets closer to the end of the segment, the arm continues to slow down, until it stops when the segment end is below the center of the sensor. This allows the arm to accurately position itself at the end of the segment, and provides an easy way to detect the end of a line segment.

Gradual curves appear as piecewise straight lines to the tactile sensor, allowing it to track them. In each cycle, new line parameters are fit to the segment of the curve that is under the sensor by the weighted least squares method. The parameters that control the weighting are the line parameters from the previous cycle. The old parameters will not be correct, as both the slope and intercept of the new section of the curve may be different. However, the old values are close enough to the correct ones that the weighting function will still be in approximately the correct location, and weighted least squares will extract the correct new parameters. Thus, the procedure of adapting the line parameters each cycle allows the system to track curves in addition to straight lines.

The sampling rate of the sensor only affects the maximum tracking velocity. As discussed above, the reference point for the hybrid controller is set to the intersection of the line with the edge of the sensor. Further, the reference velocity is set to the length of the new reference trajectory divided by the cycle time of the controller, T. As the sampling rate of the sensor decreases, T increases. Thus, desired velocities are reduced, and the reference points are placed closer together. In this scheme, there is no danger of the manipulator traveling faster than new data arrives.

4. Experimental Apparatus

In this section, we describe the hardware used in our laboratory to implement the tactile edge follower. The hardware consists of the CMU DD Arm II, control computers, a Lord Force/Torque sensor, and a Lord LTS 210 Tactile Array Sensor. The tactile control software is run on a Sun 3 computer.

4.1. Control Computers

The hardware of the DD Arm II control system consists of four integral components: the Sun workstation, the Motorola M68000 microcomputer, the Marinco processors and the TMS-320 microprocessor-based individual joint controllers. All of the computers, with the exception of the Sun are connected through a common Multibus backplane. The Eurocard Sun 3 is connected to the backplane through a serial line and interface card, operating at 4800 Baud. A simple packet based communications scheme between the M68000 Coordinating Processor and the Sun operates over this serial connection.

Previous control work included the development of the customized Newton-Euler equations for the CMU DD Arm II which achieved a computation time of 1 *ms* on the Marinco processor. The details of the customized algorithm, hardware configuration and the numerical values of the dynamics parameters are presented in [11]. For tactile sensing, we run a cartesian position controller on one of the Marinco boards, while gravity compensation torques are computed on the other Marinco. The edge tracking controller runs on the Sun. Each cycle, new reference positions are sent from the Sun to the 68000, and the current position is transmitted from the 68000 to the Sun.

4.2. Lord LTS 210 Tactile Array Sensor

To perform our taction experiments, we added a Lord LTS-210 tactile array sensor to the DD Arm II system. This sensor is mounted at the end-effector of the robot. The sensor is an array of 10×16 elements spaced on $1.8mm$ centers [13]. Each sensing site is a small plunger mounted such that as it is depressed, it blocks the light path between a LED and a photodiode [15]. Sixteen different increments in deflection may be read for each site in the sensor. A sheet of rubber protects the top surface of the sensor, but also mechanically couples the sensing sites. The sensor is interfaced to the Sun 3 through a 9600 Baud serial line.

5. Experimental Results with the CMU Direct Drive Arm II

In the ensuing paragraphs, we present the results of two different edge tracking experiments along with some observations about the use of a tactile sensor for edge tracking. First, we discuss a change in the thresholds used by the adaptive thresholder, and our strategy for orienting the tactile sensor for edge tracking. Then, we show the trajectory followed by the manipulator while tracking both straight and curved edges. The straight edge experiment allows us to view the accuracy of the tracking system, while the curved edge experiment shows the line parameter adaptation capability.

5.1. Observations

Our experiments to determine the threshold values for the adaptive thresholder show that taxel values of 2 are noise if there is a four connected neighbor of value 4 or greater [3]. During early edge tracking experiments, however, we found that after the sensor is moved over a surface for a distance of a few centimeters random 2's appear in the image. Thus, motion of the sensor against a surface makes force values of 2 unreliable. To compensate for this phenomena, the adaptive thresholder parameters were adjusted to always filter out twos regardless of the force on neighbors. No side effects in system capability are produced by the elimination of 2 as a usable force value. As discussed in Section 3, forces on the sensor are maintained above 2 for best utilization of the sensor.

We track edges with the sensor oriented such that it only contacts the edge, and not the surfaces of the object. Although the algorithms presented in the previous sections are general and may be used to track edges with the sensor in contact with the surface, we found that the friction between the object and the sensor is very high when the system is used in this mode. With our approach, two effects combine to reduce the friction. First, less area is in contact with the surface since the sensor is only contacting a line, instead of a plane. Second, a lower normal force is required. The normal force necessary to operate the sensor in the mid-region is proportional to the area of the sensor in contact with the surface. Each taxel in contact with the surface must experience a force large enough to keep it in operating range. Thus the normal force that must be exerted by the manipulator is approximately the product of the force each taxel requires and the number of active taxels. Lower forces on the sensor not only help to reduce the requirements placed on the manipulator, but also reduce wear on the sensor.

5.2. Edge Tracking

Figure 5-1 shows the result of tracking a straight edge on a metal box. In each cycle, the position of the end effector was recorded. Dots in the graph correspond to these end effector positions. Thus, the graph shows the distance between samples in addition to the robot's trajectory. The dashed line in the figure is an approximation of the location of the actual edge and is included for reference. This reference line is nearly indistinguishable from the robot's trajectory. In this experiment, the tactile sensor was oriented such that the long dimension (the 16 rows) was parallel to the direction of travel. The end effector traced a path starting at (0.47, 0.1) and ending at (0.72, 0.26), with an average speed of 5 *mm/sec*.

The plot (Figure 5-1) shows the typical characteristics of our edge tracking system. First, we note that its accuracy is acceptable and the errors are within the width of the lines in this plot. The position errors are approximately $1mm$. Remember that the tactile sensor resolution is $1.8mm$, and the reference line is only an approximation to the actual edge. Thus, we conclude that the position error is well within expectations for the system.

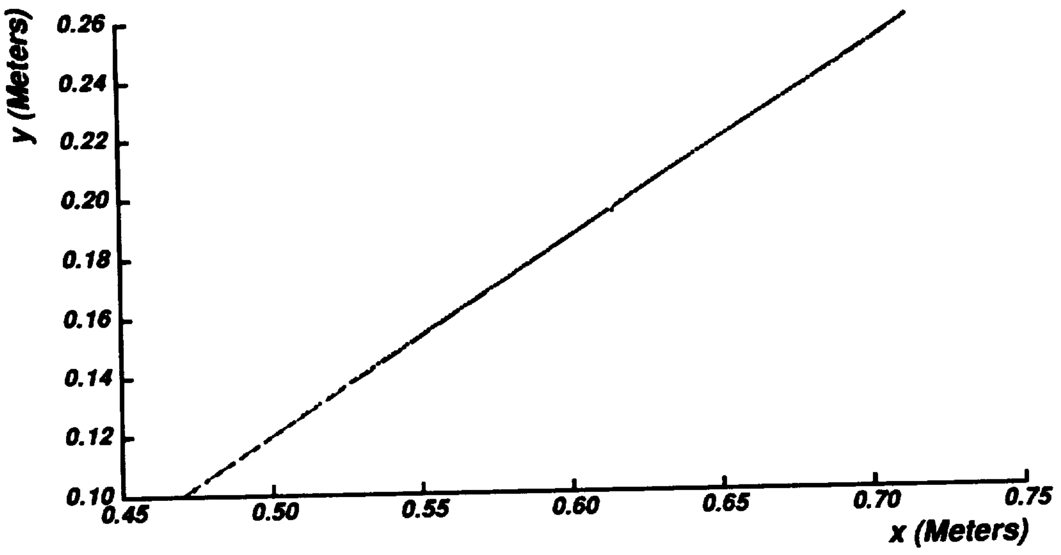


Figure 5-1: Straight Edge Tracking

Now we discuss the start and end points. At the start, (0.47, 0.1), the velocity does not appear to be as consistent as the during the remainder of the trajectory. This is to be expected as the end effector moves to place the center of the tactile array on the line, and the estimated line parameters adapt to the edge. Further, at the beginning of the line the manipulator is at rest. Thus, the first move request is a step input to the cartesian controller. Our current controller is somewhat under-damped and requires time to reach steady motion. On this particular run, the motion of the sensor smoothed out after 4 or 5 *cm*. At the very end of the trajectory, the dots become close together, indicating that the end effector slowed down. This is precisely the action designed into the system. The visible end of the line segment is always chosen as the new goal point. Thus, as the end of an edge comes into view, the commanded trajectory length, and end effector velocity decreases.

The next experiment involved tracking a *S* shaped object. Figure 5-2 shows the results when the sensor is started with the long dimension approximately oriented at a positive 45 degree angle to the x axis. Tracking follows a smooth arc beginning at (0.45, -0.14) and ending at (0.93, 0.21). The primary result from this experiment is the verification of the line parameter adaptation. The edge tracker always attempts to follow a straight line. Curves are taken to be piecewise linear, with line parameters changing slightly each cycle. The motion shown in Figure 5-2 clearly shows that line parameters are adapting properly. As with the straight line, we note a small amount of oscillation at the beginning of the trajectory, and a decrease in velocity at the end.

6. Summary

This paper presents the utilization of a tactile sensor in the feedback loop of a robot controller. There are two main components to our dynamic edge tracker: tactile signal processing and control. We base our tactile signal processing algorithms on the physical properties of the sensor. Thus, we accomplish edge detection by a two step process that first filters mechanical cross-talk noise and second finds edges by looking for transitions from non-zero to zero force. The controller uses detected line segments to generate reference signals for a manipulator. During each cycle of the edge tracker, the estimated

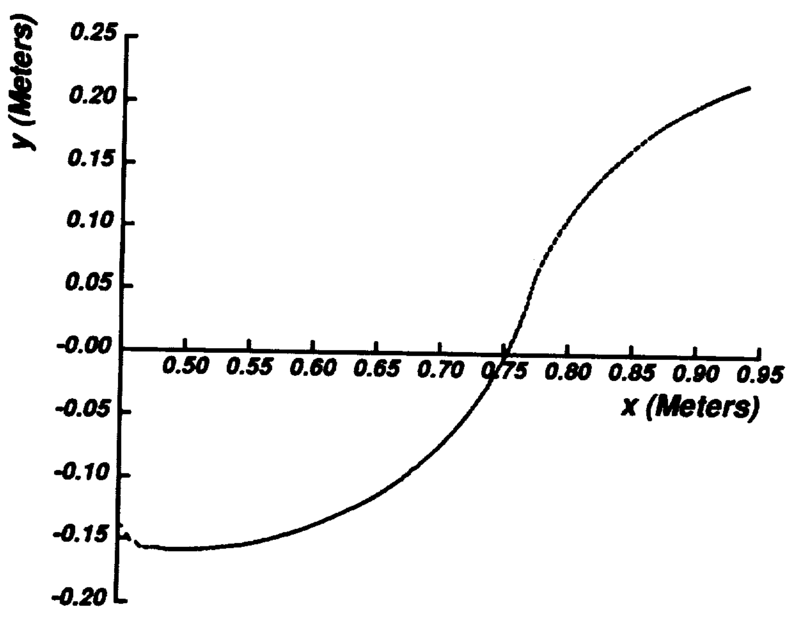


Figure 5-2: *S* Curve Tracking

parameters of the line are transformed to the current frame. These parameters are used to position a weighting function for a weighted least squares estimate of the new line. Performing this procedure every time through the control loop allows the line parameters to continuously adapt. Continuous adaptation of the parameters, in turn, allows the system to track curved objects in addition to straight objects.

References

[1] P. Allen.
Surface Descriptions from Vision and Touch.
Proc IEEE Conf on Robotics and Automation :394 - 397, 1984.

[2] R. Bajcsy.
What can we learn from one finger experiments?
In M. Brady and R. Paul (editor), *The First International Symposium on Robotics Research.* MIT Press, Cambridge, MA, 1984.

[3] A.D. Berger, and P.K. Khosla.
Edge Detection for Tactile Sensing.
Proc. SPIE's Cambridge Symposium on Optical and Optoelectronic Engineering: Advances in Intelligent Robotics Systems , November, 1988.

[4] A.D. Berger and P.K. Khosla.
Real-Time Feature Tracking Using a Tactile Sensor.
Technical Report, Departement of Electrical and Computer Engineering, Dec, 1988.

[5] A.D. Berger, and P.K. Khosla.
The Modified Adaptive Hough Transform (MAHT).
Technical Report, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, 1988.

[6] J.B. Burns, A.R. Hanson, and E.M. Riseman.
Extracting Straight Lines.
IEEE Trans. on Pattern Analysis and Machine Intelligence 8(4):425 - 455, July, 1986.

[7] P. Dario, M. Bergamasco, D. Femi, A. Fiorillo, A. Vaccarelli.
Tactile Perception in Unstructured Environments: A Case Study for Rehabilitative Robotics Applications.
Proc IEEE Conf on Robotics and Automation 3:2047 - 2054, 1987.

[8] R.E. Ellis.
Acquiring Tactile Data for the Recognition of Planar Objects.
Proc IEEE Conf on robotics and Automation 3:1799 - 1805, 1987.

[9] R.S. Fearing, and T.O. Binford.
Using a Cylindrical Tactile Sensor for Determining Curvature.
Proc IEEE Conf on Robotics and Automation :765 - 771, 1988.

[10] W.E.L. Grimson, and T. Lozano-Perez.
Model-Based Recognition and Localization from Tactile Data.
Proc IEEE Conf on Robotics and Automation :248 - 255, 1984.

[11] P.K. Khosla and T. Kanade.
Experimental Evaluation of the Feedforward Compensation and Computed-Torque Control Schemes.
In Stear, E. B. (editor), *Proceedings of the 1986 ACC.* AAAC, Seattle, WA, June 18-20, 1986.

[12] R.L. Klatzky, R. Bajcsy, and S.J. Lederman.
Object Exploration in One and Two Fingered Robots.
Proc IEEE Conf on Robotics and Automation 3:1806 - 1809, 1987.

[13] Lord LTS-210 Tactile Sensor.
Installation and Operations Manual.
Lord Corp., Cary, North Carolina, 1987.

[14] R.C. Lou, and Tsai.
Object Recognition using Tactile Image Array Sensors.
Proc IEEE Conf on Robotics and Automation :1248 - 1253, 1986.

[15] G.A. McAlpine.
Tactile Sensing.
Sensors :7 - 16, April, 1986.

[16] C. Muthukrishnan, D. Smith, D. Myers, J. Rebman, and A. Koivo.
Edge Detection In Tactile Images.
Proc. IEEE Conf. on Robotics and Automation 3:1500 - 1505, April, 1987.

[17] M.H. Raibert and J.J. Craig.
Hybrid Position/Force Control of Manipulators.
J. Dynamic Systems, Measurement, Control, 1981.

[18] A.A.G. Requicha.
Representations for Rigid Solids: Theory, Methods, and Systems.
ACM Computing Surveys 12(4):437 - 464, December, 1980.

[19] J.L. Schneiter.
An Objective Tactile Sensing Strategy for Object Recognition and Localization.
Proc. IEEE Int'l Conf. on Robotics and Automation :1262-1267, April, 1986.

[20] S.A. Stansfield.
Primitives, Features, and Exploratory Procedures: Building a Robot Tactile Perception System.
Proc IEEE Conf on Robotics and Automation 2:1274 - 1279, 1986.

N90-29024

PLANNING 3-D COLLISION-FREE PATHS USING SPHERES

Susan Bonner and Robert B. Kelley

Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

Abstract – A new scheme for the representation of objects, the Successive Spherical Approximation (SSA), facilitates the rapid planning of collision-free paths in a 3D, dynamic environment. The hierarchical nature of the SSA allows collision-free paths to be determined efficiently while still providing for the exact representation of dynamic objects. The concept of a freespace cell is introduced to allow human 3D conceptual knowledge to be used in facilitating satisfying choices for paths. Collisions can be detected at a rate better than 1 second per environment object per path. This speed enables the path planning process to apply a hierarchy of rules to create of a heuristically satisfying collision-free path.

I. INTRODUCTION

An important part of any robot application is the determination of a collision-free path through the environment for the manipulator. Typical space related applications which might require the planning of collision-free paths in 3D environments include the moving of payloads in the shuttle equipment bay, the construction of space structures, the maintenance and repair of satellites, and the navigation of both surface and winged vehicles. In this paper, we examine the use of a new object representation scheme, successive spherical approximations, which are particularly well-suited for collision detection in the planning of collision-free paths in a cluttered environments. The representation scheme is based on a hierarchy of bounding spheres and rectangular sectors of spheres which correspond to the faces of planar convex polyhedral objects. The path planning process uses an efficient generate-and-test philosophy which exploits human conceptual 3D knowledge to propose and test heuristically satisfying collision-free paths.

A satisfying path planner is one which arrives at a reasonably direct collision-free path with a minimal number of re-determinations of the path. The tradeoff is between the directness of the path and the number of iterations allowed to improve it. In addition, a good planner must be able to accommodate changes in position and orientation of objects in the environment. There are several methods which have been suggested to find such paths, but few have addressed the quality of the path found. A brief review of such methods is given next.

A popular method of planning gross motions has been to create a configuration space (C-space), use spatial occupancy enumeration to model the free space, and determine a path using explicit spatial planning. This works well in two-dimensions and with cartesian manipulators [Lozano-Perez 81], but has been found to be very computationally expensive with articulated manipulators [Gouzenes 84] and in three-dimensions [Brooks 84]. A special form of spatial occupancy enumeration using successive equivalent subdivision of space into octants, octrees, has been used in a similar manner with less computational expense in three dimensions [Faverjon 84, Hayward 86]. The major difficulty in any of these schemes, in addition to the price of computation, is the difficulty in modifying the configuration space in a dynamic environment. Goal directed incremental methods, which use obstacle sensing at run time to find collisions, have been proposed to provide safe paths [Khatib 85, Lumelsky 86]. These schemes, because they are driven by local information, cannot guarantee path directness. In the development of approach paths for grasping strategies, hypothesize-and-test methods have been used to avoid collisions with obstacles encountered along the path toward the goal [Pickett 85]. Recently, a method has been proposed [Hasegawa and Terasaki 88] which divides a workspace into areas where orientation changes are restricted and where they are not to determine a collision-free path for the manipulator.

The major concern of this paper is to provide a method for path planning which creates heuristically satisfying paths in a dynamic 3D environment. The problems associated with dynamic environments and changing orientations using C-space techniques were judged too severe for a dynamic environment, therefore, the SSA representation is used to model both moving and stationary objects and collisions are handled rapidly in operational space. The efficiency of collision detection and the heuristic nature of operational space, allow paths to be determined and directness improved upon using hypothesize-and-test techniques without unreasonable computational expense. Rules for hypothesizing paths are aided by the concept of a freespace cell, which assigns properties to obstacles depending upon their position in the environment.

First, we provide a brief introduction to the SSA representation hierarchy and its use in collision detection, which is described in detail in [Bonner and Kelley 88]. Then, we introduce a generate-and-test path planner which determines heuristically satisfying paths. We provide a definition of heuristic satisfaction and introduce the concept of a freespace cell, which is fundamental to the efficient determination of paths. A hierarchy of rules for path determination is then discussed. Finally, issues which affect path planning in a dynamic 3D environment are considered.

II. The SSA Representation Hierarchy

The SSA representation of an object is comprised of a series of successively detailed levels ranging from a sphere enclosing the object to the faces of the object itself. The hierarchical nature of these approximation levels allows for rapid and exact collision detection between 3D objects in a robot workspace with little additional cost incurred by changes in position and orientation of the objects.

The SSA representation is comprised of a series of approximations to an object referenced to a common center. Figure 1 illustrates the SSA approximations for a 2D object. Choice of the center is arbitrary; however, it must be contained within the object, which, in this paper, is limited to a convex polyhedron. Each level is composed of two bounds, an upper bound, which entirely contains the modeled portion of the object and a lower bound, which is entirely contained within the modeled portion. Uncertainty in the position of the object is modeled by adjusting these bounds by an uncertainty measure. Orientation uncertainty is handled by adjusting the angular bounds within the hierarchy.

The *bounding spheres* approximation, which provides the least accurate model of the object, is comprised of a pair of spheres positioned at the object center. The upper bounding sphere, which contains the entire object, has radius R. and the lower bounding sphere, which is entirely contained within the object, has radius r. Position uncertainty information is added at this level by simply increasing the upper bound radius and decreasing the lower bound radius by the determined measure.

The *bounding face* approximation provides an increase in the exactness of the model by dividing the spheres into rectangular sectors determined by the position of the object faces. Each sector is defined by a pair of angle ranges which locate the face with respect to the center. The upper bound for face$_i$, is a sector containing the face of radius R_i,where R_i is the maximum distance from the center to the face. The lower bound is a sector contained within the face of radius r_i, where r_i is the minimum distance from the center to the face. Position uncertainty is handled by increasing the upper bound radius and decreasing the lower bound radius, as with bounding sphere approximations. Orientation uncertainty is provided by increasing the angular ranges of each sector by the desired measure.

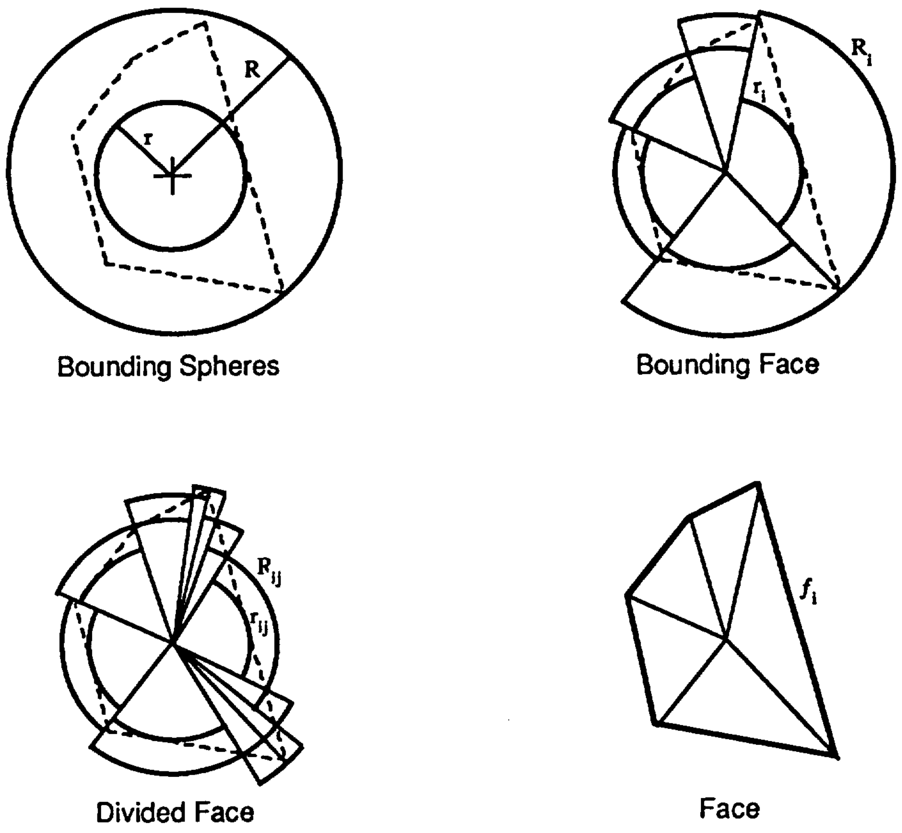


Figure 1. SSA Approximation Levels

The *divided face* approximation further divides each bounding face sector into a set of subsectors which provide an even closer approximation to the object face. Any method which provides a reasonable subdivision of the face into sectors may be used. We perform the sectoring by dividing the distance from the center to the face into even increments between r_i and R_i and assigning a subsector to each increment. The method is computationally straight forward and provides a consistent approximation to the face. Uncertainty for each divided face sector is modeled in a similar manner to a bounding face sector.

The *face* approximation is composed of the bounded planar surfaces which define the faces of the object. Upper and lower bounds are not required, as the representation of a convex polyhedron is exact at this level. Position uncertainty is handled in the definition of the faces by redefining the face vertices with reference to the object center to reflect the additional measure. Orientation uncertainty is not considered at this time.

III. Collision Detection Along a Path

The SSA representation is used to determine collisions between an object, the payload, as it moves through the environment along a straight line. The hierarchical levels of the SSA representation are used to simplify the process by determining collisions at the more approximate levels of the hierarchy or, at least, identifying those portions of the objects, if any, which require the detailed process of swept volume collision detection. Each level of the representation may result in three possible outcomes: COLLISION when the objects are found to collide, NO COLLISION when the objects do not collide, and UNKNOWN when the result is indeterminate and a more detailed approximation level must be used.

At the *bounding sphere* level, the length of the line corresponding to the closest approach of the payload center to the center of each object in the environment is used to determine the collision status. If the closest approach distance exceeds the sum of the radii of the upper bounding spheres for the payload and the object, then there is NO COLLISION for the entire path. If the distance is less than the sum of the radii of the lower bounding spheres, then there is a COLLISION with that object for that path. If the distance is between these two sums, the collision status is UNKNOWN and the next level must be consulted.

At the *bounding face* level, a cumulative sector for each object is found and compared in a similar manner to determine collisions. The first step in determining the parameters of the cumulative sector for an object is to find the swept angular range: i. e., the range of angles swept on each object as it moves relative to the other object. The upper bound radius of the cumulative sector is the maximum of the upper bound radii of all sectors which overlap the swept range. The lower bound radius of the cumulative sector is the minimum of the lower bound radii of all sectors which overlap the swept range.

The *divided face* approximation provides the next level in the hierarchy. The usefulness of this approximation for line collisions depends upon the degree of accuracy required by the path planning strategy and the computational expense required by collision detection at the face level. If collision detection to within a modeled accuracy is acceptable to the path planner, then the divided face level can be used as the final step in the collision detection process. This can result in considerable computational savings. If not, then there is a tradeoff between the additional time spent maintaining the divided face sectors and the computation saved in cases where the face level is avoided. In this paper, the path planner controls the use of the divided face approximation depending upon its requirements. Collisions at this level are determined in much the same manner as at the bounding face level. The divided face sectors which overlap the swept range are combined and compared to determine a collision.

At the *face* level, the detection of line collisions is complicated because of the difficulties involved in checking moving surfaces against each other. To guarantee a collision-free path, swept volume collision detection techniques are used. This results in considerable additional expense because each possibly colliding face of the payload must be swept along the path to create a volume and this volume examined to determine a collision.

The SSA representation is used to reduce such computational expense by eliminating faces which are not involved in a collision and, therefore, do not need to be swept. This is an inherent benefit of the hierarchical collision detection process. The SSA representation and the point collision process also provide an efficient way to model the volume swept by each face and test for collisions with obstacles.

A swept face volume (SFV) is created for each face of the payload involved in a potential collision. The endpoints of the sweep are limited to the points along the path where collisions between the upper bound radius of the cumulative sector for the obstacle and the upper bound radius of the bounding face sector of the payload could possibly occur. The SFV is formed by placing the payload at these two endpoints and creating a boundary representation of the prism formed between the two faces. This volume is then modeled using SSA techniques. The SFV is compared to the bounding face sectors of the obstacle which overlap the swept range to determine if a collision occurs. Since the SFV is stationary, the method for performing the comparison is slightly different. At the bounding sphere level, the outer and inner radii are compared to the distance between the object centers. At the bounding face level, the range of overlap between the swept object and the cumulative sector of the obstacle is found using the upper bounding radii. Sectors which overlap the base range are determined and compared to find a collision. Both the radius and the angular position of the sectors are used to find collisions in the stationary case. If the Bounding Face Level fails to determine the collision status, the planes of the faces are compared directly to determine if they overlap within the bounds of the faces. The divided face level is not used because the computational complexity involved is too great compared to the time involved in direct comparison of the faces. If any of the SFVs are found to collide with any sectors of the obstacle, then there is a COLLISION. If none collide, then there is NO COLLISION.

IV. PATH PLANNING

The generate-and-test path planning process involves an iterative technique of proposing paths between two points and testing the paths for collisions. The relative collision detection speed provided by the SSA representation allows considerable insight to be incorporated into heuristic path determination rules. The ease with which paths can be checked for collisions allow a hypothesize-and-test solution to perform several iterations to find and improve a path between two points without consuming excessive amounts of time. This section introduces the concepts required to implement a hierarchy of rules for heuristic path determination.

A. Measures of Heuristic Satisfaction

The most fundamental question which guides hypothesize-and-test path determination is "What is a heuristically satisfying path?" The obvious answer is "A path which satisfies our heuristic conception of collision-free travel between two points." To quantify notions of heuristic satisfaction, five collision-free path quality measures are defined: subpath_number, path_length, path_wander, re-orientation_length, and constrainted_length.

Subpath_number: A segmented path is a set of joined straight line segments, subpaths, which form a route from a start point to a goal point. The subpath_number is the number of subpaths in the segmented path. It is heuristically desirable to minimize the number of subpaths.

Path_length: The path_length of a segmented path is the sum of the lengths of each subpath. It is heuristically desirable to minimize the length.

Path_wander: The path_wander of a segmented path is the sum of the absolute angular changes between successive subpaths as the path is traversed. It is heuristically desirable to minimize path_wander.

Re-orientation_length: The re-orientation_length of a segmented path is the length of path over which changes in the orientation of the payload can be made. It is desirable to have as much length of path for re-orientation as possible.

Constrained_length: The constrained_length portion of a segmented path is the length of path where careful manipulation of the payload around obstacles is required. It is heuristically desirable to minimize the length of path requiring careful fitting.

A heuristically satisfying path is a collision-free path which strikes a balance in the satisfaction of the above measures. For example, a heuristically satisfying path must be collision-free and provide a minimum of subpath_number, minimal path_length and minimal path_wander, while providing sufficient re-orientation_length and minimum constrained_length path segments.

B. The Freespace Cell Concept

The freespace cell concept provides the basis for much of the decision making in the path determination procedure. The freespace cell enables path determination rules to take advantage of the inherent structure of an environment by providing a standard direction for avoiding each obstacle in the environment. The basic concept is to regard the freespace as a six-sided box or cell and to associate each obstacle in the environment with one of these six sides. Each side of the freespace cell is defined by its base plane. The interior of the cell forms the allowable space where the payload and manipulator links may travel, with the exception of the space occupied by obstacles within the cell. The freespace cell side associated with an obstacle helps to determine the path movement direction if a collision with an obstacle is detected. Motion away from the payload base plane should enable the payload to clear the obstacle and help give it a clear path through the workspace. This basic principle is utilized by many of the rules in the path determination procedure.

C. Path Determination Rules

The hypothesize-and-test method of path determination requires that paths be repeatedly proposed and tested until a collision-free path is determined. The path determination rules guide the process by providing an efficient decision-making hierarchy for altering paths in the event of a detected collision. These rules provide a means of creating paths which satisfy the criteria for heuristic satisfaction.

The rules are divided into four sets, each of which governs an increasingly tight obstacle avoidance situation. The Freespace Cell Rules are a basic exploitation of the freespace cell concept applied as a first stage in the path determination process. The Overlapping Influence Rules govern path determination when two obstacles from different freespace cell sides are interfering with the passage of the payload. The Iteration Rules are applied when the other rules fail to find a path in a tight situation. The Orientation Change Rules govern situations when it is discovered that a path cannot be found unless the orientation of the payload is changed. Each of the four sets of rules inputs a subpath with known collisions and divides it into a segmented path to be tested for collisions. A fifth set of rules, Violation Rules are applied when a path endpoint is found to be contained within an obstacle.

1) Freespace Cell Rules. The freespace cell rules use the association of obstacles with a freespace cell side to divide a colliding subpath. There are two steps in this process.

In the *grouping* step, the obstacles which collide are ordered along the subpath. Those which are adjacent and assigned to the same freespace cell side are grouped together. For each group, a cutoff plane, parallel to the base plane, is determined. The distance of the plane from the base is referred to as the height of the cutoff plane. Possible values for endpoints on the cutoff plane at each end of the group are determined and the best choice selected.

In the *combination* step, the segments are combined into a segmented path. A simple algorithm might connect the subpath start point to the first segment, connect adjacent segments, and connect the final segment to the subpath goal point. However, the information provided by the freespace cell can be used to create a more efficient segmented path. If the freespace cell sides of adjacent segments are perpendicular to each other, joining the segments end to end would result in less efficiency than joining the start point of the first segment and the final point of the second segment. The nature of the freespace cell makes this "shortcut" a reasonable alternative to a direct connection. This concept can be extended to three sides by eliminating the middle segment entirely.

An iterative process is used to alter the cutoff plane, if the initial choice results in a collision with opposing freespace cell sides. Successively "lower" heights are chosen according to the following criteria. For the first iteration, the first cutoff plane is at the height which allows clearance and free orientation change of the payload for all obstacles in the group. The second iteration assigns a separate cutoff plane to a segment endpoint and uses only the closest obstacle to that endpoint to determine clearance and orientation changes. The third iteration assigns a height based on the clearance of a single endpoint obstacle with the additional sacrifice of orientation changes to the payload. The process provides clearance of obstacles on the opposite freespace cell side and sets up the most efficient path for further processing. Note that lower heights may result in collisions with obstacles on the freespace cell side of the original segment; these are handled by reapplying freespace cell rules for the path section which collides.

2) Overlapping Influence Rules. The overlapping influence rules identify portions of a subpath which are involved in a collision with more than one obstacle from different freespace cell sides and propose a segmented path which fits the payload between the obstacles. There are three steps in the process.

The *segmentation* of a subpath is achieved by ordering the points at which the colliding obstacles collision may first influence the subpath. The obstacles are then examined for influence areas which overlap with other obstacles. Subpath portions with overlapping influence are combined into segments, the endpoints of which are determined by the part of the subpath influenced by both obstacles. Portions of the subpath influenced by no obstacles, a single obstacle or more than one obstacle from the same freespace cell side are assigned segments using freespace cell rules.

The *combination* step combines the segments into subpaths. The combination is based on the ordering of the first points of influence of the obstacles along the path. The final point of each segment is joined to the starting point of the next segment. An exception to this rule occurs when more than one overlapping influence segment is found to influence the same portion of the path. In this instance, the start point of the first segment is joined to the start point of the second segment. This provides a means of handling more than two obstacles which influence the same portion of the subpath by allowing the direction needed to navigate the first two obstacles to be established and modified by the direction needed to navigate the second two.

In the *fitting* step, the subpaths having overlapping influence are re-segmented to fit between the obstacles. The basis of the re-segmentation is the determination of a segment, perpendicular to the line between the two obstacles and passing through its midpoint. Segments parallel to the side of the obstacle closest to each subpath endpoint are chosen to connect the endpoints to the fitting segment. These segments are combined with the fitting segment create a segmented path for the subpath.

3) Iteration Rules. If a subpath proposed by the overlapping influence rules is found to contain a collision, the incremental collision detection method is applied in the form of iteration rules to find a collision-free path. The iteration rules modify the subpath by finding a collision with an obstacle and following the boundary of the obstacle until the goal point of the subpath can be reached.

The first step in using this set of rules to modify a subpath is to find the point at which the payload collides with an obstacle. This is done by placing the payload at successive intervals along the subpath and checking each point until a collision is found. The direction of the subpath and the freespace cell side of the obstacle are used to determine a direction parallel to the obstacle face involved in the collision. This direction is followed using iterative techniques until the face is cleared (in the manner of [Lumelsky, 88]). As a final step, this last point is joined to the endpoint of the subpath.

4) Orientation Change Rules. If the iteration rules cannot find a collision-free segmented path, then the orientation change rules are applied. These rules determine a new orientation for the payload which optimizes its travel parallel to an obstacle face. The orientation rules also determine areas in which the required orientation can be established.

The face of the obstacle being navigated by the payload has already been established by the iteration rules. The orientation of the payload is chosen so as to minimize its width perpendicular to the line of travel. This is done by finding a line of minimal length which passes through the payload and aligning it perpendicular to the line of travel.

Finding a place to change orientation is accomplished by finding a portion along the subpath over which an orientation change can be made. If this fails, the preceding subpaths are considered in succession until a safe place to change orientation is determined. All subpaths which are considered must be rechecked for collisions with the new orientation and modified if necessary. A similar process is used to find a place to reestablish orientation after the tight spot has been navigated.

5) Violation Rules. It is possible that a collision may occur when the payload is placed at an endpoint of a proposed subpath. This situation, which prevents the path determination rules from functioning properly, is referred to as a violation and is handled by violation rules. The rules attempt to find an alternative endpoint which does not result in a collision. Two techniques are applied in an attempt to do this, one which uses freespace cell information and a second which uses the line between the obstacle center and the endpoint.

When the first technique is applied, the endpoint is moved to a height "above" the offending obstacle in the direction indicated by the obstacle freespace cell side. Successively "lower" heights are chosen if a collision is found in a similar manner to choosing cutoff planes in *1) Freespace Cell Rules* above. If a collision still occurs, the technique is applied again using the freespace cell associated with the obstacle now involved in the collision. This technique is allowed to be applied only a limited number of times, as it is likely to cycle in a very tight situation.

If the first technique fails to find a new endpoint, a second technique is attempted. In this method, the line joining the obstacle center with the payload is determined and the endpoint is pushed out along that line until the payload no longer collides with the obstacle. If a collision occurs with another obstacle, this technique is applied with the new obstacle. This method is also only applied a limited number of times. If it fails to determine an endpoint, the environment is too cluttered for the path planner to handle.

V. PATH PLANNING ISSUES

There are several issues involved in path planning which are not addressed in the above path planning algorithm. These issues are vital to the design of a successful path planner in a structured environment. The first issue is the optimization of the generated path to improve its heuristic satisfaction. The second is the ability to provide a change in orientation of the payload as it travels over the path. The third is the ability to find paths in the presence of varying degrees of uncertainty. And the fourth is the affect of the manipulator links as it moves the payload along the planned path.

A. Optimization

Once a collision-free path has been found, a final set of rules, designed to provide additional optimization based on the path_length and path_wander measures of heuristic satisfaction, is applied. These rules attempt to shorten the total path length and make the path more direct by joining together adjacent subpaths and eliminating paths which are not in general agreement with the overall path direction. If the new subpath is collision-free, it replaces the two old ones.

B. Orientation Changes

The path planner has chosen a simple means of effecting orientation changes to the payload which allows maximum flexibility in the determination of where to make the change. The basic approach is to find two paths for the payload, one using the initial orientation and one using the final orientation and to make an reasonable choice of where to make an orientation change based on these two paths. Some simple rules govern this choice.

Orientation changes are only allowed over subpaths which are marked as capable of sustaining one. This marker is established as the path determination rules are applied. Longer subpaths are better suited to changes in orientation because they allow a longer time for the manipulator to effect the change. It is pointless to make a desired orientation change prior to a fit situation requiring a specific orientation. Therefore, the path planner eliminates such subpaths from consideration prior to these changes and gives special consideration to establishing the desired orientation along the subpath indicated for the re-establishment of the original orientation. If there is no subpath available for orientation changes, the longer subpaths are examined to determine if any portion can sustain an orientation change of the payload. If so, the subpath is divided to supply a choice for orientation changes.

First, the orientation change algorithm creates the two segmented paths and searches the first path for a subpath over which to make the orientation change, using the rules described above. It then creates a subpath, over which to make the change. Next it checks the subpath for collisions. If collisions are found, the path planner uses the basic path planning algorithm to find a collision-free path. The final result is a collision-free path which accomplishes a desired orientation change between the start and goal points.

C. Uncertainty

The nature of the SSA representation enables the path planner to generate paths using different degrees of uncertainty. This allows the system executing the path to rely on interactive sensing to optimize paths and enables path determination in tight situations. In this paper, three levels of uncertainty are established: maximum uncertainty, nominal uncertainty and no uncertainty. The maximum uncertainty guarantees a collision-free path. The nominal uncertainty should provide a collision-free path, but sensing is recommended in tight situations. No uncertainty provides a path in the ideal case and requires the use of sensing to execute. These levels of uncertainty are easily built into the SSA representation of the objects in the environment. Once paths have been established for each level, the path planner examines the relative merits of each, chooses the best, and stores those remaining for possible use in error recovery.

D. Manipulator Restrictions

The physical construction of the manipulator which moves an object through the environment puts restraints on the path the object can take. The payload is comprised of the object being moved, the gripping mechanism which holds it, and any links of the manipulator which affect orientation changes. To find collision-free paths which provide areas for orientation change, the entire payload must be modeled by a single SSA representation and used to find paths through the environment.

Once a path for the payload has been established, possible collisions of the positioning links must be examined. This is done by examining the paths dictated by the motion of the payload on each positioning link. To check the paths of the links for collisions, each link is modelled using the SSA representation and checked for collisions with the environment obstacles as it travels along its dictated path. Each manipulator link has a work envelope out of which it cannot travel. Environment obstacles which do not fall within the work envelope of a link need not be considered for collisions with that link. Since a straight line path for the payload does not map into straight line paths for the links, the incremental collision detection method is used to check for collisions at regular intervals along the path. If a collision is found, two alternatives are tried to eliminate the problem.

The first alternative is to chose an alternate kinematic configuration to find another path for the link. In an articulated arm such as the PUMA there are two sets of kinematic configurations: left_elbow and right_elbow, and elbow_up and elbow_down. The configuration used for initial path determination depends upon the position and function of the manipulator in the environment. Alternative configurations are checked according to the freespace cell side of the obstacles involved in the collision and the elbow configuration chosen. If the configuration changes fail to establish a collision-free path, changes to the path of the payload are attempted.

The second alternative attempts to alter subpaths of the payload path to eliminate link collisions. The only subpath types which are considered alterable are those formed using freespace cell rules. Subpaths which require fitting between obstacles are assumed to be too finely tuned to enable alterations substantial enough to create enough change in link position to make any difference. The freespace cell rules for altering the height of the subpath are applied in a manner which might eliminate the collision problem. If the alterations result in a collision-free path, then the links are rechecked for collisions.

VI. CONCLUSION

In this paper, we have introduced a method for the efficient determination of collision-free paths in a 3D environment. The method is based on a novel spherical representation of environment objects. The paper defines goals for creating heuristically satisfying paths and introduces the concept of a freespace cell, which exploits the structure of the environment to facilitate satisfying choices for paths. Collisions can be detected at a rate better than 1 second per environment object per path. This speed enables the path planning process to accommodate uncertainty, object re-orientation and a dynamic environment while creating a heuristically satisfying collision-free path.

REFERENCES

Bonner, Susan and Kelley, Robert, B., "A Representation Scheme for Rapid 3-D Collision Detection," Third International Symposium on Intelligent Control, Aug. 1988 (in press).

Brooks, Rodney A., "Planning Collision-free Paths for Pick and Place Operations," First International Symposium on Robotics Research, Aug./Sept. 1984, pp. 5-37.

Faverjon, Bernard, "Obstacle Avoidance using an Octree in the Configuration Space of a Manipulator," IEEE International Conference on Robotics, Atlanta, GA, Mar. 1984, pp. 504-512.

Gouzenes, Laurent, "Strategies for Solving Collision-free Trajectories Problems for Mobile and Manipulator Robots," *The International Journal of Robotics Research,* v3 n4, Winter 1984, pp. 51-65.

Hasegawa, Tsutomu, and Terasaki, Hajime, "Collision Avoidance: Divide-and Conquer Approach by Space Characterization and Intermediate Goals," *IEEE Transactions on Systems, Man and Cybernetics,* v18 n3, May/June 1988, pp. 337-347.

Hayward, Vincent, "Fast Collision Detection Scheme by Recursive Decomposition of a Manipulator Workspace," IEEE International Conference on Robotics and Automation, San Francisco, CA, Apr. 1986, pp. 1044-1049.

Khatib, Oussama, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," IEEE International Conference on Robotics and Automation, St. Louis, MO, Mar. 1985, pp. 500-505.

Lozano-Perez, Tomas, "Automatic Planning of Manipulator Transfer Movements," *IEEE Transactions on Systems, Man and Cybernetics,* v11 n10, Oct. 1981, pp. 681-698.

Lumelsky, Vladimir J., "Continuous Motion Planning in Unknown Environment for a 3D Cartesian Robot Arm," IEEE International Conference on Robotics and Automation, San Francisco, CA, Apr. 1986, pp. 1050-1055.

Pickett, E. E. and Jha, R., "Geometric Reasoning in Task Planning," Intelligent Robots and Computer Vision (SPIE v579), Cambridge, MA, Sept. 1985, pp. 121-130.

NAVIGATION

N90-29025

Map Learning with Indistinguishable Locations

Kenneth Basye and Thomas Dean[1]
Department of Computer Science
Brown University
Box 1910, Providence, RI 02912

Abstract

Nearly all spatial reasoning problems involve uncertainty of one sort or another. Uncertainty arises due to the inaccuracies of sensors used in measuring distances and angles. We refer to this as *directional* uncertainty. Uncertainty also arises in combining spatial information when one location is mistakenly identified with another. We refer to this as *recognition* uncertainty. Most problems in constructing spatial representations (*maps*) for the purpose of navigation involve both directional and recognition uncertainty. In this paper, we show that a particular class of spatial reasoning problems involving the construction of representations of large-scale space can be solved efficiently even in the presence of directional and recognition uncertainty. We pay particular attention to the problems that arise due to recognition uncertainty. The results described in this paper are applicable to the construction of global maps from satellite data as well as the construction of local navigation maps from measurements made by a rover in exploring a planetary surface.

1 Introduction

A *map* is a model of large-scale space used for purposes of navigation. *Map learning* involves exploring the environment, making observations, and then using the observations to construct a map. The construction of useful maps is complicated by the fact that observations involving the position, orientation, and identification of spatially remote objects are invariably error prone. Most studies in map learning have made the simplifying assumption that previously encountered locations can be identified with certainty. In this paper, we consider what happens when you relax that assumption.

In general, local uncertainty accumulates as the product of the distance in generating global estimates. One way to avoid this sort of accumulation is to establish strategies such that a robot can discern properties of its environment with certainty. Most existing map learning schemes exploit this sort of certainty in one way or another. The rehearsal strategies of Kuipers [Kuipers and Byun, 1988] are one example of how a robot might plan to eliminate uncertainty. In situations in which it is not possible to eliminate local uncertainty completely, it is still possible to reduce the effects of accumulated errors to acceptable levels by performing repeated experiments. To support this claim, we describe a map-learning technique based on Valiant's ***probably approximately correct*** learning model [Valiant, 1984] that, given small $\delta > 0$, constructs a map to answer global queries such that the answer provided in response to any given query is correct with probability $1 - \delta$.

2 Spatial Modeling

We model the world, for the purposes of studying map learning, as a graph with labels on the edges at each vertex. In practice, a graph will be induced from a set of measurements by identifying a set of distinctive locations in the world, and by noting their connectivity. For example, we might model a city by considering intersections of streets to be distinguished locations, and this will induce a grid-like graph. Kuipers [Kuipers and Byun, 1988] develops a mapping based on locations distinguished by sensed features like those found in buildings, and Levitt [Levitt *et al.*, 1987] develops a mapping based on locations in the world distinguished by the visibility of landmarks at a distance. In general, different mappings result in graphs with different characteristics, but there are some properties common to most

[1]This work was supported in part by the National Science Foundation under grant IRI-8612644 and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

mappings. For example, if the mapping is built for the purpose of navigating on a surface, the graph induced will almost certainly be planar and cyclic. In what follows, we will always assume that the graphs induced are connected, undirected, and of bounded degree; any other properties will be explicitly noted.

Following [Aleliunas *et al.*, 1979], a graph model consists of a graph, $G = (V, E)$, a set L of labels, and a labeling, $\phi : \{V \times E\} \rightarrow L$, where we may assume that L has a null element $\perp$ which is the label of any pair $(v \in V, e \in E)$ where e is not an edge from v. We will frequently use the word *direction* to refer to an edge and its associated label from a given vertex. With this notation, we can describe a path in the graph as a sequence of labels indicating the edges to be taken at each vertex.

If the graph is a regular tessellation, we may assume that the labeling of the edges at each vertex is consistent, *i.e.*, there is a global scheme for labeling the edges and the labels conform to this scheme at every vertex. For example, in a grid tessellation, it is natural to label the edges at each vertex as North, South, East, and West. In general, we do not require a labeling scheme that is globally consistent. You can think of the labels on edges emanating from a given vertex as local directions. Such local directions might correspond to the robot having a compass that is locally consistent but globally inaccurate, or local directions might correspond to locally distinctive features visible from intersections in learning the map of a city.

In the following, we identify two sources of uncertainty in map learning. First, there may be uncertainty in the movement of the robot. In particular, the robot may occasionally move in an unintended direction. We refer to this as *directional* uncertainty, and we model this type of uncertainty by introducing a probabilistic movement function from $\{V \times L\} \rightarrow V$. The intuition behind this function is that for any location, one may specify a desired edge to traverse, and the function gives the location reached when the move is executed. For example, if G is a grid with the labeling given above, and we associate the vertices of G with points (i, j) in the plane, we might define a movement function as follows:

$$\psi(i,j,l) = \begin{cases} (i, j+1) & 70\% \text{ if } l \text{ is North} \\ (i+1, j) & 10\% \text{ if } l \text{ is North} \\ (i-1, j) & 10\% \text{ if } l \text{ is North} \\ (i, j-1) & 10\% \text{ if } l \text{ is North} \\ \ldots \end{cases}$$

where the "..." indicate the distribution governing movement in the other three directions. The probabilities associated with each direction sum to 1. In this paper, we will assume that movement in the intended direction takes place with probability better than chance.

A second source of uncertainty involves recognizing locations that have been seen before. The robot's sensors have some error, and this can cause error in the recognition of places previously visited; the robot might either fail to recognise some previously visited location, or it might err by mistaking some new location for one seen in the past. We refer to this type of uncertainty as *recognition* uncertainty, and model it by partitioning the set of vertices into equivalence classes. We assume that the robot is unable to distinguish between elements of a given class using only its sensors.

3 Map Learning

For our purposes, a map is a data structure that facilitates queries concerning connectivity, both local and global. Answers to queries involving global connectivity will generally rely on information concerning local connectivity, and hence we regard the fundamental unit of information to be a connection between two nearby locations (*i.e.*, an edge between two vertices in the induced undirected graph). We say that a graph has been *learned completely* if for every location we know all of its neighbors and the directions in which they lie (*i.e.*, we know every triple of the form $\langle u, l, v \rangle$ where u and v are vertices and l is the label at u of an edge in G from u to v).

We assume that the information used to construct the map will come from exploring the environment, and we identify two different procedures involved in learning maps: *exploration* and *assimilation*. Exploration involves moving about in the world gathering information, and assimilation involves using that information to construct a useful representation of space. Exploration and assimilation are generally handled in parallel, with assimilation performed incrementally as new information becomes available during exploration.

The problem that we are concerned with in this paper involves both recognition and directional uncertainty with general undirected graphs. In the following, we show that a form of Valiant's probably approximately correct learning is possible when applied to learning maps under these conditions.

At any point in time, the robot is facing in a direction defined by the label of a particular edge/vertex pair—the vertex being the location of the robot and the edge being one of the edges emanating from that vertex. We assume that the robot can turn to face in the direction of any of the edges emanating from the robot's location. Directional uncertainty arises when the robot attempts to move in the direction it is pointing. Let $\alpha > 0.5$ be the probability that the robot moves in the direction it is currently pointing. More than 50% of the time, the robot ends up at the other end of the edge defining its current direction, but some percentage of the time it ends up at the other end of some other edge emanating from its starting vertex.

To model recognition uncertainty, we assume that the vertices V are partitioned into two sets, the distinguishable vertices D and the indistinguishable vertices I. We are able to distinguish only vertices in D. We refer to the vertices in D as *landmarks* and to the graph as a *landmark graph*. We define the *landmark distribution parameter*, r, to be the maximum distance from any vertex in I to its nearest landmark (if $r = 0$, then I is empty and all vertices are landmarks). We say that a procedure learns the *local connectivity within radius* r of some $v \in D$ if it can provide the shortest path between v and any other vertex in D within a radius r of v. We say that a procedure learns the *global connectivity of a graph* G *within a constant factor* if, for any two vertices u and v in D, it can provide a path between u and v whose length is within a constant factor of the length of the shortest path between u and v in G.

In the following, we assume that the probability of the robot guessing that it did traverse a path p given that it actually did traverse p is γ, that $\gamma > \frac{1}{2}+\epsilon$ where ϵ is positive, and that the robot knows these two facts. The answers to these guesses might be arrived at by various means. First, some monitoring of the robot's movement mechanisms could provide an indication of the quality of the traversal. Any *a priori* information about the path could be used to provide the answer, and some information regarding features seen in the previous exploration steps might be useful here as well.

We begin by showing that the multiplicative error incurred in trying to answer global path queries can be kept low if the local error can be kept low, that the transition from a local uncertainty measure to a global uncertainty measure does not increase the complexity by more than a polynomial factor, and that it is possible to build a procedure that directs exploration and map building so as to answer global path queries that are accurate and within a small constant factor of optimal with high probability.

Lemma 1 *Let G be a landmark graph with distribution parameter r, and let c be some integer > 2. Given a procedure that, for any $\delta_l > 0$, learns the local connectivity within cr of any landmark in G in time polynomial in $\frac{1}{\delta_l}$ with probability $1-\delta_l$, there is a procedure that learns the global connectivity of G with probability $1-\delta_g$ for any $\delta_g > 0$ in time polynomial in $\frac{1}{\delta_g}$ and the size of the graph. Any global path returned as a result will be at most $\frac{c}{c-2}$ times the length of the optimal path.*

Proof sketch: Let m be the length of the longest answer we might have to provide to a global query. Then the probability of correctness for any global answer obeys

$$p(\text{correct answer}) \geq (1-\delta_l)^m$$

A simple expansion gives

$$(1-\delta_l)^m = 1 - m\delta_l + E \geq 1 - m\delta_l$$

because $E \geq 0$. Thus, ensuring that every $\delta_l = \delta_g/m$ will ensure that

$$p(\text{correct answer}) \geq 1 - \delta_g$$

We use the local procedure on every distinguishable vertex in the graph and the resulting representation is sufficient to provide a path between any two distinguishable vertices. Note that we do not have to know $|V|$ in order to calculate δ_l, only the length of the longest answer expected. The proof that the resulting paths are within a constant factor of optimal appears in [?].

Lemma 2 *There exists a procedure that, for any $\delta_l > 0$, learns the local connectivity within cr of a vertex in any landmark graph with probability $1-\delta_l$ in time polynomial in $\frac{1}{\delta_l}$, $\frac{1}{2\gamma-1}$ and the size of G, and exponential in r.*

Proof sketch: The learning algorithm can be broken down into three steps: a landmark identification step in which the robot finds and identifies a set of landmarks, a candidate selection step in which the robot finds a set of candidates for paths in G connecting landmarks, and a candidate filtering step in which the robot determines which of those candidates actually correspond to paths in G. In order to prove the lemma, landmark identification has to succeed in identifying all landmarks in G with high probability, candidate selection has to find all paths (or at least all of the shortest paths) between landmarks with high probability, and candidate filtering has to determine which of the candidates correspond to paths in G with high probablity. Let $1-\delta_i$, $1-\delta_s$, and $1-\delta_f$ correspond, respectively, to the probabilities that the three steps succeed in performing their associated tasks. We will consider each of the three steps in turn.

The first step is easy. The robot identifies all the landmarks in G with probability $1-\delta_i$ by making a random walk whose length is polynomial in $\frac{1}{\delta_i}$ and the size of G. A more sophisticated exploration might be possible, but a random walk suffices for polynomial-time performance.

Having identified a set of landmarks, the robot has to try all paths of length r or less starting from each identified landmark. If d is the maximum degree of any vertex in G, then there can be as many as d^r paths of length r or less starting from any vertex in G. This requires than an exhaustive search will be exponential in r. Since we expect that r will generally be small, this "local" exponential factor should not be critical. For each landmark, the robot tries some number of paths of length r trying to connect other landmarks within a radius r. Again, a simple coin-flipping algorithm will do for our purposes. Starting from a landmark A, the robot chooses randomly some direction to follow, it records that direction, and then attempts to follow that direction. It continues in this manner until it has taken r steps. If it encounters one or more landmarks (other than A), then it records the set of directions attempted as a candidate path. The resulting candidates look like:

$$A_{out_0},\ {}_{in_1}X_{out_1}, \ldots,\ {}_{in_{k-1}}X_{out_{k-1}},\ {}_{in_k}B$$

where B is the landmark observed on a path starting from A, and the notation ${}_{in}X_{out}$ indicates that the robot *observed* itself entering a vertex of type X on the arc labeled *in* and it *observed* itself attempting to leave on the arc labeled *out*. The probability that the robot will traverse a particular path of length r on any given attempt is $\frac{1}{d^r}$. The probability that the robot will traverse the path of length r that it attempts to traverse is α^r. Since the robot records only those paths it attempts, it has to make enough attempts so that with high probability it records all the paths. The probability that the robot will record any given r-length path on n attempts starting at A is:

$$1 - \left[1 - \left(\frac{\alpha}{d}\right)^r\right]^n$$

In order to ensure that we record all such paths with probability $1 - \delta_s$ we have to ensure that:

$$1 - \delta_s \le \left[1 - \left[1 - \left(\frac{\alpha}{d}\right)^r\right]^n\right]^{d^r}$$

Solving for n we see that the robot will have to make a number of attempts polynomial in $\frac{1}{\delta_s}$ and exponential in r.

Candidate filtering now proceeds as follows for each candidate path. The robot attempts to traverse the path, and, if it succeeds, it guesses whether or not it did so correctly. A traversal of the path that was correct indicates that the path really is in G. With directional uncertainty, it is possible that although the traversal started and ended at the right locations and seemed to take the right direction at each step, the path actually traversed is not the one that was attempted. This results in a "false positive" observation for the path in question. The purpose of the guess after a traversal is to distinguish false positives from correct traversals. For each traversal that succeeds, we record the answer to the guess, and we keep track of the number of positive and negative answers. After n traversals and guesses, if the path really is in G, we expect the number of positive answers to be near $n\gamma$. We use $n/2$ as the threshold, and include only paths with more than $n/2$ positive answers in our representation. By making n sufficiently large, we can assure that this filtering accepts all and only real paths with the desired probability, $1 - \delta_f$. We now consider the relation between n and δ_f.

The entire filtering step will succeed with global probability $1 - \delta_f$ if we ensure that each path is correctly filtered with some local probability, which we will call δ_{fl}. An argument similar to the one used in the proof of Lemma 1 shows that the local probability is polynomial in the global probability, δ_f, d, and the size of G, and exponential in r. We now show that the number of traversals, n, is polynomial in $\frac{1}{\delta_{fl}}$ and $\frac{1}{(2\gamma-1)}$.

As mentioned above, in n traversals we expect about $n\gamma$ positive answers if the path is really in G. We use $n/2$ as a threshold, and we wish to ensure that this includes all and only real paths. We therefore consider the probability that we will get $n/2$ or fewer positive responses even though the path is really in G. This case covers the possibility of wrongly including a path; the analysis covering wrongly excluding a path is similar. We assume that the number of positive responses will be normally distributed about a mean of $n\gamma$ if the path is real. The probability of making an error is the probability that the number of positive answers will deviate from this mean enough to fall below $n/2$. If X is the number of positive responses we get, then

$$P(error) \le P(|X - n\gamma| \le n(\gamma - \frac{1}{2})) \le \frac{\gamma(1-\gamma)}{n(\gamma - \frac{1}{2})^2}$$

Replacing $\gamma(1-\gamma)$ with $\frac{1}{4}$, we have

$$P(error) \le \frac{1}{n(2\gamma - 1)}$$

which will be less than δ_{fi} provided that

$$n \ge \frac{1}{\delta_{fi}} \frac{1}{(2\gamma - 1)^2}$$

Theorem 1 *It is possible to learn the global connectivity of any landmark graph with probability $1-\delta$ in time polynomial in $\frac{1}{\delta}$, $\frac{1}{2\gamma-1}$, and the size of G, and exponential in r.*

Theorem 1 is a simple consequence of Lemma 1 and 2. It has an immediate application to the problem of learning the global connectivity of a graph where all the vertices are landmarks. In this case, the parameter $r = 0$, and we need only explore paths of length 1 in order to establish the global connectivity of the graph.

Corollary 1 *It is possible to learn the connectivity of a graph G with only distinguishable locations with probability $1-\delta$ in time polynomial in $\frac{1}{\delta}$, $\frac{1}{1-2\alpha}$, and the size of G.*

4 Discussion

The proof in the previous section relies on the assumption that the robot knows it can identify the correct execution of a set of instructions specifying a path with probability better than γ. Knowing the value of γ enables the robot to determine how many experiments it must perform in order to construct a map that is correct with probability $1-\delta$. The intuition behind this is that, in generating each candidate path in the initial exploration phase, the robot also compiles a set of observations (*e.g.*, local features, distances traveled, and angles turned) to be used as expectations during the candidate filtering step. The expectations are used to rule out situations in which the robot fails to correctly execute the instructions in the candidate path.

We have also considered the case in which movement in the intended direction takes place with probability better than chance, and that, upon entering a vertex, the robot knows with certainty the local name of the edge upon which it entered. We call the latter ability *reverse movement certainty*. In traversing an edge the robot will not know that it has ended up at some unintended location, but it will know what direction to follow in trying to return to its previous location.

With the assumption of reverse movement certainty, we have additional information that we can bring to bear on distinguishing successes from failures. As mentioned earlier, in the initial exploration phase, the robot generates a set of candidate paths from observations, where each candidate is of the form:

$$p = A_{out_0}, {}_{in_1}X_{out_1}, \cdots, {}_{in_{k-1}}X_{out_{k-1}}, {}_{in_k}B.$$

Given reverse movement certainty, the set of directions indicated by the labels $in_k, in_{k-1}, \ldots, in_1$ are guaranteed to describe a path from B to A in G. What we have to determine is whether or not the set of directions $out_0, out_1, \ldots, out_{k-1}$ describe a path from A to B in G.

To make this determination, the robot runs a set of experiments. In each experiment, the robot tries to follow the directions indicated by $in_k, in_{k-1}, \ldots, in_1$, and it keeps track of the number of *hits*: experiments in which it observes the sequence of labels $out_{k-1}, out_{k-2}, \ldots, out_0$ on entering vertices. If p is a path, then in n experiments the expected number of hits is $\alpha^d n$. If p is not a path, then the expected number of hits is $\alpha^{d-1}(1-\alpha)n$ or less depending upon how many movement errors were made in the original traversal. It is this separation between α^d and $\alpha^{d-1}(1-\alpha)$ that we exploit in determining whether or not a candidate path is actually a path in G.

Given the notion of global connectivity defined above, no attempt is made to *completely learn* the graph (*i.e.*, to recover the structure of the entire graph). It is assumed that the indistinguishable vertices are of interest only in so far as they provide directions necessary to traverse a direct path between two landmarks. But it is easy to imagine situations where the indistinguishable vertices and the paths between them are of interest. For instance, the indistinguishable vertices might be partitioned further into equivalence classes so that one could uniquely designate a vertex by specifying its equivalence class and some radius from a particular global landmark (*e.g.*, the bookstore just across the street from the Chrysler building). In [?], we show how our approach can be applied to completely learn the graph by first completely learning local neighborhoods of each landmark.

5 Related Work

Kuipers defines the notion of "place" in terms of a set of related visual events [Kuipers, 1978]. This notion provides a basis for inducing graphs from measurements. In Kuipers' framework [Kuipers and Byun, 1988], locations are arranged in an unrestricted planar graph. There is recognition uncertainty, but there is no directional uncertainty (if a robot tries to traverse a particular hall, then it will actually traverse that hall; it may not be able to measure exactly how long the hall is, but it will not mistakenly move down the wrong hall). Kuipers goes to some length to deal with recognition uncertainty. To ensure correctness, he has to assume that there is some reference location that is distinguishable from all other locations. Since there is no directional uncertainty, any two locations can be distinguished by traversing paths to the reference location. Given a procedure that is guaranteed to uniquely identify a location if it succeeds, and succeeds with high probability, we can show that a Kuipers-style map can be reliably probably almost always usefully learned using an analysis similar to that of Section 3. In fact, we do not require that the edges emanating from each vertex be labeled, just that they are cyclically ordered.

Levitt *et al* [Levitt *et al.*, 1987] describe an approach to spatial reasoning that avoids multiplicative error by introducing local coordinate systems based on landmarks. Landmarks correspond to environmental features that can be acquired and, more importantly, reacquired in exploring the environment. Given that landmarks can be uniquely identified, one can induce a graph whose vertices correspond to regions of space defined by the landmarks visible in that region. The resulting problem involves neither recognition nor movement uncertainty. Our results bear directly on any extension of Levitt's work that involves either recognition or movement uncertainty.

References

[Aleliunas *et al.*, 1979] Romas Aleliunas, M. Karp, Richard, Richard J. Lipton, Lasslo Lovass, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of mase problems. In *Proceedings of the 20th Symposium on the Foundations of Computer Science*, pages 218–223, 1979.

[Basye *et al.*, 1989] Kenneth Basye, Thomas Dean, and Jeff Vitter. Coping with uncertainty in map learning, 1989.

[Kuipers and Byun, 1988] Benjamin J. Kuipers and Yung-Tai Byun. A robust, qualitative method for robot spatial reasoning. In *Proceedings AAAI-88*, pages 774–779. AAAI, 1988.

[Kuipers, 1978] Benjamin Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[Levitt *et al.*, 1987] Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, and Philip C. Nelson. Qualitative landmark-based path planning and following. In *Proceedings AAAI-87*, pages 689–694. AAAI, 1987.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

N90-29026

Three-dimensional Motor Schema Based Navigation

Ronald C. Arkin
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Reactive schema-based navigation is possible in space domains by extending the methods developed for ground-based navigation found within the Autonomous Robot Architecture (AuRA). Reformulation of two dimensional motor schemas for three dimensional applications is a straightforward process. The manifold advantages of schema-based control persist, including modular development, amenability to distributed processing, and responsiveness to environmental sensing. Simulation results show the feasibility of this methodology for space docking operations in a cluttered workarea.

1. Introduction

One of the most important aspects of intelligent robotic control, whether teleoperated or otherwise, is a tight coupling between sensor data and motor action. It is crucial for the successful real-time operation of a robotic system that incoming perceptions be used as rapidly as possible. This strategy typically precludes the building of dynamic world models to reason over. Reflexive navigation provides highly reactive robotic control systems at a level beneath high-level planning and reasoning.

Space applications for reactive control require reformulation of the techniques developed for ground-based navigation. On earth, mobile robots typically have three controllable degrees of freedom: two for translation and one for rotation. In the micro-gravity environments of space, six degrees of freedom are present: three of translation and three of rotation (roll, pitch, and yaw). Navigation is both simplified and complicated by this change; simplified in the sense that there are more ways too move about in the world, complicated by the increased search space for solutions and the increased complexity of control.

Our previous work in ground-based navigation, conducted within the context of the Autonomous Robot Architecture (AuRA), can be readily extended into three dimensional problem domains. This includes both aerospace and undersea environments. One of the design goals of AuRA was to ensure domain independence as much as possible. This was accomplished through the use of modular design for perceptual strategies and motor behaviors, sensor and vehicle independence, and techniques for knowledge representation that are easily generalized. We have successfully demonstrated navigation of a ground-based mobile robot in the interior of buildings [2], the outdoors of a college campus [6], and in manufacturing settings [3].

This paper illustrates how the reactive/reflexive component of the AuRA architecture can be extended into three dimensional worlds. Other researchers have addressed reactive navigation for ground-based applications. Brooks' subsumption architecture [7], Payton's reflexive behaviors [12], Kadonoff's [8] arbitration techniques are several examples of this navigational paradigm. Our work in motor-schema based navigation [1] also fits into this category. It is a straightforward extension of our behavioral methodology into this new domain.

We first review two-dimensional schema-based navigation in order to provide a firm basis for its extension into three dimensional worlds. The next section describes the modifications made to the motor schemas to produce 3D navigation. Simulations are then presented showing the ability of the robot to navigate in a cluttered world and successfully dock with a workstation. Finally, a summary, conclusions, and discussion of future work completes the paper.

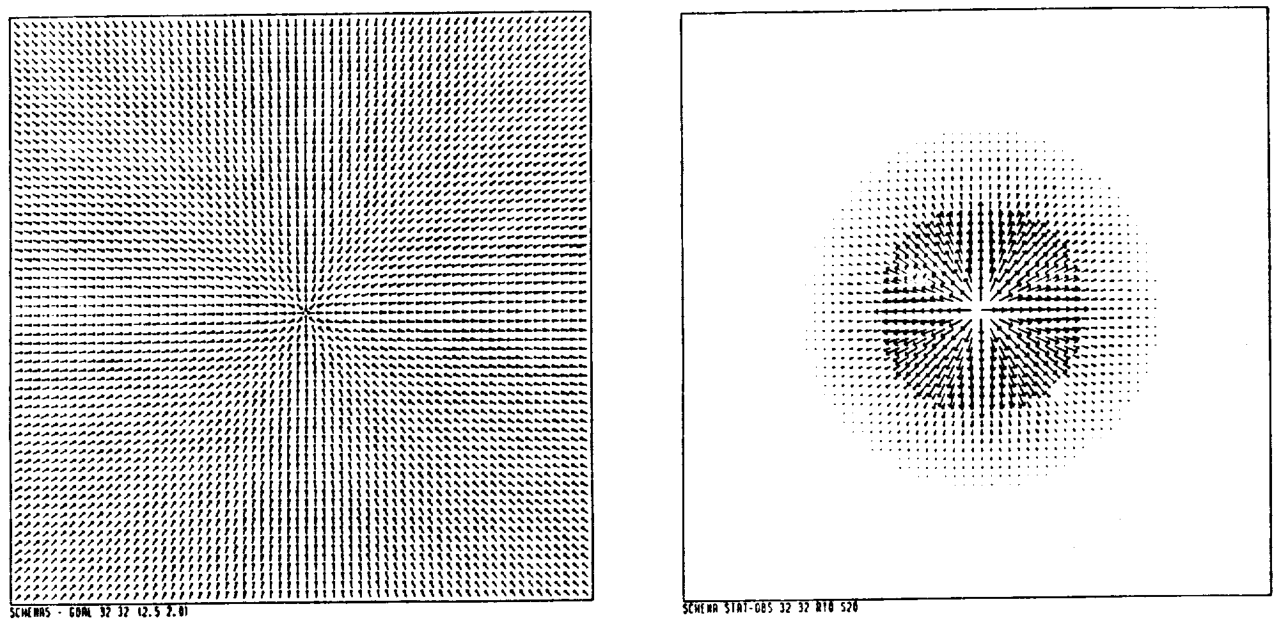
2. Review of 2D schema-based navigation

Schema-based navigation [1] involves the decomposition of motor tasks into a collection of primitive behaviors called motor schemas. Each of these schemas produces an individual velocity vector using an analog of the potential field methodology [9,10,11]. The vector output of each of these individual motor schemas is summed and transmitted to the robot. This overall vector constitutes the desired speed and direction of the robot.

Embedded within each of the motor schemas is one or more perceptual schemas that provide the necessary information for a particular robot behavior. We have used video cameras [6], shaft encoders, and ultrasonic sensors [2] as input sensor devices for the perceptual schemas. Action-oriented perception is the basis for sensor interpretation. Only the information that is required for a particular motor activity is extracted from the incoming data. This makes computational processing tractable. The use of a divide-and-conquer strategy for partitioning sensor algorithms based on motor needs, focus-of-attention mechanisms, and the application of expectation-based perception (both from *a priori* environmental knowledge and previous sensor data) facilitate rapid response. We have previously described the relationship of this methodology to psychological and neuroscientific evidence [5].

For 2D ground-based navigation we have specified several motor schemas and tested them successfully both in simulation and on our mobile robot George [2,3]. Those developed thus far include the following:

- **Move-ahead**: Move the robot in a general direction along the ground.
- **Move-to-goal**: Move the robot towards a recognized goal.
- **Avoid-static-obstacle**: Move the robot away from a detected obstacle.
- **Stay-on-path**: Keep the robot located on a hallway or road.



(a) (b)

Figure 1: Representative Two Dimensional Motor schemas
a) **Move-to-goal.**
b) **Avoid-static-obstacle.**

- **Noise:** Add some randomness to the robot's motor behavior (useful for wandering and certain pitfalls in potential field navigation).
- **Docking:** Couple ballistic and controlled motion in a manner that allows a robot to safely approach a workstation.

Two of these 2D schemas are depicted in Figure 1. Although the entire vector field in these figures is illustrated for the reader's understanding, it should be noted that the robot computes only one vector for each of its active schemas based on its location relative to a detected environmental feature. No path planning is performed. Instead the robot reacts immediately to perceived sensor data as it navigates through the world.

One must acknowledge the weaknesses of the potential fields methods along with its strengths. The possibility of local maxima, local minima, and cyclic behavior make any system that relies entirely upon this form of reflexive navigation prone to failure. We have previously reported a solution to the local maxima problem using a background noise process [1]. The other problems remain. For this reason AuRA incorporates a hierarchical planner which allows path and behavioral replanning at both the local and global levels when required. Navigational failure can be detected by observing that the velocity drops to zero while goal attainment has not been achieved (for local minima) or by exceeding a hard real-time deadline for goal completion (for cyclic behavior). In a teleoperated environment if cycle detection or deadlock due to local minima is detected the teleoperator could be informed allowing for human intervention and replanning.

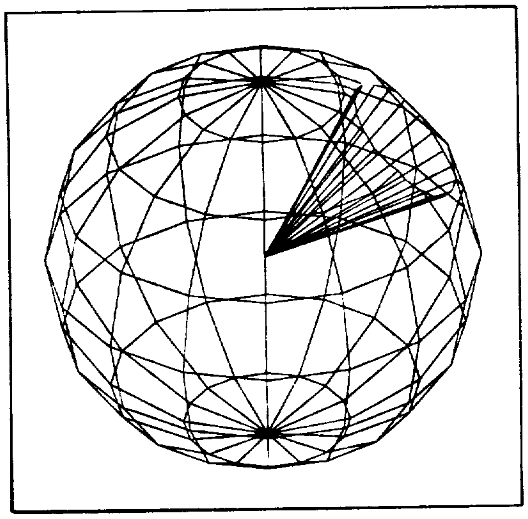
The advantages of schema-based reactive navigation are many. The ability to reflect uncertainty in perception, the simple mapping onto distributed processing systems, and the modular design facilitating incremental system growth are a few. These advantages also extend into our new work on three dimensional navigation described below.

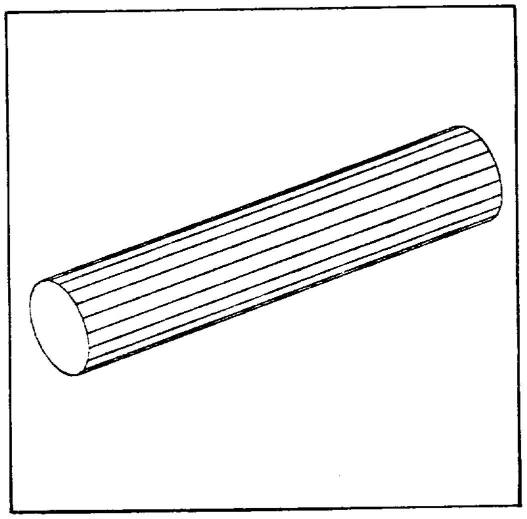
3. Three dimensional schemas

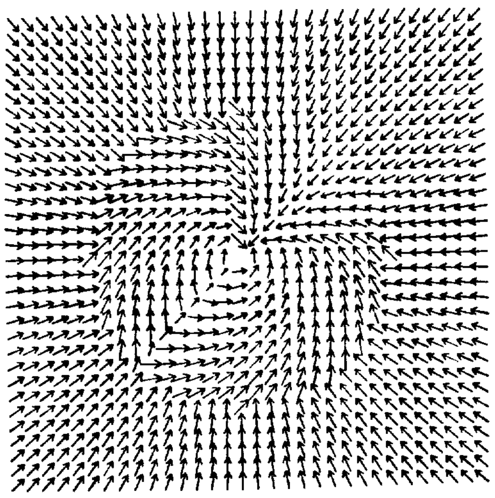
Extending 2D schema-based navigation into three dimensions is a straightforward process. All of the schemas itemized above have been reformulated from 2D cartesian space to produce vectors in three dimensional space. Although the mathematics is a bit more complex and the computations a bit more costly than for the ground-based navigation, it is still a very low cost methodology for navigation.

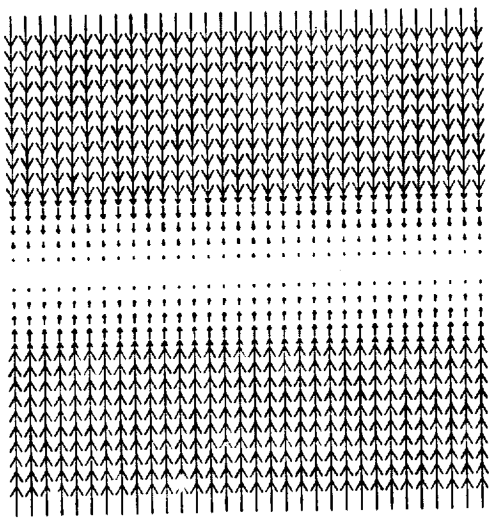
Illustrations for two of the 3D motor schemas are presented in Figures 2-3. Both perceived environmental views and cross-sectional representations of the potential fields are presented. The schemas that are not shown in figures can be readily envisioned: the **avoid-static-obstacle** schema can be viewed as a repulsive sphere instead of a repulsive disk as shown in Figure 1b; the **move-to-goal** schema has vectors pointing from all directions towards the observed goal location; the **move-ahead** schema has identical vectors located at all locations in 3D space; and the **noise** schema has random vectors scattered in 3D space instead of 2D space. Our current formulations for the 3D motor schemas are presented below.

- **Avoid-static-obstacle:**
  $V_{magnitude} =$
  $$\begin{array}{l} 0 \ for \ \ d > S \\ \frac{S-d}{S-R} * G for \ \ R < d \leq S \\ \infty \ \ for \ \ d \leq R \end{array}$$
  where:
  S = Sphere of influence (radial extent of force from the center of the obstacle)
  R = Radius of obstacle
  G = Gain
  d = Distance of robot to center of obstacle
  $V_{direction}$ = along a line from robot to center of obstacle moving away from obstacle
- **Stay-in-channel**
  $V_{magnitude} =$
  $$\begin{array}{l} \text{P for } d > (W/2) \\ \frac{d}{(W/2)} * G \text{ for } \ \ d \leq \frac{W}{2} \end{array}$$
  where:
  W = Width of channel
  P = Off path gain
  G = On path gain
  d = Distance of robot to center of channel
  $V_{direction}$ = along a line from robot to center of channel heading toward centerline


(2a)


(3a)


(2b)


(3b)

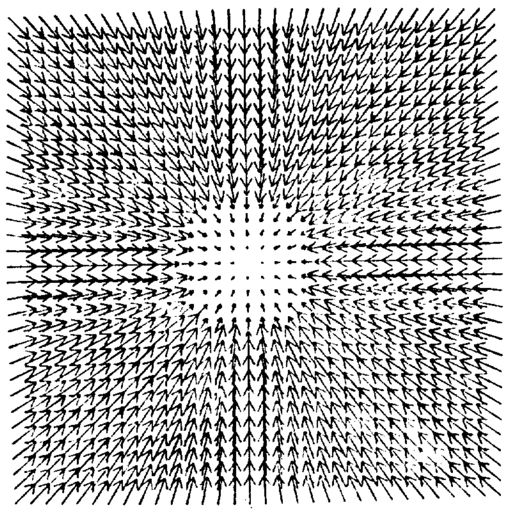

(3c)

Figure 2. Docking schema
a) Ballistic to controlled transition
 sphere and approach zone
b) Cross-section of field sliced through
 center of approach zone.
Figure 3. Stay-in-channel - 3D analog
of stay-on-path
a) Channel representation
b) Lengthwise cross-section of field.
c) Perpendicular cross-section of field.

- **Move-ahead**
  $V_{magnitude}$ = fixed gain value
  $V_{direction}$ = in specified compass direction
- **Move-to-goal**
  $V_{magnitude}$ = fixed gain value
  $V_{direction}$ = in direction towards perceived goal
- **Noise**
  $V_{magnitude}$ = fixed gain value
  $V_{direction}$ = random direction
- **Docking**
  for ballistic component: same as **move-to-goal**.
  for controlled component (inside transition zone):
    for coercive zone (outside of approach zone): sum of a linearly decreasing tangential vector dependent on correctness of orientation and a constant attractive vector to the dock.
    for approach zone: sum of a constant tangential vector and linearly decreasing attractive vector dependent on distance from the dock.

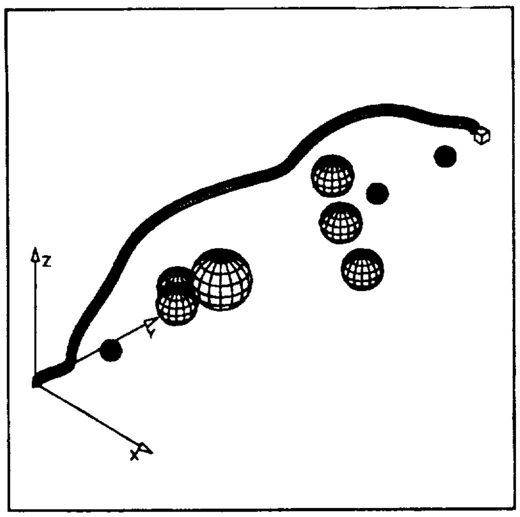
The actual control of a robot in the 3D domain is considerably more complex than the ground-based counterpart. This is a direct consequence of the increased number of degrees of freedom and the difficulty in controlling an object in free flight. Nonetheless the simulation studies presented in the next section show the success that can be attained if these engineering problems can be overcome.

4. Simulations

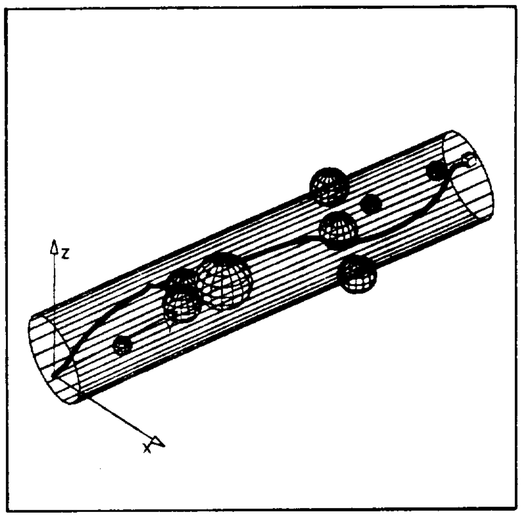
Several simulation runs are shown in Figure 4. These involve variations on a field of nine obstacles, a channel, and a goal or a dock. In each case, all of the behavioral goals are satisfied: there are no collisions with any of the obstacles, and where appropriate the robot remains within the channel and successfully migrates into the approach zone for the docking operation. Uncertainty in perception is built into this simulation run, with the robot's certainty of the presence of a particular obstacle decreasing with its distance from the obstacle. These examples clearly show that even in a highly cluttered world, reactive schema-based navigation can be successfully used to navigate a robot.

The first simulation run (Fig. 4a) shows a field containing nine obstacles. The robot starts at the origin and moves towards a goal on the other side of the obstacle field. One **Move-to-goal** schema and from zero to nine **avoid-static-obstacle** schemas are active at any one time (depending on the proximity of the root to the obstacles). The robot is pushed away from the obstacle field while moving towards its goal, completing its mission successfully.

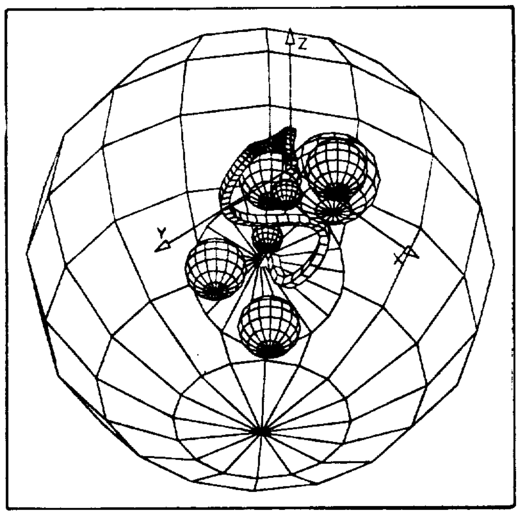
The same obstacle field and start and goal positions are present in the second simulation run (Fig. 4b). In this case, however, a **stay-in-channel** schema has been added. This forces the robot to negotiate the obstacles within the confines of the specified channel.



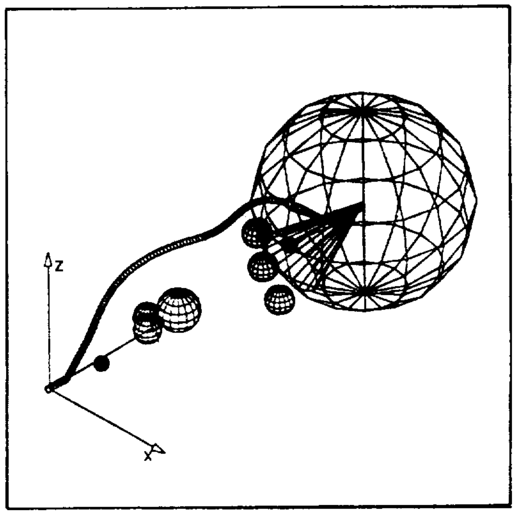
(a)



(b)

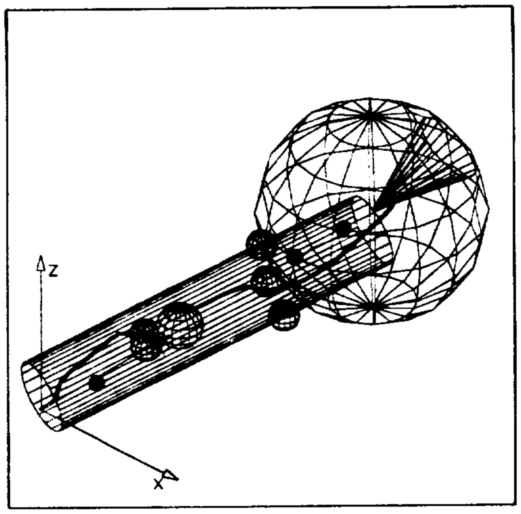


(c)



(d)

Figure 4: Simulation runs
Five different simulations of the route taken by a robot through a 3D course.
a) 9 obstacles and a **move-to-goal** schema.
b) Same as (a) with a **stay-in-channel** schema added.
c) 9 obstacles, **stay-in-channel** and **docking** schemas.
d) Same as (c) but with no **stay-in-channel** schema.
e) Same as (c) but docking approach zone is in opposite direction.



(e)

The next simulation (Fig. 4c) contains the same configuration as Figure 4b but with the goal replaced by a docking schema. The channel is not illustrated in this figure for clarity but it is present nonetheless. This altered view from the origin looking towards the dock clearly shows the robot's path as it moves past the obstacles and safely into the approach zone of the docking schema.

Figure 4d shows the same simulation environment as that of Figure 4c but without the **stay-in-channel** schema. This path should also be compared to Figure 4a (the same environment but the **move-to-goal** has been replace with the **docking** schema).

Finally, Figure 4e shows what occurs when the approach zone for the dock is on the opposite side of the channel. The robot enters into the controlled zone of the **docking** schema after successfully negotiating the obstacle course, and then is coerced to the opposite side before its final approach to the dock.

5. Summary and conclusions

We have demonstrated that schema-based navigation can be readily extended into three dimensional robot navigation domains. The advantages of this type of reactive control are many.

- Schemas are highly suitable for distributed processing.
- Their modular construction allows incremental development.
- They are responsive to environmental sensing.
- They can reflect uncertainty in perception.

We believe this work can be readily applied to both autonomous navigation and semi-autonomous teleoperation in space. By allowing the low-level obstacle avoidance and motor behaviors to be handled by reflexive sensing mechanisms, a teleoperator can be freed from the drudgery of the minute details of control and only needs to be concerned with the high-level intents of the robotic device. This approach can also cope with the large time lags in communication often found in space applications. The teleoperator can choose the behaviors that are relevant to a particular task and then let the robot strive, on its own, to satisfy the operator-specified goals. The fact that navigational snags can be detected through the use of hard real-time deadlines or the presence of unacceptably low velocities in the absence of goal attainment enables a teleoperator to be alarmed when these conditions occur. Autonomous operation, a major goal of our research, can also be developed by integrating planners that operate with a combination of *a priori* knowledge in addition to dynamically acquired world models.

Related work in progress includes the development of 3D path planning techniques based on the 2D navigational path planning strategies already in use in AURA [4]. The convex regions used in our "meadow map" for ground-based applications are being changed to convex volumes ("crystals") for path production in both undersea and aerospace applications. The A* search algorithms will be modified accordingly for this domain. We are also developing new visual strategies that are applicable to the multiple perceptual needs of the docking operator. Work on the development of a complete planning and navigation system capable of working in microgravity such as would be found in a space station environment is underway. The target robot would be capable of performing duties both in the interior and exterior of the spacecraft.

Acknowledgments

The author would like to thank Warren Gardner for the production of the simulation graphics in this paper. This research is supported in part by the Computer Integrated Manufacturing Systems Program and the Material Handling Research Center at Georgia Tech.

REFERENCES

[1] Arkin R., (1987). "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", *Proc. IEEE International Conference on Robotics and Automation*, Raleigh, N.C., pp. 264-271.

[2] Arkin, R., (1987). "Towards Cosmopolitan Robots: Intelligent Navigation of a Mobile Robot in Extended Man-made Environments", *Ph.D. Dissertation*, *COINS Technical Report 87-80*, Department of Computer and Information Science, University of Massachusetts.

[3] Arkin, R., (1988). "Intelligent Mobile Robots in the Workplace: Leaving the Guide Behind", *Proceedings of The First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp. 553-561.

[4] Arkin, R., (1989). "Navigational Path Planning for a Vision-based Mobile Robot", To appear in *Robotica*.

[5] Arkin, R., (1988). "Neuroscience in motion: The Application of Schema Theory to Mobile Robotics", chapter to appear in *Visuomotor Coordination: Amphibians, Comparisons, Models, and Robots*, eds. J.-P. Ewert and M. Arbib. New York: Plenum Press.

[6] Arkin, R., Riseman, E. and Hanson, A., (1987). "Visual Strategies for Mobile Robot Navigation", *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, Miami Beach, Florida, pp. 176-181.

[7] Brooks, R., (1986). "A Robust Layered Control System for a Mobile Robot", *IEEE Jour. of Robotics and Auto.*, Vol. RA-2, No. 1, pp. 14-23.

[8] Kadonoff, M., Benayad-Cherif, F., Franklin, A., Maddox, J., Muller, L., Sert, B. and Moravec, H., (1986). "Arbitration of Multiple Control Strategies for Mobile Robots", Mobile Robots - *SPIE proc. Vol. 727.*

[9] Khatib, O. (1985), "Real-time Obstacle Avoidance for Manipulators and Mobile Robots", *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 500-505, St. Louis.

[10] Krogh, B. (1984). "A Generalized Potential Field Approach to Obstacle Avoidance Control", *SME - RI Technical Paper* MS84-484.

[11] Krogh, B. and Thorpe, C., (1986). "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles", *IEEE Conf. on Robotics and Auto.*, pp. 1664-1669.

[12] Payton, D., (1986). "An Architecture for Reflexive Autonomous Vehicle Control", *IEEE Conf. on Robotics and Auto.*, pp. 1838-1845.

N90-29027

PERIODIC GAITS FOR THE CMU AMBLER

Swaminathan Mahalingam
Graduate Student
Department of Mechanical Engineering
Carnegie Mellon University

Dr. Suren N. Dwivedi
Professor
Department of Mechanical and Aerospace Engineering
West Virginia University

ABSTRACT

The configuration of the Carnegie Mellon University Ambler, a six legged autonomous walking vehicle for exploring Mars, enables the recovery of a trailing leg past the leading leg to reduce the energy expenditure in terrain interactions. In this paper gaits developed for this unprecedented configuration are described.

A stability criterion has been developed which ensures stability of the vehicle in the event of failure of any one of the supporting legs. Periodic gaits developed for the Ambler utilize the Ambler's unique abilities, and continuously satisfy the stability criterion.

INTRODUCTION

THE CARNEGIE MELLON UNIVERSITY (CMU) AMBLER

The CMU Ambler[1] is being developed to study the feasibility and appropriateness of legged vehicles for rugged, barren planetary surface traversal in general and for Martian terrain in particular. It is configured to overcome the general drawbacks attributed to walking machines, such as, power inefficiency, control complexity and low payload to weight ratio. A complete description of the configuration can be found in (Bares & Whittaker, 1988). Only features relevant to the current discussion are described in this paper.

Most walkers have been configured to have an identical number of legs attached to either side of an elongated body, similar to the arrangement in a mammal or a reptile. The configuration used by Ignatiev (Vukobratovic 1973) and that of ODEX I (Russell 1983) are exceptions, having six legs disposed symmetrically about a vertical axis.

The six legs of the Ambler too are configured to be symmetric about a vertical axis, but unlike the previous configurations there is a complete overlap of the leg workspaces. To provide this characteristic, the Ambler legs are mounted at different elevations on the central axis of the body, so that they can rotate fully around the axis (Figures 1(a) and (b)).

Each leg (Figure 2) of the Ambler has two revolute motions in the leveled, horizontal plane, in the manner of SCARA (Asada & Youcef-Toumi, 1987, p. 8) robot arms, and a vertical telescoping link at the end of the horizontal mechanism. After each leg is positioned over the terrain with its revolute links, the vertical telescoping motion extends the foot into contact on the terrain.

1 Ambler is an acronym for Autonomous MoBiLe, Exploration Robot.

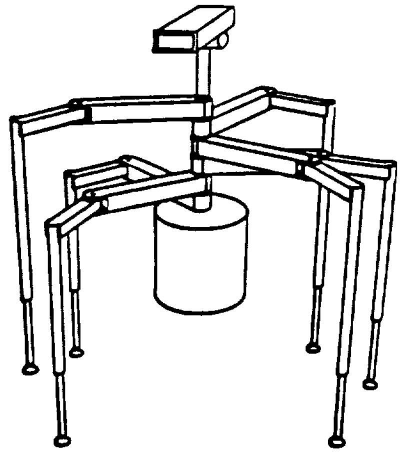


Figure 1(a). The CMU Ambler

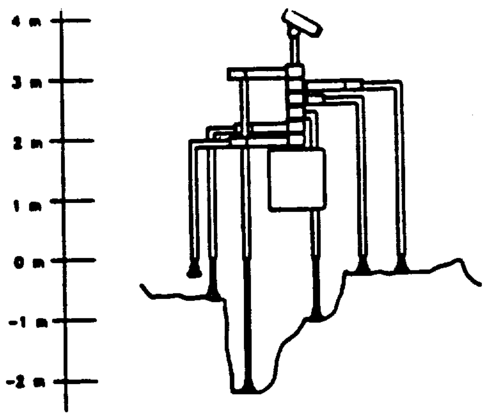


Figure 1(b). The CMU Ambler (Side View)

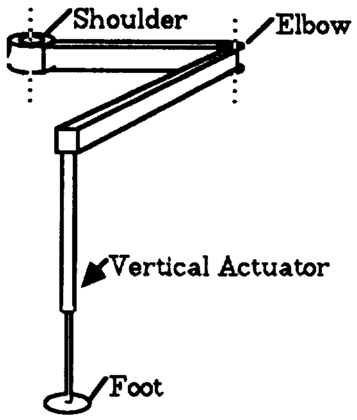


Figure 2. An Ambler Leg

The horizontal part is identical in all the legs, the vertical parts differ in height according to the leg's position in the stack — the leg which is mounted at the bottom of the stack being the shortest and the one mounted at the top being the tallest. The legs have a horizontal reach of two meters, the vertical actuators range from thirteen to seventeen feet.

THE NEED FOR A NEW STABILITY CRITERION FOR PERIODIC GAITS

The philosophy adopted for the Ambler is that autonomous natural terrain traversal warrants the utmost conservatism to be ingrained into the walk planning. It is therefore most desirable for autonomous agents to operate, as far as is practical, in a world of facts and not assumptions.

Although not explicitly stated, an underlying assumption in the development of the previous stability criteria has been that the theoretical states of the legs (either 0 or 1) considered while quantifying stability would be physically realizable and maintainable. The philosophy adopted here is that the assumption that a footing would be secure is not enough, but that *in-situ* determination of the foot-terrain interaction results is necessary. It is shown in (Mahalingam, 1988) that prediction of the outcome could prove to be extremely difficult and even if possible would be of questionable accuracy. It is therefore proposed that a new stability criterion be developed for the Ambler that admits the possibility that any leg might spontaneously fail to provide support. This could occur by reason of structural or subgrade failure.

THE NEED FOR NEW PERIODIC GAITS

The most striking feature of the Ambler is that all the legs are stacked coaxially. This opens up new possibilities in walking, the most novel ones being the opportunity to recover the trailing leg past the leading leg and to pick a leg from one side and place it on the other. As this configuration is the first of its kind, gait development for it is an unexplored territory. There is a need to generate an array of periodic gaits that utilize the Ambler's unique capabilities to the fullest.

Periodic gaits are well suited for flat terrains and free gaits for very rugged conditions. But the terrain type is relative to the size and capability of the vehicle. What may be a rugged terrain for a smaller vehicle may be a flat terrain for a large vehicle. It is viewed that most of the Martian terrain will appear mild to the Ambler. There is a need to address the efficient traversal of such terrain.

PERIODIC GAITS FOR THE AMBLER

THE STABILITY CRITERION

The Ambler is expected to move with an average speed of one meter per minute and to reach that speed from rest in thirty seconds. The mass of the Ambler is expected to be about one thousand five hundred kilograms. The inertial vector, therefore, would at worst be 0.3% of its weight, excluding the resisting frictional inertia. Static stability criteria are therefore deemed to be sufficient, and the weight vector alone suffices to establish the stability criterion.

Concept of the conservative support polygon

A support pattern is a polygon comprised of the legs in the support state. The static stability criterion requires that the weight vector pass within the support pattern. While one or more legs recover, the others are expected to maintain their states to provide continued stability. If any of the support legs changes state (structural or soil failures), the number of points in the support pattern would be reduced, and a new support pattern would become effective which might not satisfy the static stability criterion.

It is proposed that the point of intersection of the weight vector with the support polygon be kept within a sub-region of the support pattern such that stability survives the loss of any of the supporting legs. This region is termed the conservative support polygon, and is defined below for an n legged machine.

The conservative support polygon

Conservative support polygon: The conservative support polygon (CSP) of an n-legged vehicle with m legs on the ground is the intersection of the support patterns due to (m - 1) feet contacting the terrain.

The term m-1 represents that of the m legs in the support state, one is assumed to have undergone a state change and m-1 legs remain in the support state. If the number of legs assumed to have undergone a state change is n, then n cannot be greater than one, because the intersection of m!/((m-n)! * n!) polygons for n>1 does not always exist. The CSP therefore safeguards against instability in the event of losing one leg only.

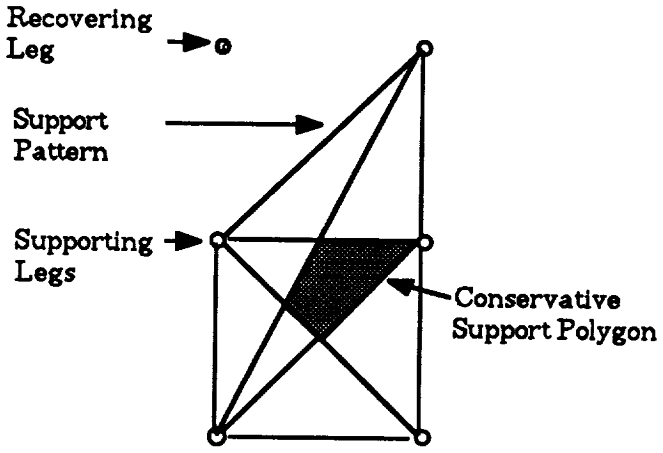


Figure 3. The Conservative Support Polygon for a six legged walking vehicle executing crawl gait is formed by the intersection of the support patterns due to four legs in the support phase.

For the existence of the CSP, the following condition must be satisfied:

A necessary but not sufficient condition for the existence of the CSP is that the number of legs in the support phase should be equal to or greater than five, i.e., $m \geq 5$.

It is a necessary condition as the intersection of m-1 legged support patterns do not exist for m less than five and hence the CSP is undefined. It is not a sufficient condition because in some configurations, the m supporting legs can generate a support pattern equivalent to four or less legs.

This condition necessitates six legged vehicles to execute crawl gaits only. For the Ambler, this complements the mission's strong belief in conservatism rather than restrict the performance of the Ambler.

Vehicles utilizing this stability criterion should ensure that the body center of gravity never lies outside the CSP. This has two implications. First, the CSP determines the allowable advancement of the vehicle before a new support pattern comes into effect. Second, gaits should provide for continuity of CSPs in consecutive support patterns.

Consecutive CSPs having point contact between them will necessitate the body center of gravity to exit one CSP and the next CSP through the contact points. The chain of consecutive CSPs therefore determines the heading of the vehicle. Consecutive CSPs having area or line contact provide more latitude for transition between CSPs than do consecutive CSPs having point contacts.

Comparison with existing stability criteria

The CSP is distinguished from other stability criteria in that the CSP does not provide a quantitative measure of stability, and therefore does not provide an optimum position to locate the body center of gravity within the CSP. The CSP establishes that as long as the weight vector intersects the support polygon within the CSP, the vehicle would be safe against instability in the event of any single leg failure. The CSP stability criterion could be expanded to include a more generalized energy stability criterion.

The CSP also supports the LSM concept. Gaits using the SM and the LSM criteria have to, depending on the terrain, dynamically determine and set appropriate limits for the stability criteria. Such techniques fail in rugged terrains where the terrain values may not attain a steady state. The CSP obviates this requirement by maintaining the body center of gravity within constant conservative LSM bounds. The LSM value is maximum when the body center of gravity is either at the lower vertex or at the upper vertex of the CSP (points a and b in Figure 6). The maximum value occurs at the mid point between the vertices. For example, in Figure 6, the CSP limits the LSM to one-fourth the distance between the leading and trailing legs.

THE AMBLER PERIODIC GAITS .

The objectives of the periodic gaits for the Ambler are to help preserve acceptable stability during locomotion, and to provide directional motion to the vehicle body with a minimum number of footfalls per distance traversed. The first objective requires that proposed periodic gaits provide conservative support polygons within consecutive support patterns having continuity to enable continuous body motion. The second criterion provides a basis of arriving at an optimal solution — a gait with optimal stability, number of footfalls per distance traversed, and directionality.

The configuration of the Ambler provides the ability to recover a trailing leg past a leading leg and the ability to pick up any leg and place it anywhere around the body Fewer footfalls per distance traversed result from the former ability, which reduces energy expenditure to the terrain as compared to conventionally configured walkers, and lessens perception and planning requirements associated with footing selection. These advantages motivate the utilization of the ability of recovering the trailing leg past the leading leg as an essential feature of the periodic gaits of the Ambler[1].

[1] It is to be noted that this ability renders the Ambler directional. The Ambler would be at a disadvantage to perform maneuvers which require the use of its omnidirectional capability when aligned for straight line motion.

Straight Line Locomotion
In initial configuration, the Ambler is assumed to have three feet on either side of the body for straight line locomotion with a periodic gait[1]. All periodic gaits therefore initiate and complete a cycle with three feet on either side of the body. For maximum advancement of the body during straight line locomotion with periodic gaits, the configuration must be directionally biased, with maximum possible spacing between consecutive ipsilateral feet, measured along the desired direction of motion. Therefore the three ipsilateral feet are assumed to be in a straight line, parallel to and equidistant from the instantaneous longitudinal axis. To ensure uniformity in advancement throughout the cycle, the three feet are configured to have equal separation between them.

The standard gait
The Standard Gait (so called because it is expected to be the default gait of the Ambler) is a crawl gait, in which a rear leg, extended to its maximum operational length[2], is recovered to the front, again extended to its maximum operational length, and the foot set on the ground. The gait assures that at this instant another leg would be ready for recovery at the rear, thereby generating a continuous motion of picking up its rear feet, one at a time, bringing it to the front, and placing it on the ground — analogous to a freestyle swimmer's hand motions.

Maximum productivity of advance argues that the foremost and the rearmost legs are to be at full extension at the same time at the initiation of each cycle, so ipsilateral feet are configured to be one-third of a cycle out of phase, and contralateral feet one-sixth of a cycle out of phase in the standard gait.

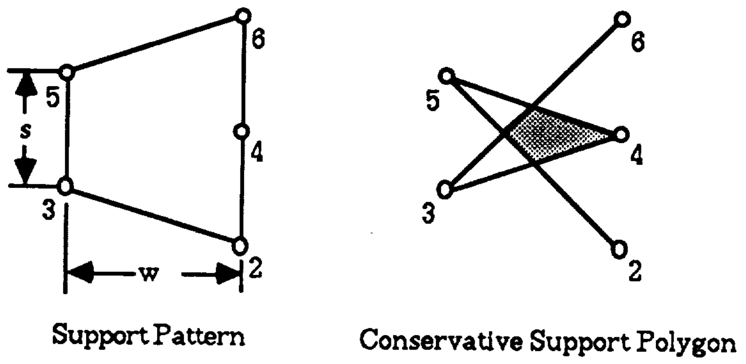


Figure 4. Configuration of the Support Pattern and the CSP at the Initiation of a Cycle of the Standard Gait. Leg 1 is in the recovery phase.

1 The body does not inherently have 'sides'. The desired direction of motion gives the 'heading 'of the body, which is defined at the initiation of each cycle. An instantaneous longitudinal axis of the vehicle can be defined by a line passing through the center of gravity of the body, parallel to the heading. Legs with feet on either side of this axis are referred to as being on the 'sides' of the body for convenience.

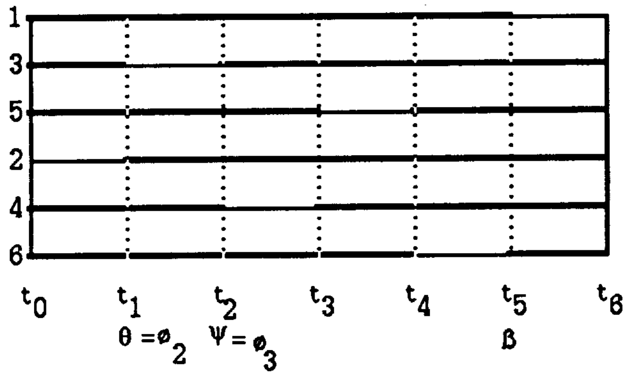
The 'orientation' of the body differs from its heading. Orientation is the rotational displacement of a body-centered co-ordinate frame with respect to an external global frame.

2 Not equal to the maximum length, which would require the two rotary links to be outstretched, introducing the possibility of control singularities.

The support pattern of the Ambler at the initiation of a cycle of the standard gait is shown in Figure 4 together with its corresponding CSP[1], illustrating the staggered placement of the feet due to the phase differences between the contralateral legs. The support patterns and the CSPs for the complete cycle are given in Figure 5.

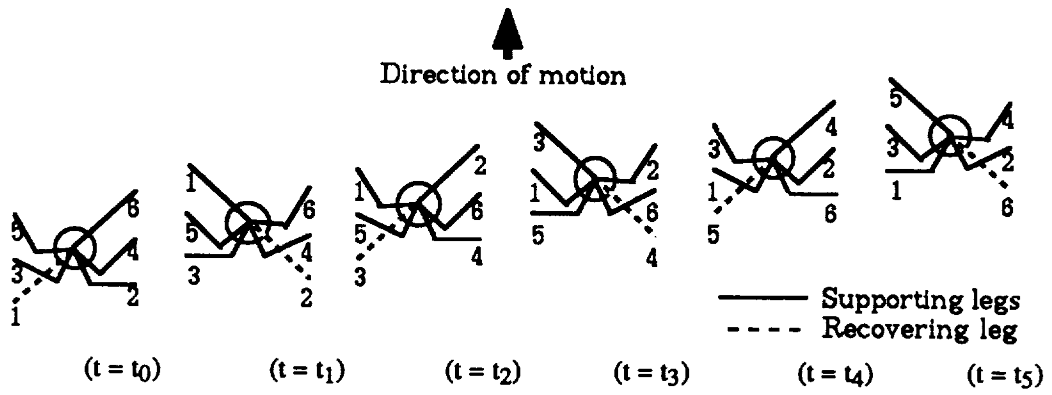
The gait diagram in Figure 5 (a) gives information about the fraction of the cycle time a leg spends supporting the body (the duty factor, β, denoted by thick black lines) and the fraction of the cycle time a leg lags behind a reference leg in contacting the ground (the leg phase, ϕ_i). The phase difference between any two adjacent ipsilateral legs (ψ) is the same in the standard gait. The cycle time is divided into six equal time divisions, t_0 - t_5 and the corresponding configurations of the Ambler are shown in Figure 5 (b). For example, at $t = t_0$, leg 1 starts recovering (denoted by a thin line in the gait diagram, and a dotted line in the configuration diagram), at $t = t_1$ leg 1 enters the support phase and leg 2 enters the recovery phase, and so forth. While the leg is recovering, the body is in continuous, uniform motion.

It is the objective of this gait to provide consecutive CSPs which are chained together at the vertices. The CSPs resulting from the standard gait for the Ambler are shown in Figure 5 (c). The center of gravity of the body lies at the rear end of the longitudinal diagonal of the CSP (point 'a' in Figure 6) at the initiation of a leg recovery, and lies at the front end of the same diagonal (point 'b' in Figure 6) at the time of foot set down. The two consecutive CSPs have a point contact between them, and are symmetric

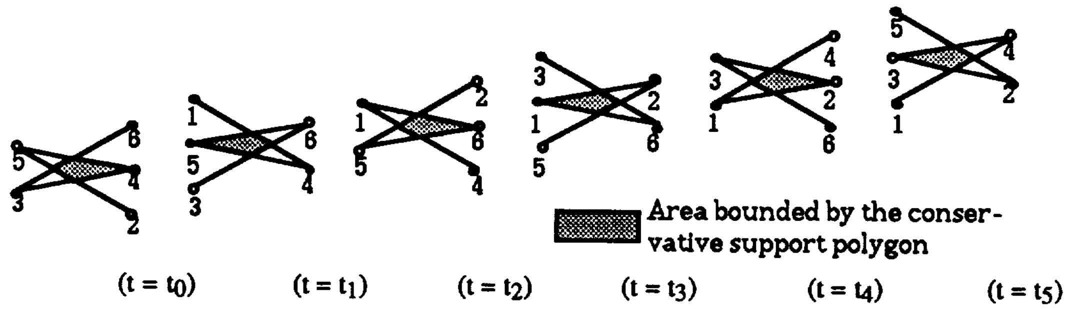


(a). Gait Diagram of the Standard Gait

1 A simple geometrical construction of the CSP for the Standard Gait is as follows: from the center foot on the side with three feet, straight lines are drawn to the two feet on the opposite side (lines 4-5 and 3-4 in Figure 3.7). From the two feet on the side with two feet, straight lines are drawn to the extreme feet on the other side, so that the two lines make an 'X' (lines 3-6 and 2-5). The four sided polygon obtained by the intersection of these lines defines the CSP.



(b). Successive Configurations of the Ambler in the Standard Gait.



(c). Successive Configurations of the CSPs in the Standard Gait.

Figure 5. The Gait Diagram, Configurations, and Conservative Support Polygons for One Cycle of the Standard Gait.

about the contact point. Therefore, the exit point of a CSP at $t = t_{i-1}$ corresponds to an entry point of a CSP at $t = t_i$, thus providing continuity between consecutive CSPs. This satisfies the first requirement of the Ambler periodic gaits, making it possible for the Ambler to have continuous motion while maintaining the body center of gravity within the CSP.

Feasibility and desirability of line and area contact between consecutive CSPs

It is worth considering whether gaits exist which provide a line or area (rather than a point) contact between consecutive CSPs, and if so, whether they would provide advantages over the standard gait.

Two consecutive support patterns differ in the location of only one foot and share four feet in common. If these four feet define a parallelogram, its diagonals will divide the support plane into four quadrants. The necessary and sufficient conditions for the existence of a point contact is that

- the four common feet define a parallelogram, and
- the fifth foot of the consecutive support polygons be on the opposite quadrants defined by the diagonals of the parallelogram.

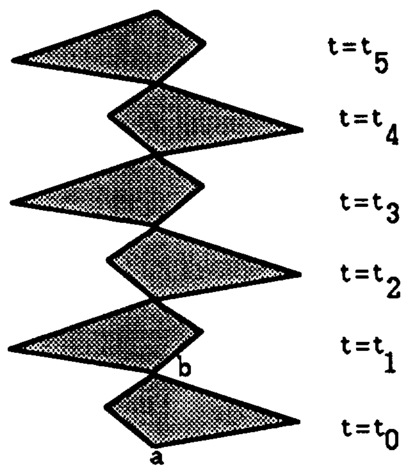


Figure 6. The Conservative Support Polygon of the Ambler with leg in recovery.

Although the generation of CSPs without point-contact is feasible, they result in gaits which have reduced average advancement per footfall. Maximum reach of the vehicle is achieved when the front and the rear legs are extended to the maximum. The maximum distance between two extreme feet in any of the gaits, which also defines the longitudinal spacing for each support pattern, is limited to twice the maximum reach of the legs. The larger the ratio between the maximum distance and the longitudinal separation, the smaller will be the advancement per footfall. Attempts to generate point-contact-free CSPs result in gaits which have larger ratios as compared to the standard gait, and therefore require a larger number of footfalls to traverse the same distance. The present view is that point contact between CSPs is not a sufficient deterrent to compromise advancement, given that the CSPs provide a high degree of guaranteed stability.

Curvilinear Locomotion

Periodic gaits which maintain the same leg sequencing as that of the standard gait, but which have different ipsilateral leg phase difference on either side, can be devised for executing motion along paths of constant curvature. Transition between two paths of different curvatures may also be accomplished by periodic gaits. Depending on the curvatures of the paths and the number of transitions from a path of one curvature to another, however, it may not be kinematically feasible to maintain periodicity continuously.

Periodic Gaits with Terrain Adaptability

The above discussed periodic gaits can be extended to have terrain adaptability features. The challenge in generating terrain adaptive gaits is to maintain β and ϕ at their gait defined constant values while providing the freedom to select a footing from within a specified area. The parameter which lends itself to variation without affecting β and ϕ, and which is significant for defining a footing selection area is the leg recovery velocity.

By varying the velocity of the legs and keeping the time of flight constant, it is possible to recover the leg a variable distance, while maintaining the β and T values constant. Two factors limit the possible range of recovery: the control schema imposes an upper limit on the leg recovery velocity which defines a reachable region from the current location of the foot, and the kinematic constraints imposed on the leg due to the current configuration. Determination of this area for each foot recovery, and placement of the foot at the optimal site would result in terrain adaptation without changing the operational gait. Maintainability of the current gait for the next footfall, can therefore be assured.

SUMMARY

Terrain adaptive gaits for the CMU Ambler were presented in this paper. The conservative support polygon (CSP) was proposed as a stability criterion. The CSP ensures that the vehicle will remain stable in the event of a state change of any of the supporting legs. This requires that at least five feet remain in the support phase at any point in time.

A number of regular periodic crawl gaits were developed for the Ambler configuration, which provided continuity between consecutive CSPs. The periodic gait which provides the minimum number of footfalls per advancement is proposed as the standard Ambler gait. The standard gait parameters were determined.

REFERENCES

Asada, H. & Youcef-Toumi, K. (1987). Direct Drive Robots: Theory and Practice. Cambridge, MA: The MIT Press.

Bares, J.E. & Whittaker, W.L. (1988). Configuration of an Autonomous Robot for Mars Exploration. Proceedings of the SME 1988 World Conference on Robotics Research.

Mahalingam, S. (1988). Terrain Adaptive Gaits for the Ambler. MS Thesis, University of NC at Charlotte.

Messuri, D.A. & Klein, C.A. (1985). Automatic Body Regulation for Maintaining Stability of Legged Vehicle During Rough-Terrain Locomotion. IEEE Journal of Robotics and Automation, RA-1, (3), 132-141.

Russell, M. (1983, Sep/Oct). ODEX I: The First Functionoid. Robotics Age, 12-18.

Vukobratovic, M. (1973). Legged Locomotion Robots: Mathematical Models, Control Algorithms and Realizations, 5th IFAC Symp. on Automatic Control in Space, Genoa.

ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation under Grant No. ROA-8615072.

N90-29028

EXPLOITING MAP PLANS AS RESOURCES FOR ACTION

David Payton

Artificial Intelligence Center
Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, CA 90265

Abstract

When plans are used as programs for controlling the action of autonomous or teleoperated robots, their abstract representation can easily obscure a great deal of the critical knowledge that originally led to the planned course of action. In this paper, we highlight an autonomous vehicle experiment which illustrates how the information barriers created by abstraction can result in undesirable action. We then show how the same task can be performed correctly using plans as a resource for action. As a result of this simple change in outlook, we become able to solve problems requiring opportunistic reaction to unexpected changes in the environment.

I. Introduction

The teleoperation of robotic vehicles in space will require significant autonomous capabilities within the robotic vehicles. With long delays between sending commands from Earth and receiving them in space, telerobotic vehicles must be sufficiently responsive to their environment so that human operators need not be involved with every detail of the robot's motion. Ultimately, robotic vehicles must have such a high degree of autonomy that they may be capable of maneuvering through difficult terrain entirely on their own accord, forming their own plans to achieve user-specified goals.

In the endeavor to develop intelligent autonomous robotic agents capable of interacting with a dynamic environment, there has been a growing awareness that traditional planning methods may not be compatible with the demands for real-time performance. Recent efforts to re-evaluate the relationship between plans and action have led to alternative viewpoints in which plans are not primarily responsible for controlling a robot's behavior. Work by Brooks, for example, is aimed at avoiding the use of plans altogether [Br]. In this approach, intelligent action is a manifestation of many simple processes operating concurrently and coordinated through the context of a complex environment. While there is no tangible representation for plans in such a system, plans are implicitly designed into the system through the pre-established interactions between behaviors. Similarly, Agre and Chapman have shown how a system that determines its actions through the constant evaluation of its current situation can perform complex tasks that might otherwise have been thought to require planning [AC1]. Despite their emphasis on the theme that action is obtained by always knowing what to do at any instant, Brooks, Agre, and Chapman do not discard the notion that look-ahead and anticipation of future events are desirable activities. While these activities are normally associated with planning, there is a difference in how the resultant "plans" are represented and used in their systems.

Agre and Chapman, for example, draw a sharp distinction between the concept of plans as communication and the more traditional views of plans as programs [AC2]. The key difference lies in the idea that plans must be constructed as a *resource* to the autonomous agent, not as an explicit set of instructions to be followed [Su]. As a resource, plans must serve as sources of information and advice to agents that are already fairly competent at dealing with the immediate concerns of their environment. In this sense, plans are used optionally, and serve only to enhance system performance. This is a significant departure from the conventional view of plans which puts them in the role of specifying a distinct course of action to systems which are often incapable of doing anything without them.

The differences between these two perspectives on planning are clearly evidenced when information from a map must be used to help guide an autonomous vehicle that must also make extensive use of sensors for detailed maneuvering and obstacle avoidance. In a plan-driven system, map-based plans are typically constructed to describe the optimal path that must be followed in order to arrive at a specified goal location. However, since the vehicle will invariably stray from the ideal path as it avoids sensed obstacles, the plan must be expressed in an abstract form that allows for error. In contrast, when map-based plans are represented for use as resources for action, this abstraction is not necessary. Instead, it is possible to make direct use of all information within the state-space of the map. As a result, information of all possible alternatives may be retained, allowing for flexible opportunistic behavior.

Our own experience with the DARPA Autonomous Land Vehicle (ALV) has led to some valuable insights into some of these issues. In a series of experiments performed by members of the Hughes Artificial Intelligence Center in August and December of 1987, a number of successful tests of autonomous cross-county navigation were performed using a system with integrated map and sensor-based control [Da] [KPR]. Some of the difficulties encountered in these experiments have pointed out certain consequences of the inappropriate use of abstraction that can occur in plan-driven systems. In this paper, we highlight one of these experiments to illustrate how the information barriers created by abstraction can lead to undesirable action. We then show how the same task can be accomplished without abstraction using plans as a resource for action, and we discuss how this approach may be extended for more complex problems.

II. The Misuse of Abstraction

In one of the cross-country experiments performed with the ALV we witnessed a surprising example of how easily plans can be misinterpreted in a plan-driven system. In this experiment, a very simple abstraction of a map-based plan was used to provide guidance to sensor-based obstacle avoidance behaviors. As shown in Figure [map plan], the basic mission objective was for the vehicle to get from one location to another while maintaining radio contact at all times. The map-based planner generated an appropriate route plan and abstracted a sequence of intermediate sub-goals to represent the critical points along this path. A portion of this sequence is illustrated in Figure [map plan] as Goals 1, 2, and 3. Note that the route had to veer specifically around one side of a rock outcrop in order to avoid loss of radio contact. To accomplish the mission, the sensor-based behaviors had primary control of the vehicle so that all obstacles could properly be avoided. The behavior decisions, however, were always biased in favor of selecting a direction toward the current map sub-goal whenever possible. As soon as the vehicle got within a specified radius of its current sub-goal, that goal would be discarded and the next sub-goal would be selected. On paper and in simulation, it seemed that this approach would be effective.

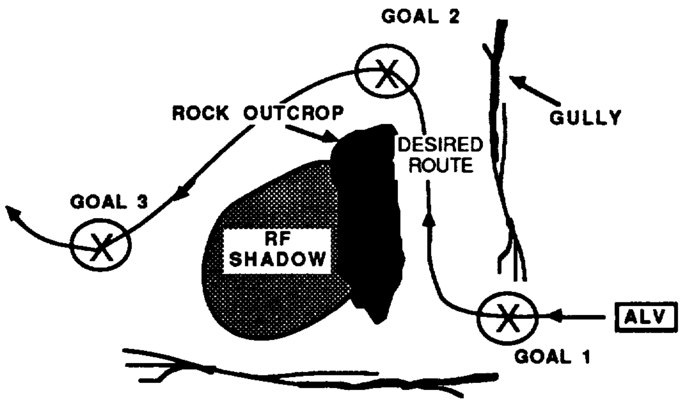


Figure [map plan]. An ALV route plan expressed as a sequence of intermediate goal points.

When we attempted to perform this mission with the ALV, the deficiencies of our method became strikingly clear. During the execution of this route, the vehicle achieved Goal 1 but then, because of local obstacles, was unable to turn appropriately to reach Goal 2. Figure [plan error] depicts the difference between the desired and actual routes. While this error is clearly apparent from the map data, the control behaviors had only the abstract route description as their guide, and this gave no indication that there was any problem with their action. Fortunately, contrary to our expectations, radio contact was not lost behind the obstacle. The mission could still be completed successfully if the vehicle were to move onward to Goal 3. Despite this new opportunity, however, the vehicle continued to persist toward Goal 2 because the abstract route description failed to give any indication that the original goal sequence was no longer suitable.

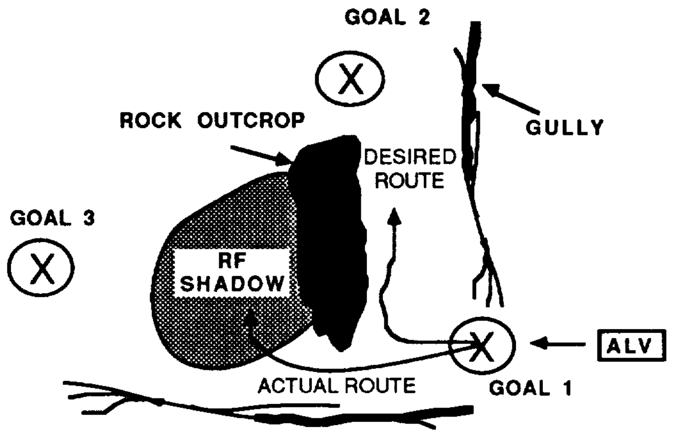


Figure [plan error]: Errant vehicle action while executing its route plan.

This example highlights the system's inability to take opportunistic advantage of unexpected situations when such situations are not properly accounted for in the abstract plan. We know from our understanding of the mission constraints that Goal 2 was merely an intermediate waypoint intended to keep the vehicle away from the RF shadow. Looking at the abstract plan in isolation, however, there is no way of knowing why a particular sub-goal has been established. The Goal 2 location could just as easily have been a critical choke point along the only path to Goal 3. It is only through our understanding of the underlying mission constraints that we can both identify the vehicle's failure to turn right and see the opportunity that arose as a result.

The apparent shortcoming of the abstract route plan is that it lacks environmental and mission constraints that are quite evident in the map. A more suitable plan would have explicated the concerns about staying out of the RF shadow. We therefore might wish to add more of this type of information to the plan. Once we start augmenting the plan, however, we have to ask how we might ever know when a sufficient amount of information has been added to prevent other types of mistakes. Consider, for example, the system's failure to realize that the intermediate sub-goal could be skipped when the opportunity arose. The problem arises because the true purpose of the sub-goal was never indicated. However, if the state-space of the plan could be expanded to include all the reasons for when and why the particular sub-goal was significant, then the location itself would become inconsequential. Consequently, the simple sequence of sub-goals is both an overspecification and an underspecification. The problem is inherent in any attempt to build an abstraction of the map data.

III. Avoiding Unnecessary Abstraction

In order to minimize the amount of information lost in forming a plan for action, it is best if all relevant knowledge is organized with respect to a given problem and then, without any further abstraction, provided in full for use in real-time decision-making. In order for this to be possible, the plan must no longer be viewed as a program for action, but rather, as a resource to help guide the decision-making process. When this viewpoint is adopted, there is no longer a need to translate plans into awkward representations for action. Instead, the original state-space in which the plan is formulated can be retained, enabling the plan to provide advice to decision-making processes whenever the current state of the system can be identified within that state-space. We refer to plans formulated and used in this manner as *internalized plans*, since they embody the complete search and look-ahead performed in planning, without providing an abstracted account of an explicit course of action [Pa].

The difference between the use of internalized plans and conventional abstracted plans is best illustrated in the context of the previous example. In contrast to the abstract route plan, consider a gradient description of a plan to achieve the same objectives. As illustrated in Figure [grad], there is no explicit plan shown, yet one can always find the best way to reach the goal simply by following the arrows. Such a representation would not ordinarily be thought to be a plan because it provides no specific course of action. As a resource for guiding action, however, the gradient field representation is extremely useful. No matter where the vehicle is located, and no matter how it strays from what might have been the ideal path, turn decisions can always be biased in favor of following the arrows.

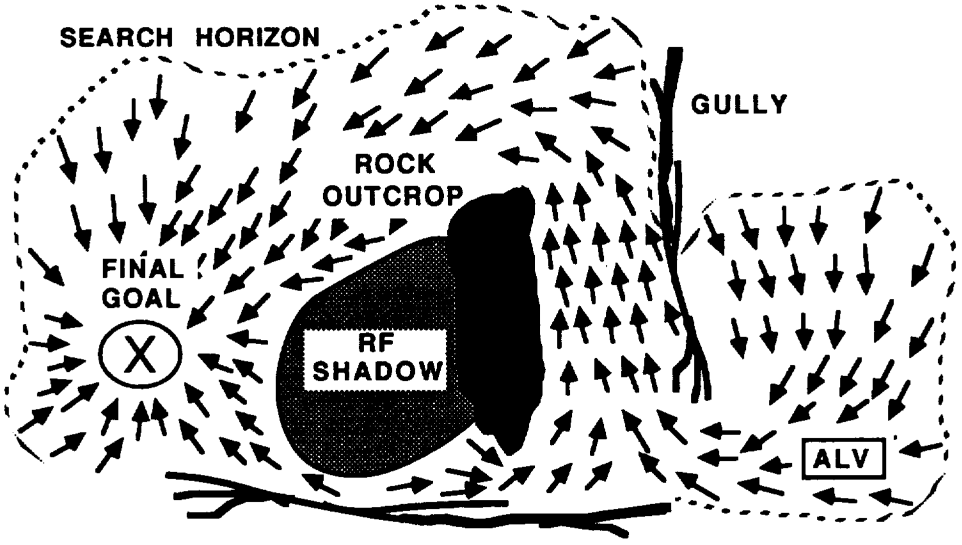


Figure [grad]: A gradient field representation provides one form of internalized plan.

Upon closer examination of Figure [grad], we can see not only how the mistake of entering the RF shadow could be avoided, but we see also how the system could be opportunistic should the vehicle happen to enter the shadow and be able to continue onward. First, when the vehicle had to make a choice between going left or right near the bottom of the rock outcrop, the gradient field would strongly bias its decision in favor of going right. If the vehicle got too close to the shadow on the left, the gradient field would actually be telling it to turn around. Further, should the vehicle happen to be forced to go below the rock outcrop and enter the RF shadow, then it would continue to be directed toward the final goal despite the radical deviation from its expected path. This type of behavior is opportunistic in that the vehicle is not constrained to reach any arbitrary pre-established sub-goals, and therefore all action can be directed exclusively toward achieving the mission objectives.

A more dramatic illustration of the difference between a conventional route plan and an internalized plan can be seen in problems requiring the attainment of any of several possible goals. This type of problem is often referred to as the "Post Office Problem" [Ed] because it can be likened to the task of finding the shortest route to the nearest of several post offices in a neighborhood. In the example shown in Figure [multi goal], the mission requires that the vehicle reach either of two distinct goal locations. The resultant gradient field is computed by propagating a search wavefront simultaneously from each of the two goals. As the wavefronts meet at a Voronoi edge, a ridge is created in the gradient field which will cause the vehicle to be guided toward one goal or the other depending on which side of the ridge it happens to be located.

Clearly, it would be difficult for an abstract route plan to capture the essence of choice contained in the gradient field representation. If we were to produce a route plan, we would invariably have to select a route to the closest goal, as shown in Figure [multi goal]. Once such a choice is made, however, we have discarded all that is known about the alternate goal even though that goal was nearly as close as the one selected. In contrast, by using the gradient field directly, the choice of goals may be made during the execution of the mission. Without having made an *a priori* selection of goals, the best choice may be made at every instant in time, regardless of how the vehicle might stray while avoiding obstacles.

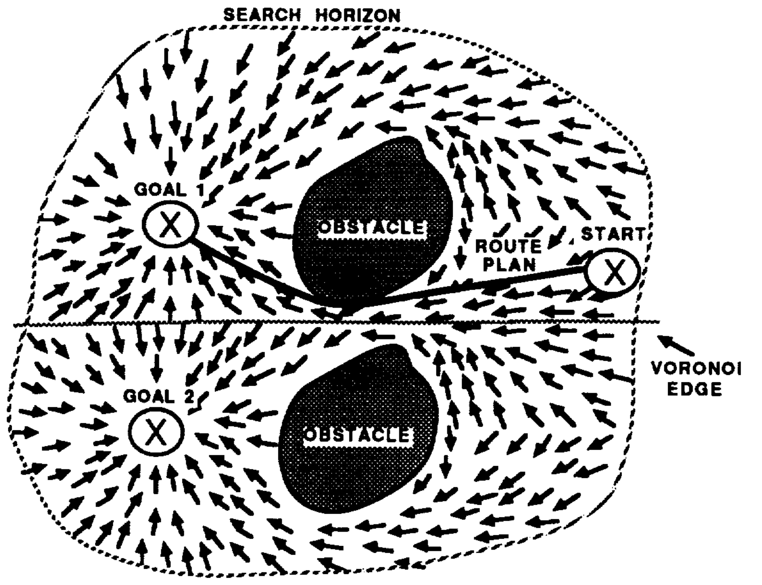


Figure [multi goal]: The gradient field provides a useful internalized plan for reaching either of two goals.

The gradient field is an ideal example of an internalized plan because the map-grid state-space in which the original problem is formulated is the same state-space in which the plan is represented. The gradient field, in fact, is a natural by-product of existing route planning algorithms [MPK]. These algorithms begin by assigning a cost to each grid cell of a digital terrain map. By associating high costs with locations that are undesirable according to mission criteria, a combination of mission constraints can be represented. Whether an A^* [Ni], or Dijkstra [Di] search algorithm is employed in the cost grid, the net result of the search is a score for each grid cell, indicating the minimum cost remaining to get from that cell to the goal. From any given grid cell, the best incremental step to get to the goal is the neighboring grid cell which has the lowest score. Ordinarily, when we use these scores to compute a standard route plan, we simply begin at the starting point and locally choose the lowest-score adjacent cell until we finally reach the goal. The record of our steps along the way gives us the minimum cost path to the goal. If we look at these scores in a slightly different way, we see that the best path to the goal from any grid cell may be determined by selecting the direction of the lowest-score adjacent cell. Thus, without any further abstraction, search in the map-grid can provide a useful resource for action.

IV. Using Plans as Resources

The method of use of a gradient field is an important factor in establishing it as an internalized plan representation. Since a digital terrain map generally cannot provide adequate resolution to support detailed maneuvering around small obstacles, there is inevitably a need to incorporate the advice provided by the gradient field into real-time decision-making processes which are attending to immediate sensory data. While, ordinarily, a single abstract route plan is generated, some approaches have taken advantage of a gradient field in order to quickly generate new route plans should the constraints of an initial plan be violated [LMD] [CF]. Problems with establishing and monitoring these constraints, however, are still unavoidable. In contrast, use of the gradient field as an internalized plan requires that the real-time decision-making processes continuously attempt to locate the system within the state-space of the plan and bias each decision in favor of the recommended course of action. The absence of an explicit course of action means that no arbitrary plan constraints need be established or monitored. The plan is a resource, providing suggestions for preferred action but never actually controlling the system. If, for any reason, no suggestion is available from the plan, the real-time decision-making processes must proceed in a reasonable manner on their own accord.

Another vector field type of representation, the *artificial potential field*, appears superficially very similar to the gradient field and it also is used for robot navigation and obstacle avoidance [Kr][Kh][Ar]. The basic differences, though, between how these two types of representations are constructed and used sheds further light on what it means for a plan to serve as a resource for action. The computation of potential fields is generally based on a superposition model in which charges are distributed such that repulsive forces are generated near obstacles and attractive forces are generated near goals. Superposition allows the potential field vector at any point to be computed quickly by adding up the contributions from each charge. The resultant field, however, does not represent an optimal path, and may easily contain local minima and traps. In contrast, the gradient field is computed from a more time consuming graph search process. As a result of this search, the gradient field has no local minima and will always yield the set of all optimal paths to the goal.

A more significant distinction between gradient fields and potential fields, however, is in how they are used. Often, when potential field methods are employed for navigation, the potential field is used for direct control of action. All sensory information is compiled into a single representation which is suitable for modeling an appropriate distribution of charges. The local potential field forces are then continuously computed at the location of the vehicle, and these forces are used directly to compute the desired motion. On the other hand, as internalized plans, gradient fields are never used to provide direct control of the vehicle. Instead, they are merely an additional source of information provided to a set of real-time decision-making processes. Since these processes can make use of many disjointed representations of the world in order to control the vehicle, there is never a need for all features of the environment to be abstracted into a single representational framework.

It is helpful to view internalized plans as though they were sources of supplementary sensory input data. From this perspective, it is clear that action is not controlled by plans any more than it is by sensory input. Instead, the system must be viewed as an entity which interacts with its environment, responding to both internal and external information sources. The gradient field plan, for example, can be thought of as a phantom compass that always gives a general idea of the right way to go. Just like other sensors, data from this internal sensor influences action but is never used to the exclusion of other sensory data. At any given time, however, a single information source can have significant influence over system behavior if need be. Just as an external sensor can be used to ensure that the vehicle never runs into obstacles, an internalized plan can be used to ensure that mission constraints are not violated. Thus, despite the fact that there is no top-down control, the system can adhere to high level mission requirements.

V. Multiple Internalized Plans

A significant advantage of using internalized plans as resources for action is that it is possible to use multiple internalized plans simultaneously. Each plan can contribute an additional piece of advice which can enhance the overall performance of the system. In this way, different plans may be formulated in incompatible state-spaces without the need to merge these state-spaces through abstraction.

We can consider as an example, the combined use of map-based plans with plans based on symbolic mission constraint data. In the case of the RF shadow problem, a constraint to maintain radio contact may be derived from mission knowledge. If this knowledge is used in conjunction with a signal strength sensor, then whenever the vehicle enters an RF shadow, it can immediately back up in order to regain contact. In the absence of such problems, the gradient field produced from map data can constantly provide advice on which way to go. An unexpected loss of radio contact would then be treated much like an encounter with an obstacle. The vehicle would have to make special maneuvers in order to regain contact and ensure that the same mistake would not be repeated. After this, the map-based plan would regain primary influence.

There are also many cases in which it might be desirable to use multiple internalized plans formulated within the same state-space. For example, a gradient field plan could be augmented with information about the amount of fuel and time required to get from each grid-square to the goal. While this information could not directly indicate a course of action, it might allow available fuel and time resources to be monitored constantly and compared with expected needs. If there were barely enough fuel to succeed but plenty of time available, the vehicle might be able to switch to a simple fuel conserving strategy such as reducing its speed. If time and fuel were both in short supply, the gradient field might need to be re-computed, placing more emphasis on conserving fuel and time resources and possibly less emphasis on other factors such as vehicle safety.

Another form of internalized plan exploits the map as a resource for action by probing it directly during execution. As the vehicle is traveling, the portion of the map corresponding to the area just in front of the vehicle is examined to determine what types of features should be detected. This understanding of the local environment can have a direct bearing on how sensor data is interpreted for action. Remember, for example, the problem illustrated earlier in Figure [plan error]. Here, one of the main reasons the vehicle failed to avoid the RF shadow was that its sensors indicated a clear path in this area. This error could be overcome by differentiating between obstacles that are observable and those that are not, and then appropriately discounting sensor readings that are known to be inapplicable. Thus, by treating the map as if it were a sensor, the value of real sensor data can be greatly enhanced.

A great diversity of behavior may also be gained by dynamically combining information from multiple gradient fields. Consider, for example, two independent gradient fields, one which can guide a vehicle along a safe, well hidden route, and another which can lead the vehicle to nearby observation points. We can imagine that the vehicle is guided by the safe gradient field until the time comes for it to make an observation. Then, the gradient field for getting to observation points would become the primary guiding factor. Such a gradient field, formed similar to the field in Figure [multi goal], would lead the vehicle to the nearest of several possible observation points. Once an observation point had been reached and observation data collected, the safe gradient field would again be used for guidance. Using such a combination of internalized plans allows the performance of tasks that would be difficult to accomplish with a symbolic plan. Without an explicit plan for action, it is the interplay between the vehicle and its environment that determines how the mission will ultimately be carried out.

VI. Conclusion

Although abstraction is necessary if we are to provide organization and structure to the vast amounts of information available to an intelligent agent, we have seen examples in which the abstraction of plans can obscure their true intent and result in serious failures. In light of these issues we must ask whether forming the abstraction was really necessary or whether it was merely an artifact of an approach in which plans are regarded as programs rather than as resources for action. Using internalized plans, we have shown that with no abstraction of the map-based plan, we can obtain an ideal resource for action.

Just as the grid of a digital terrain map is an abstraction of the Earth's surface, abstraction may be used to create other state-spaces which are suitable to use for planning. In many cases, however, it may be best not to attempt the fusion of information from different sources if an excessive degree of abstraction is required to do so. Instead, state-spaces should be formed to suit the type of information available, and once planning is performed in these state-spaces, no further abstraction of the results should be performed. The unabstracted product of planning search provides a measure of desirability for transitions from one state to the next, and this measure may be used directly as a resource for action.

VII. Acknowledgments

Many of the concepts presented in this paper were fostered through frequent discussions with David M. Keirsey, J. Kenneth Rosenblatt, and Charles Dolan. I am extremely grateful for their contributions and insights. The advice of Joseph Mitchell and Rodney Brooks was also very helpful in bringing to light many important areas of related work. Many thanks also to my wife Karen and Jimmy Krozel for their editing assistance.

VIII. References

[AC1] Agre, P., and D Chapman, "Pengi: An implementation of a Theory of Activity", *Proc. of the Sixth National Conf. on Artificial Intelligence*, Seattle, Washington, July, 1987, pp. 268-272.

[AC2] Agre,. P., and D. Chapman, "What are plans for?" AI Memo 1050, MIT Artificial Intelligence Laboratory, 1987.

[Ar] Arkin, R., "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", *IEEE Conf. on Robotics and Automation*, March 1987, pp. 264-271.

[Br] Brooks, R. A., "Intelligence Without Representation," *Preprints of the Workshop in Foundations of Artificial Intelligence*, Endicott House, Dedham, MA, June, 1987.

[CF] Chan, Y.K., and M. Foddy, "Real Time Optimal Flight Path Generation by Storage of Massive Data Bases", IEEE National Aerospace and Electronics Conf. (NAECON), Dayton OH, May 1985.

[Da] Daily, M., J. Harris, D. Keirsey, K. Olin, D. Payton, K.Reiser, J. Rosenblatt, D. Tseng, and V. Wong, "Autonomous Cross-Country Navigation with the ALV," *Proceedings of DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987, (also appearing in *Proceedings of IEEE Conference on Robotics and Automation*, Philadelphia, PA., April, 1988.)

[Di] Dijkstra, E.W., "A Note on Two Problems in Connection with Graph Theory", Numerische Mathematik, Vol 1, 1959, pp. 269-271.

[Ed] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin, 1987, pp 298-299.

[KPR] Keirsey, D.M., D.W. Payton, and J.K. Rosenblatt, "Autonomous Navigation in Cross Country Terrain", *Proceedings Image Understanding Workshop*, Boston, MA, April, 1988.

[Kh] Khatib, O. "Real Time Obstacle Avoidance for Manipulators and Mobile Robots", *IEEE Conf. on Robotics and Automation*, March 1985, pp. 500-505.

[Kr] Krogh, B.H., "A Generalized Potential Field Approach to Obstacle Avoidance Control", *Robotics International Robotics Research Conference*, Bethlehem, PA, August 1984.

[LMD] Linden, T.A., J.P. Marsh, and D.L. Dove, "Architecture and Early Experience with Planning for the ALV", *IEEE International Conf. on Robotics and Automation*, April, 1986, pp. 2035-2042.

[MPK] Mitchell, J.S.B., D.W. Payton, and D.M. Keirsey, "Planning and Reasoning for Autonomous Vehicle Control," *International Journal for Intelligent Systems*, Vol. 2, 1987.

[Ni] Nilsson, N.J., *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.

[Pa] Payton, D.W., "Internalized Plans: a representation for action resources", Workshop on Representation and Learning in an Autonomous Agent, Lagos, Portugal, Nov. 1988.

[Su] Suchman, L., "Plans and Situated Actions: the problem of human macnine communication", Cambridge University Press, 1987.

N90-29029

LEARNED NAVIGATION IN UNKNOWN TERRAINS: A RETRACTION METHOD

Nageswara S.V. Rao
Dept. of Computer Science
Old Dominion University
Norfolk, VA 23529-0162

N. Stoltzfus
Dept. of Mathematics
Louisiana State University
Baton Rouge, LA 70803

S.S. Iyengar
Dept. of Computer Science
Louisiana State University
Baton Rouge, LA 70803

ABSTRACT

We consider the problem of learned navigation of a circular robot R, of radius δ (≥ 0), through a terrain whose model is not *a priori* known. We consider two-dimensional finite-sized terrains populated by an unknown (but, finite) number of simple polygonal obstacles. The number and locations of the vertices of each obstacle are unknown to R. R is equipped with a sensor system that detects all vertices and edges that are visible from its present location. In this context we deal with two problems. In the *visit problem*, the robot is required to visit a sequence of destination points, and in the *terrain model acquisition problem*, the robot is required to acquire the complete model of the terrain. We present an algorithmic framework for solving these two problems using a retraction of the free-space onto the Voronoi diagram of the terrain. We then present algorithms to solve the visit problem and the terrain model acquisition problem.

1. INTRODUCTION

We consider the problem of the collision-free navigation of a circular robot through an *unknown* terrain, i.e., a terrain whose model is not *a priori* known. Several variants of this problem have been investigated. An algorithm for a point robot to escape out of a maze using touch sensing ability is given in [1]. In [6], algorithms for a point robot to move from a source point to a destination point using touch sensing are presented. The algorithms that enable a point robot to navigate to a destination point, and at the same time "learn" about the parts of terrain that are encountered on the way to the destination are presented in [2,7]. This process of learning is termed as *incidental learning*. Here the robot uses a combination of touch sensing and distance probing. The same problem is also solved in the case where the point robot is equipped with a sensor that obtains all the visible obstacle boundaries [8]. The above problems can be grouped under a generic name of the *visit problem*, wherein a robot is required to visit a sequence of destination points through an unknown terrain. Another problem, called the *terrain model acquisition problem*, wherein a point robot is required to acquire the complete model of the terrain is also studied [10]. The solutions of [7,8,10] are based on an incremental construction of the visibility graph of the terrain. The above problems have to be distinguished from those that deal with the navigation in *known* terrains, i.e. the terrains whose models are available. A comprehensive treatment of these problems can be found in [11].

These formulations of the navigational problem are motivated by a practical application involving the development of an autonomous rescue robot. This robot is intended for carrying out rescue operations in nuclear power plants in the events of radiation leakages, and other incidents that prevent human operation. A solution to the visit problem helps in developing a robot that can carry out a set of operations in different locations in unfamiliar environments. Since the motion planning here is essentially sensor-based, the navigation involves expensive sensor operations. Further more, the robot could temporarily navigate into local detours because of the partial nature of the information returned by the sensors. By incorporating the incidental learning feature, we reduce the expected number of sensor operations, and the expected number of detours, as the robot visits newer locations. Instead, if the complete terrain model is available, the robot can avoid the local detours, and also avoid the expensive sensor operations. Thus a solution to the terrain model acquisition problem helps the robot in acquiring the terrain model during the period in between the rescue operations. A dedicated rescue robot typically idles in between two successive rescue operations, and the rescue operations could be fairly infrequent. In such cases, the resources are better utilized if the robot is employed in the terrain model acquisition process during this period. The proposed methodology in solving these navigational problems provides a basic algorithmic framework that aids the design of a navigational system for the abovementioned rescue robot. The same methodology can also aid the development of navigational systems for other autonomous mobile machines in applications such as space navigation, underwater explorations, maintenance of space laboratories etc. However, a practical implementation of the proposed system calls for advances in more general theoretical aspects as well as several other issues such as sensing and movement errors, etc., which are not discussed in the above works as well as in this paper.

The visit problem and the terrain model acquisition problem have been solved separately [7,8,10]. In this paper, we present a unified framework for solving both the problems using a method based on a retraction of free-space onto the Voronoi diagram of the terrain. In this framework, we use the single approach of implementing a graph search algorithm on a graph, called the *navigational course*. We deal with a circular robot as opposed to the point robot of earlier works. Moreover, this method has an advantage of keeping the robot as far away from the obstacles as possible. This aspect seems very important in practical implementations as the earlier methods, based on the visibility graph methods, may require that the robot navigate along the obstacle boundaries. Additionally, the proposed method results in a storage complexity of $O(N)$ as opposed to $O(N^2)$ of visibility graph based methods [7,10]. Also, this method results in a path-planning complexity of $O(N^2\sqrt{\log N})$, whereas the visibility graph method has a complexity of $O(N^3)$ for the same. In this paper, we present briefly present our results and the details can be found in our report [9].

The organization of the paper is as follows: The basic framework of our solution is outlined in Section 2. In Section 3, we present the definition and properties of the navigation course to be used for navigational purposes. In Section 4, we first present solutions for the visit problem, and the terrain model acquisition problem.

2. BASIC ALGORITHM

We first describe the problem scenario and then present the basic algorithm used in the solution of the visit problem and the terrain model acquisition problem.

Terrain: We consider a finite-sized two-dimensional *terrain* populated by a finite set $O=\{O_1,O_2, \cdots O_n\}$ (n is finite) of simple disjoint polygons, called the *obstacles*. Each obstacle O_i has a finite number of vertices. The terrain in completely *unknown* to R, i.e., the number of obstacles, and also the number and locations of vertices of each obstacle are unknown to R. The free-space is given by $\Omega=\bigcap_{i=1}^{n} O_i^C$, where O_i^C is the complement of O_i in the plane. The closure of the free-space is denoted by $\overline{\Omega}$. Let N denote the total number of vertices of all obstacles. Let $VER(O_i)$ denote the set of vertices of O_i.

Robot: We consider a circular body R of radius δ, $(\delta \geq 0)$. The location of the center of R is called the *position* of R. We treat R as an open disc of radius δ centered at the position of R. R houses a computational device with storage capability. Also, R is capable of moving along a straight-line path or a curved path of second degree (in each case the path is specified). R takes a finite amount of time to move through a finite amount of distance. Further, R is equipped with an algorithm B that plans a collision-free path (for R) through a known terrain. In particular, we can use a suitable algorithm from [11] for this purpose.

Sensor System: Let x be a position of R. A point $y \in \overline{\Omega}$ is said to be *visible* to R if the straight line joining x and y is entirely contained in $\overline{\Omega}$. R is equipped with a sensor that detects the maximal set of points on the obstacle boundaries that are visible from its present location. Such an operation is termed as the *scan* operation. We assume that the scan operation is precise and error-free.

Two Navigational Problems

Initially, R is located at a point d_0 without intersecting any obstacle polygons and at a finite distance from an obstacle. In the *terrain model acquisition problem*, R is required to acquire the model of the terrain to a degree such that it can navigate to any reachable destination location by planning a path using the known terrain algorithm B alone. If a destination position is not reachable then R should report this fact without performing sensor operations. Note that after the terrain model is completely acquired, no sensor operations are needed for navigational purposes. Second, in the *visit problem*, R is required to visit the points $d_1, d_2, \cdots, d_N$ in the specified order if there exists a path through these points. If no such path exists, then R must report this fact in a finite amount of time.

Navigation strategy

We now present the algorithm *NAV* which is the basic underlying strategy used by R to solve the visit problem and the terrain model acquisition problem. Here, R performs a "graph exploration type" of navigation using a combinatorial graph called the *navigation course*, $\xi(O)$, of the terrain O. $\xi(O)$ is a 1-skeleton embedded in Ω. The nodes (edges) of $\xi(O)$ are called ξ-nodes (ξ-edges). Each ξ-node specifies a collision-free position for R, i.e. a position for R such that it is entirely contained in Ω. An edge that joins two x-nodes v_1 and v_2 specifies a collision-free path, of finite length, from v_1 to v_2 for R. The $\xi(O)$ is initially unknown and it is incrementally constructed using the data obtained through the sensor operations. The algorithm *NAV* is given below:

Consider the execution of *NAV* by R. *NAV* in initiated with a ξ-vertex $v = v_0$ and $S_2 = \{v_0\}$. The set S_1 contains all the ξ-vertices that are visited by R. The set S_2 contains all the ξ-vertices that not visited by R, but each $v \in S_2$ is adjacent to some ξ-vertex in S_1. R keeps visiting new ξ-vertices until the set S_2 becomes empty. When R visits v for the first time the adjacency list of v is computed. In this way, the $\xi(O)$ is incrementally constructed. The scan operation of line 1 and the computational operations in lines 2-7 and 11-12 can be directly executed by R. The path planning of line 8 involves finding a graph path from v to v^*. R actually moves along the edges of the computed path in line 9.

S_1 forms a connected (graph) component with the edge set being the set of all edges that are traversed by R. Further $S_1 \cup S_2$ forms a connected (graph) component with the edge set being the union of the set of all edges traversed by R and the set of all edges computed by R. Thus there exists a path along the edges (of the component) from any vertex of S_1 to any vertex of S_2. In each step, R, located at v, selects a $v^* \in S_2$, and then it moves to v^*. In this aspect, *NAV* is similar to a standard graph exploration algorithm except for one difference. In a graph algorithm the cost associated with accessing the node v^* (after it is chosen) is a single memory access. In *NAV*, when R accesses v^* there are two associated costs: (a) the computational cost of planning a path from v to v^* (b) the cost of moving R along the computed path.

```
algorithm NAV(v);
  begin
1.  perform a scan operation from $v$;
2.  mark $v$ as visited and delete it from $S_2$ and append to $S_1$;
3.  compute the adjacency list of $v$;
4.  append to $S_2$ all neighbors of $v$ that are not visited;
5.  if ($S_2$ is not empty)
6.  then
7.     select $v^* \in S_2$;
8.     plan a path from $v$ to $v^*$;
9.     move to $v^*$;
10.    $NAV(v^*)$;
11.  else
12.    return to start vertex $v_0$;
     endif
  end;
```

In order to yield correct solutions to the visit problem and the terrain model acquisition problem, the navigational course $\xi(O)$ has to satisfy a set of properties. These properties for the proposed $\xi(O)$ are discussed in detail in the next section. Suppose that $\xi(O)$ satisfies the property of *local-constructibility*, i.e., the adjacency list of a ξ-vertex v can be computed from the information obtained by a scan operation performed from v. Further, suppose that $\xi(O)$ satisfies *finiteness* property, i.e., $\xi(O)$ has finite number of vertices. Also let $\xi(O)$ satisfy graph *connectivity* property, i.e., any two ξ-vertices are connected by a path of ξ-edges. Then we have the following observation.

Observation 1: *If $\xi(O)$ satisfies the properties of finiteness, connectivity and local-constructibility, then, R, executing the algorithm NAV, visits all vertices of $\xi(O)$ in a finite amount of time.* □

3. THE NAVIGATIONAL COURSE

We first present a structure that yields a navigational course to be used by a point robot. We then extend our discussion to a circular robot.

3.1. Point Robot

For $x \in \Omega$, we define $Near(x)$ as the set of points that belong to the boundaries of obstacles O_i, $i=1,2, \cdots ,n$ and are closest to x. The *Voronoi diagram*, $Vor(O)$, of the terrain populated by O is the set of points:

$\{x \in \Omega | Near(x)$ contains more than one point $\}$

In this case, $Vor(O)$ is a union of $O(N)$ straight lines and parabolic arcs (see [4,5] for more details). Each of this line or parabolic arc is referred to as *V-edge*. The points at which the edges meet are called *V-vertices*. Furthermore, $Vor(O)$ can be specified as a combinatorial graph in which each edge is labeled with two end V-vertices, and an equation defining it as a curve in the plane. Each V-vertex is labeled with its coordinates.

Consider the convex hull $C(O)$ of union of vertices of all obstacles (i.e. convex hull of $\bigcup_{i=1}^{n} VER(O_i)$). Let $E(O)$ denote the polygonal region obtained by pushing the edges of $C(O)$ out-

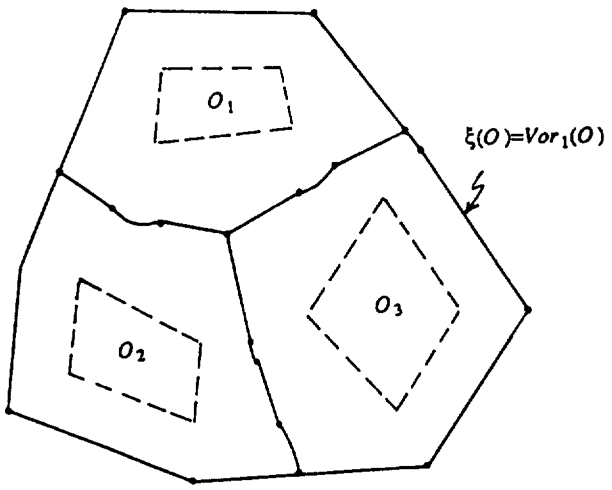


Figure 1. The $Vor_1(O)$ for the terrain $O=\{O_1,O_2,O_3\}$.

wards by a distance of s and taking the interior of 'grown' region. Let us define $Vor_1(O)=(Vor(\Omega)\cap E(O))\cup\partial E(O)$, where $\partial E(O)$ is the boundary of $E(O)$. We note that $Vor_1(O)$ precisely contains the Voronoi diagram of O that lies inside $E(O)$ and the boundary of $E(O)$. The edges (vertices) of $Vor_1(O)$ are called V_1-edges (V_1-vertices). See Fig. 1 for an example. The set of vertices of $Vor_1(O)$ is the union of V-vertices, vertices of the envelop $E(O)$ and intersection points of edges of $\partial E(O)$ with V-edges. Similarly the set of edges of $Vor_1(O)$ is the union of edges of $Vor(O)$ that are contained in $E(O)$ and the edges of $\partial E(O)$. It is easy to see $Vor_1(O)$ as a planar graph formed by V_1-vertices and V_1-edges. The set of all V_1-vertices that are adjacent to a V_1-vertex v constitute the set of *neighbors* of v. The following four basic properties of $Vor_1(O)$ are shown in [9]:

(i)*Combinatorial properties*: The number of V_1-vertices is at most $4N-n-2$, and the number of V_1-edges is at most $6N-3n-3$.

(ii)*Connectivity*: $Vor_1(O)$ is topologically connected, and consequently $Vor_1(O)$ is graph connected when viewed as a combinatorial graph i.e., there exist a path along V_1-edges between any two V_1-vertices.

(iii)*Terrain-visibility*: Every point in the closure of free-space Ω is visible from some V_1-vertex, i.e., for $x\in\overline{\Omega}$, there exist a V_1-vertex v such that the line joining x and v is entirely contained in $\overline{\Omega}$.

(iv) *Local-constructibility*: All the neighbors of a V_1-vertex v can be correctly computed from the terrain boundary information obtained by performing a scan operation at v.

3.2. Circular Robot

For $x\in\Omega$, let $Clearance(x)$ denote the distance of x from a member of $Near(x)$ (in terms of the Euclidean distance). Consider $Vor_1(O)$ such that the distance s used in obtaining $E(O)$ is at least δ. Let us consider a subset of $Vor_1(O)$ given by $\{x\in Vor_1(O) \mid Clearance(x)>\delta\}$ which is the set of points of $Vor_1(O)$ with clearance greater than δ. This set consists of a set of connected components. Initially, let R be located at $d_0\in\Omega$. Let $Vor^*{}_1(O)$ be the connected component that contains $\mathrm{Im}(d_0)$, i.e., $Im(d_0)\in Vor^*{}_1(O)$. $Vor^*{}_1(O)$ contains either all or none of the edges of $E(O)$. Further an edge of $Vor^*{}_1(O)$ could be a truncated version of an edge of $Vor(O)$, in which case we attach a vertex at the truncated end. These vertices are called *truncated vertices*. The edge formed as a result is called the *truncated edge*. We now summarize the properties of $Vor^*{}_1(O)$.

Properties 2: *$Vor^*_1(O)$ satisfies the properties of finiteness, connectedness, terrain-visibility and local-constructibility.*

4. NAVIGATION ALGORITHMS

We first discuss the navigational course and the navigation strategy used by R. Then we present the algorithm *ACQUIRE* that solves the terrain model acquisition problem. We then present the algorithm *LNAV* that navigates R from d_i to d_{i+1} if a path exists from d_i to d_{i+1}. Then we obtain the algorithm *GNAV* that solves the visit problem. *GNAV* uses *LNAV* as a component and also incorporates the feature of incidental learning.

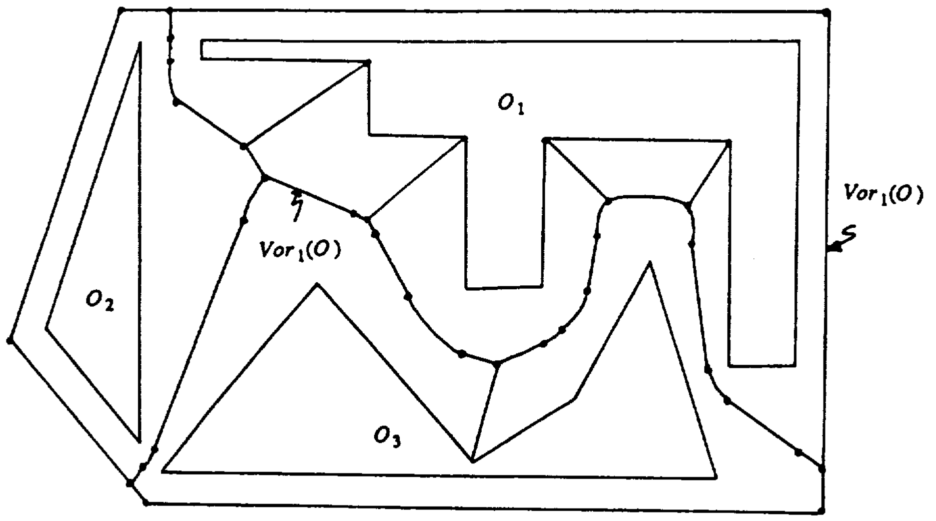


Figure 2. The terrain O={O$_1$,O$_2$,O$_3$} and Vor$_1$(O).

4.1. Preliminaries

For a point robot, $\xi(O)$ is obtained by deleting from $Vor_1(O)$ all the V_1-edges that terminate on a concave corner. Such edges are formed by two obstacle edges that meet at a concave corner. For a circular robot $\xi(O)$ is obtained by deleting from $Vor^*_1(O)$ all the truncated edges. In Fig. 2, we show the terrain $O=\{O_1,O_2,O_3\}$, and the corresponding $Vor_1(O)$. In Fig. 3, we show $\xi(O)$ for a circular robot. Note that every V-edge that terminates at a concave corner generates a truncated edge. We assume that the process of deletion of an edge retains the vertex that connects the edge to the rest of $Vor_1(O)$ or $Vor^*_1(O)$. Now, view $\xi(O)$ as 1-skeleton embedded in the plane. It is clear from the definition that any point on $\xi(O)$ - in particular a ξ-vertex - specifies a collision-free position for R. Consequently, a ξ-edge specifies a collision-free path for R. It is direct to see that $\xi(O)$ satisfies the properties of finiteness and local-constructibility. The connectivity of $\xi(O)$ can be shown by observing that each edge that is removed from $Vor_1(O)$ and $Vor^*_1(O)$ is pendant and can not disconnect resultant set. Let C denote the number of concave corners of the terrain O. Note that we delete (at least) C V-edges and V-vertices from $Vor_1(O)$ ($Vor^*_1(O)$) for a point (circular) robot. Then we have the following properties.

Properties 3: $\xi(O)$ *for a point or a circular robot satisfies the properties of finiteness, connectivity, terrain-visibility and local-constructibility. Further more*

(i) #ξ-vertices $\leq 4N-n-C-2$

(ii) #ξ-edges $\leq 6N-3n-C-6$. □

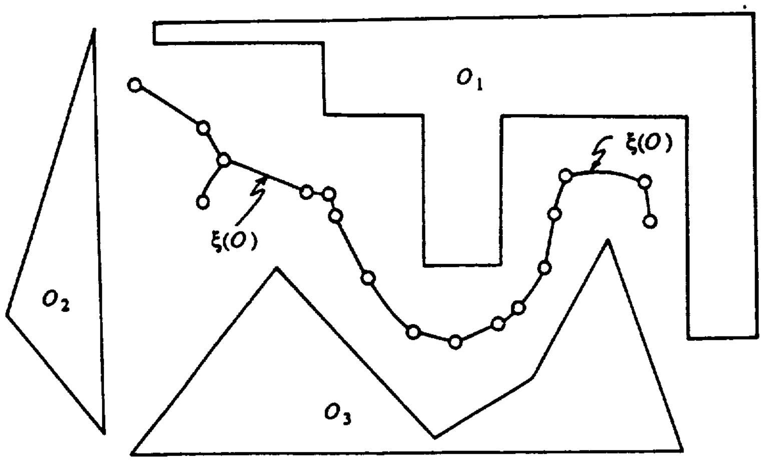


Figure 3. ξ(O) for a circular robot for O of Fig. 2.

Now consider the execution of the algorithm *NAV*. For ease of presentation, we discuss a depth-first implementation of *NAV* which specifies a particular way to select v^* the ξ-vertex to be visited next (line 7). R is presently located at the ξ-vertex v. If v has neighbors that are not visited then R chooses one of the unvisited neighbors as v^*. If all neighbors of v_* are visited then R chooses the vertex of S_2 that is reachable by a minimal distance path. This path is obtained by invoking one-to-all shortest path algorithm of [2] on the presently available $\xi(O)$ and picking the vertex that is reachable from v by a path of minimal length. Note that $\xi(O)$ is a planar graph. Cost of this computation is $O(N\sqrt{\log N})$.

4.2. Terrain Model Acquisition

The algorithm *ACQUIRE* is a direct implementation of the algorithm *NAV*. Once the terrain model is available one can use the algorithm B to plan a path to reach any destination point. This algorithm has a time complexity of $O(N\log N)$ [11]. Thus we have the following theorem.

Theorem 1: *The algorithm ACQUIRE solves the terrain model acquisition problem in a finite amount of time such that*

(i) *The number of scan operations performed is at most* $4N-n-C-2$.

(ii) *The total distance traversed by R while executing ACQUIRE is at most twice the total length of the depth-first tree of $\xi(O)$ rooted at v_0.*

After the execution of ACQUIRE, R can navigate to any reachable destination with a time complexity of $O(N\log N)$ and with no sensor operations. □

In our implementation we use the adjacency list representation of $\xi(O)$. We store the coordinates of each ξ-vertex in the adjacency lists. We maintain a table called MAP-TABLE as an AVL tree. The cost of this operation is $O(\log N)$ using the table.

Theorem 2: *The complexities of various tasks carried out by ACQUIRE are as follows:*

(i) *the storage complexity is $O(N)$,*

(ii) *cost of construction of* $\xi(O)$ *is* $O(N^2\log N)$

(iii) *total cost of path planning is* $O(N^2\sqrt{\log N})$,

(iv) *cost of construction of MAP-TABLE is* $O(N\log N)$, *and total cost of accesses to MAP-TABLE is* $O(N\log N)$. □

4.3. Visit Problem

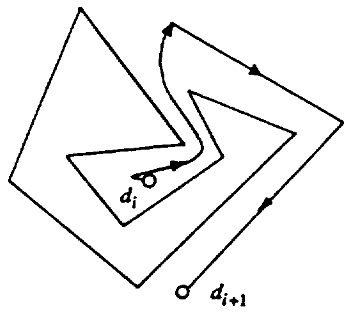
We now discuss the algorithm *LNAV* that navigates R from its present location at d_i to a destination point d_{i+1} if such path exists. If there is no path from d_i to d_{i+1}, then R will declare the same in a finite amount of time. The algorithm *LNAV* is obtained by modifying *NAV*. Initially a scan is performed from d_i and if d_{i+1} is found reachable, then R moves to d_{i+1}. If d_{i+1} is not found reachable then R computes a ξ-vertex v_0 and moves to v_0. From v_0, the algorithm *NAV* is invoked. Let R be located at v. After a scan is performed from v, R checks if d_{i+1} is reachable. If d_{i+1} is reachable, then R moves to d_{i+1} and terminates *NAV*. If not, R continues to execute *NAV* until completion. If d_{i+1} is not found after S_2 becomes empty then d_i is declared as not reachable.

Theorem 3: *Algorithm LNAV navigates R from* d_i *to* d_{i+1} *in a finite amount of time if the latter is reachable. If* d_{i+1} *is not reachable then R declares so in a finite amount of time. In executing the algorithm LNAV by R,*

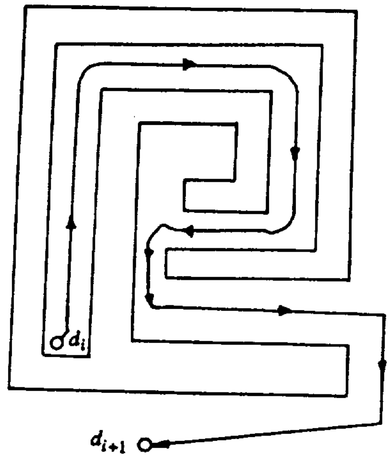
(i) *the number of scan operations is at most* $4N-n-C-2$,

(ii) *the total distance traversed is at most equal to twice the length of the depth first tree of* $\xi(O)$ *rooted at* v_0. □

The computational complexity of executing the algorithm *LNAV* follows along the lines of previous section. Thus, in executing the algorithm *LNAV*, (i) the storage required is $O(N)$, (ii) cost of construction of $\xi(O)$ is $O(N^2\log N)$, (iii) complexity of path planning is $O(N^2\sqrt{\log N})$ (iv) the cost of construction of MAP-TABLE is $O(N\log N)$, and the total cost of accesses to MAP-TABLE is $O(N\log N)$.



(a) R escaping out of a concavity



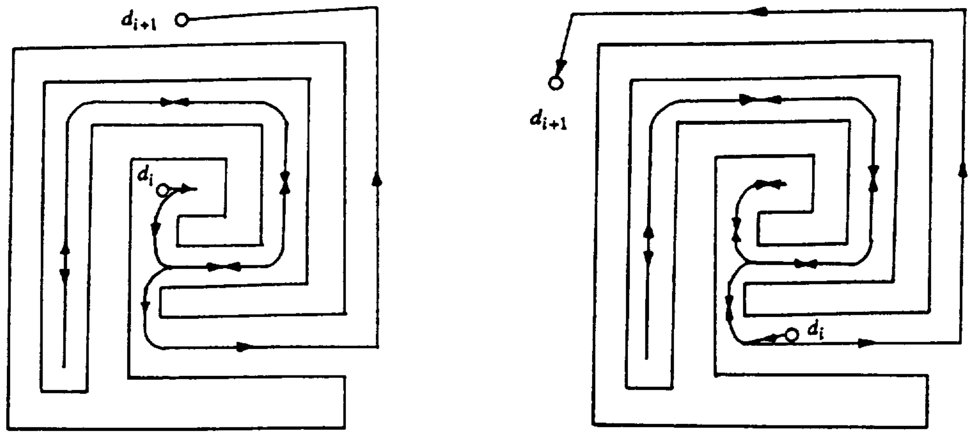
(b) R moving out of a maze

Figure 4. Execution of LNAV by R

In Fig. 4 we show a point robot moving out of a concavity, and moving out of a maze. In Fig. 5 we show a point robot moving out of a maze with backtracking. We can solve the visit problem by a repeated invocation of *LNAV*. *LNAV* completely relies on the sensor information for navigation.

Since the sensor obtains only a partial information about the terrain, as a result R might navigate into local concavities as in Fig. 5. If R is required to navigate in the regions that it navigated in previous traversals, then it can use the previous information to plan its present course of navigation. Note that the partial model of the terrain depends on the paths traversed by R in earlier traversals. We now obtain the algorithm *GNAV* as follows: We store the adjacency lists computed by R over the traversals in a global $\xi(O)$. Further the set S_2 is also stored over the traversals. Consider the navigation from d_i to d_{i+1}. Then *GNAV* computes a ξ-vertex that is reachable from d_i and moves to this vertex. Then R computes a ξ-vertex d^* that is closest to d_{i+1} according to some criterion such as distance. Then R moves along a path on $\xi(O)$ to d^*. From d^*, R uses *LNAV* to navigate to d_{i+1}. It is direct to see that *GNAV* correctly solves the visit problem. Moreover, R checks the set S_2 after every scan operation. After S_2 becomes empty, R switches-off its sensor and navigates the further traversals using the algorithm B alone. At this stage R has acquired the terrain model that is sufficient to navigate to any reachable point. Thus after this stage R does not perform scan operations for the purpose of navigation, and also R would avoid local concavities. Using the arguments of previous section it is clear that such stage will be reached after at most $4N+M-n-C-1$ scan operations. Thus we have the following theorem.

Theorem 4: *The terrain model will be completely built by R in at most* $4N+M-n-C-1$ *scans, then the execution of each traversal involves no scan operations with a time complexity of* $O(N\log N)$ □



(a) R backtracks once **(b) R backtracks twice**

Figure 5. Execution of LNAV by R.

Since the process by which R acquires the terrain is incidental, i.e., depends on the previous traversals it executed, it is possible to make probabilistic statements about the performance of *GNAV*. Let $\xi(O)=(V,E)$. Let p_v (>0) be the probability that R visits a ξ-vertex during any traversal. Probability that a scan is performed from v in ith traversal is $(1-p_v)^{i-1}p_v$. Then the probability that the terrain model will be complete during the mission of M traversals if $\prod_{v\in V}[1-(1-p_v)^M]$ which is non-zero. Moreover this probability approaches to 1 as M approaches infinity. Thus in a limiting case R obtains the complete terrain model with a probability of one.

5. CONCLUSIONS

We presented an algorithmic framework based on a retraction of free-space to solve two navigational problems for a circular robot moving in an unknown terrain. We consider the visit problem in which the robot is required to visit a sequence of destination points. We present an algorithm that enables the robot to visit the destination points using an ideal sensor, and also build the terrain model in the regions it navigates. Further the robot can detect the completion of the terrain model, and at this stage it switches to a known terrains navigation algorithm. After this stage, the future navigation is carried out without using the sensor. We also consider the terrain model acquisition problem wherein the robot is required to autonomously explore the terrain and build a model of the terrain such that the future navigation to any reachable destination can be carried out using the algorithms of known terrains alone.

References

[1] Abelson, H., and A. diSessa, (1980), "Turtle Geometry", MIT Press, 1980, pp. 179-199.

[2] G.N. Frederickson, Shortest path problems in planar graphs, Proc. 24th Ann. Symp. on Found. of Comput. Sci., 1983, pp. 242-247.

[3] Iyengar, S.S., C.C. Jorgensen, S.V.N. Rao and C.R. Weisbin, (1986), Robot navigation algorithms using learned spatial graphs, *Robotica*, vol. 4, January 1986, pp. 93-100.

[4] Kirkpatrick, D.G., (1979), Efficient computation of continuous skeletons, Proc. 20th Ann. Symp. on Found. of Comput. Sci., 1979, pp. 18-27.

[5] Lee, D.T. and S. Drysdale, (1981), Generalization of Voronoi diagrams in the plane, *SIAM J. Comput.*, vol.10, no.1, 1981, pp. 73-87.

[6] Lumelsky, V.J. and A.A. Stepanov, (1987) Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape, *Algorithmica*, vol.2, 1987, pp. 403-430.

[7] Oommen, J.B., S.S. Iyengar, N.S.V. Rao and R.L. Kashyap, (1987) Robot navigation in unknown terrains using visibility graphs. Part I:The disjoint convex obstacle case, *IEEE J. Robotics and Automation*, vol. RA-12, 1987, pp. 672-681.

[8] Rao, N.S.V., S.S. Iyengar and G. deSaussure, (1988a), The visit problem: Visibility graph-based solution, Proc. 1988 IEEE Int. Conf. Robotics and Automation, pp. 1650-1655.

[9] Rao, N.S.V., N. Stoltzfus and S.S. Iyengar, (1988b), A 'retraction' method for learned navigation in unknown terrains for a circular robot, Tech. Rep. #88-018, Dept. of Computer Science, Old Dominion University, 1988.

[10] Rao, N.S.V., S.S. Iyengar, J.B. Oommen and R.L. Kashyap, (1988c), Terrain acquisition by a point robot amidst polyhedral obstacles, *IEEE J. Robotics and Automation*, vol. 4, No. 4, 1988, pp. 450-455.

[11] Yap, C.K. (1987), Algorithmic motion planning, *in* "Advances in Robotics: Vol. 1: Algorithmic and Geometric Aspects of Robotics", Ed. J.T. Schwartz and C.K. Yap, Lawrence Erlbaum Associated Pub., Hillsdale, New Jersey, pp. 95-144.

NEURAL NETWORKS

"Computational" Neural Learning Formalisms for Manipulator Inverse Kinematics

Sandeep Gulati Jacob Barhen S. Sitharama Iyengar[§]

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

ABSTRACT

An efficient, adaptive neural learning paradigm for addressing the inverse kinematics of redundant manipulators is presented. The proposed methodology exploits the infinite local stability of terminal attractors - a new class of mathematical constructs which provide unique information processing capabilities to artificial neural systems. For robotic applications, synaptic elements of such networks can rapidly acquire the kinematic invariances embedded within the presented samples. Subsequently, joint-space configurations, required to follow arbitrary end-effector trajectories, can readily be computed. In a significant departure from prior neuromorphic learning algorithms, this methodology provides mechanisms for incorporating an in-training "skew" to handle kinematics and environmental constraints.

1. INTRODUCTION

Space telerobots envisioned for exacting scientific and military applications in unstructured and hazardous space environments, e.g., satellite servicing, space system construction and maintenance, planetary missions etc., must be able to dexterously and adaptively manipulate objects in a nonstationary task workspace. Redundancy in the design of robot manipulators has been suggested as one means to enhance their dexterity and adaptability. In contradistinction to other engineering contexts where redundancy implies fault-tolerance or superfluity, redundancy in robotics is determined relative to the task [4]. It refers to a manipulator with more than the minimum number of degrees of freedom necessary to accomplish general tasks. The major objective motivating introduction of redundancy in robot design and control is to use the additional degrees of freedom to improve performance in complex and unstructured environments. It helps overcome kinematic, mechanical and other design limitations of non-redundant manipulators. Also, the extra degrees of freedom can be used during real-time manipulator operation to simultaneously achieve end-effector trajectory control while satisfying additional constraints.

There are two primary goals in developing control strategies which take advantage of redundancy. First, given the initial and final end-effector task coordinates, simultaneously generate, in real time, a Cartesian-space trajectory that enables the robot to achieve a goal (*the path planning problem*) and a set of joint space configurations, which cause the end-effector to follow the desired trajectory *(inverse kinematics problem)* while satisfying additional constraints, such as obstacle avoidance, servo-motor torque minimization and joint limit avoidance. Developing algorithms to use the additional degrees of freedom to satisfy constraints is known as the *redundancy resolution problem* [1,4,7,16]. Secondly, provide adaptive mechanisms for responding to any unforeseen changes in the workspace or the manipulator geometry. Despite a tremendous growth in research activity on "model-based" adaptive control algorithms, the above problems entail a level of computational and paradigmatic complexity far exceeding that which can be provided by the existing strategies.

Artificial neural networks on the other hand, defined as massively parallel, adaptive dynamical systems modeled on the general features of biological networks, interact with the objects of the real world and its

[§] Robotics Research Lab., Dept. of Computer Science, Louisiana State University, Baton Rouge, LA 70803.

statistical characteristics in the same way as living beings do. The potential advantages of neuronal processing arise as a result of their ability to perform parallel, asynchronous and distributed information processing in a dynamic self-organizing manner typical of living systems. Neurons with simple properties and interacting according to relatively simple rules can accomplish collectively complex functions such as generalization, error correction, pattern classification, learning etc. However, their paradigmatic strength for potential applications, which require solving intractable computational problems or adaptive modeling, arises from a spontaneous emergent ability to achieve *functional synthesis* and thereby learn topological mappings [8] i.e., establish relationships between multiple continuous-valued inputs and outputs, based on a presentation of a large number of examples. Once the underlying invariances have been learned and encoded in the strengths of the synaptic interconnections, the neural network can generalize to solve arbitrary problem instances. In addition, the operational versions of these trained networks can be dynamically "regularized" at run-time to satisfy one or more task-specific constraints, without any explicit retraining or reprogramming. Since the inverse mappings are acquired from real-world examples, network functionality is not limited by assumptions regarding parametric or environmental uncertainty [3]. Thus, neural networks provide an attractive alternate algorithmic basis towards designing real-time manipulator control architectures for automating "man-out-of-the-telerobot-loop" tasks beyond the existing technology. In this paper we introduce a powerful neural learning methodology for addressing the inverse kinematics problem commonly encountered during the design of real-time, adaptive systems operating in redundant environments.

2. MANIPULATOR INVERSE KINEMATICS

A forward kinematic function, Φ, is defined as a nonlinear differentiable function which uniquely relates a set of N_Q joint variables, $\bar{q}$, to a set of N_X task-space coordinates, $\bar{x}$: $\bar{x} = \Phi(\bar{q})$. For serial chain robot manipulators the forward kinematic function is easily derived [11]. The more difficult problem, which is of primary practical interest in robot manipulation, is the inverse transformation: $\bar{q} = \Phi^{-1}(\bar{x})$. In other words, determine one or more sets of joint configurations which take the end-effector into a desired task position and orientation in the operational workspace. While the inverse kinematic function is highly nonlinear, closed-form analytical solutions can be found for a number of non-redundant manipulators with special architecture. Complete positioning capability in Cartesian space can be nominally achieved by using only six degrees of freedom. However, most manipulators have degenerate configurations, or kinematic singularities, near which small displacements of the end-effector require physically unrealizable joint speeds. These singularities effectively lead to a loss of usable workspace and capability, and there is a strong incentive to design robots with additional degrees of freedom to overcome this and other problems. However, incorporation of redundancy injects additional complexity into the inverse kinematic problem. For redundant manipulators, the inverse kinematics problem has an infinity of solutions, which can be mapped into a finite set of manifolds [4].

Because of this infinity of solutions, many redundant manipulator investigators have chosen to focus on the instantaneous or differential kinematics [15], in which the instantaneous end-effector velocity is related to the instantaneous joint velocities by the manipulator Jacobian matrix. For redundant robots the manipulator Jacobian is not uniquely invertible, and pseudo-inverse techniques can be used to select a solution from the infinity of possible solutions. But pseudo-inverse resolution techniques are generally not-cyclic, i.e., do not generate closed joint-space trajectories corresponding to closed end-effector trajectories, thereby posing a serious limitation for practical implementations. So, in the absence of satisfactory closed-form solutions, offline iterative approximation techniques based on "local-methods" have been used, e.g., "augmented task method" proposed by Goldenberg et al. [5]. The latter however suffers from algorithmic singularities and is computationally prohibitive for manipulators with large degrees of freedom. In addition, the existing algebraic and geometric strategies provide limited mechanisms for resolution of kinematic redundancy with respect to multiple criteria [3] and have little susceptibility to unforeseen changes in the workspace or the manipulator geometry, etc. Since no mechanisms are provided for resolving redundancy over more than one internal self-motion manifold, each different application requirement may entail reprogramming the control algorithm, thereby severely limiting manipulator functionality.

In contrast, neuromorphic approaches to the inverse kinematics problem entail systems composed of many simple processors ("neurons"), fully or sparsely interconnected, whose functions are determined by the topology and strength of the interconnections. The synaptic elements of such neural systems must capture the transcendental kinematic transformations by using *a priori* generated examples enabling subsequent generalization to other points in the workspace. Thus, the inverse transformation equations do not need to be explicitly programmed or derived. Once they have been learned, the network's inherent self-organizing abilities enable it to adapt to changes in the environment, e.g. planning joint trajectories in the presence of obstacles or to any unforeseen changes in the mechanical structure of the manipulator, with little effort [8]. Within a neuromorphic framework, a solution of the inverse kinematic problem involves two phases: a training phase and a recall phase. The training phase involves encoding the inverse mapping in the network's synaptic weight space, through repeated presentations of a finite set of *a priori* generated examples, linking Cartesian space end-effector coordinates to the corresponding joint angles. Once the network has acquired the nonlinear mapping imbedded within the training set, it can be used to rapidly recall, or generalize the joint configuration corresponding to any arbitrary Cartesian-space orientation within its workspace of training, thereby eliminating the intensive computational overheads that plague the existing iterative techniques. Also, once the training cycle is completed, the time required to obtain a solution practically depends in a weak fashion on the number of degrees of freedom. In the past, Josin [8], Guez et al. [6] and Tawel et al. [14] have applied this generic neuromorphic paradigm to the inverse kinematics problem for a 3-DOF redundant manipulator. In particular, they train a heteroassociative, multi-layered feed-forward neural network by using the backpropagation algorithm (for details refer to [13]).

Despite its conceptual simplicity, there are a number of non-trivial issues, both from the kinematics perspective and from the computational cost perspective that have hitherto limited the efficacy of such neuromorphic solutions to the inverse kinematics problem for redundant motion control. The major limitations, as discerned from the existing implementations, include an unacceptably large number of training iterations ($O(10^6)$ even for generalizing over small manifolds, see Tawel et al. [14]). Also the interpolated angular coordinates have relatively poor precision as compared to their algebraic or iterative counterparts. Besides, the backpropagation algorithm fails to efficiently scale-up to configurations with large number of degrees of freedom. For example, manipulators with seven or more degrees of freedom could not be satisfactorily trained by use of the standard back-propagation algorithm even after several million iterations. Furthermore, the back-propagation algorithm *per se* does not provide any intrinsic mechanism to simultaneously exploit redundancy to increase the task workspace (design constraints) and satisfy additional requirements inherent to operations in an unstructured environment such as obstacle avoidance in real-time. Since the latter flexibility is essential to the purpose of redundant manipulators [16], there is a strong incentive to develop an alternative neural network methodology that alleviates the above limitations to provide an efficient and accurate solution to the inverse kinematics problem.

3. NEURODYNAMICAL FORMULATION

3.1. Training Network Specification

Consider a fully connected neural network with N graded-response neurons, implementing a nonlinear functional mapping from the N_X-dimensional input space to the N_Q-dimensional output space. The network is topographically partitioned into three mutually exclusive regions comprising a set of input neurons, S_X, that receives the input coordinates, an output set, S_Q, which provides the output coordinates required to achieve the desired output, and a set of "hidden" neurons, S_H, that partially encode the input / output mapping. The network is presented with K randomly sampled training pairs of input-output, $\{ \bar{x}^k, \bar{q}^k \mid k = 1, \ldots, K\}$ obtained by solving the well-posed forward kinematics formulation (see Paul [11]).

The neuromorphic reformulation of the inverse kinematics problem requires determining synaptic interconnection strengths that can accurately capture the transcendental transformations imbedded within the training samples. Our approach is based upon the minimization of a constrained Hamiltonian ("neuromorphic energy"), given by the following expression:

$$E = \frac{1}{2K} \sum_{k=1}^{K} \left(\frac{1}{N_X} \sum_{l \in S_X} [\, u_l^k - x_l^k \,]^2 + \frac{1}{N_Q} \sum_{l \in S_Q} [\, u_l^k - q_l^k \,]^2 \right) + \sum_r \lambda_r \, g_r(\cdot) \qquad (3.1.1)$$

where u_l^k denotes the l−th neuron's activity when processing the k−th training sample, $g_r(\cdot)$ reflects network constraints and the design considerations related to specific applications, e.g., singularity avoidance [4], obstacle avoidance [10], joint availability etc., and λ_r denotes the Lagrangian multiplier corresponding to the r−th application of design requirement. The proposed objective function therefore includes contributions from two sources. Firstly, it enforces convergence of every neuron in S_X and S_Q to attractors corresponding to the presented input-output coordinates and joint coordinates respectively for every sample pair in the training set, thereby enforcing the network to learn the underlying kinematic invariances. Secondly, it enforces the synaptic elements to satisfy network constraints of the type

$$g_r(\cdot) = \frac{1}{2} (i - j)^2 \, T_{ij}^2$$

which minimize the interconnection strengths in line with the Gauss's least-constraint principle. For a more detailed treatment of redundancy resolution refer to [2,3]. We now proceed with the derivation of the learning equations (time evolution of the synaptic weights) by minimizing the energy function given in eqn. (3.1.1).

Lyapunov's stability criteria require an energy function to be monotonically decreasing in time. Since in our model the internal dynamical parameters of interest are the synaptic interconnection strengths, T_{nm}, and the Lagrange multipliers λ_r, this implies that

$$\dot{E} = \sum_n \sum_m \frac{\partial E}{\partial T_{nm}} \dot{T}_{nm} + \sum_r \frac{\partial E}{\partial \lambda_r} \dot{\lambda}_r < 0 \qquad (3.1.2)$$

One can choose

$$\tau_T \, \dot{T}_{nm} = - \frac{\partial E}{\partial T_{nm}} \qquad (3.1.3)$$

where τ_T is an arbitrary but positive time-scale parameter. Then substituting in Eqs. (3.1.2) we have

$$\sum_r \frac{\partial E}{\partial \lambda_r} \dot{\lambda}_r < \tau_T \, \dot{T} \oplus \dot{T}. \qquad (3.1.4)$$

In the above expression $\oplus$ denotes tensor contraction, i.e., $\dot{T} \oplus \dot{T} \equiv \sum_i \sum_j \dot{T}_{ij} \, \dot{T}_{ij}$. This will be true *a fortiori* if for some $\theta > 0$,

$$\sum_r \dot{\lambda}_r \frac{\partial E}{\partial \lambda_r} + \theta < \tau_T \dot{T} \oplus \dot{T} \qquad (3.1.5)$$

The equations of motion for the Lagrange multipliers λ_r must now be constructed in such a way that Eq. (3.1.4) is strictly satisfied. Noting that the analytic expression for the energy function results in $\frac{\partial E}{\partial \lambda_r} = g_r(\cdot)$, we adopt the following model:

$$\dot{\lambda}_r = \tau_T \frac{\dot{T} \oplus \dot{T} - \theta}{\bar{g} \oplus \bar{g} + \theta} \, g_r \qquad (3.1.6)$$

where $\bar{g} \oplus \bar{g} \equiv \sum_r g_r(\cdot) \, g_r(\cdot)$, and θ is an arbitrary positive constant. It is easy to see that $\dot{E} < 0$ is strictly satisfied. Also on differentiating Eqs. (3.1.1) with respect to T_{nm} we get

$$\frac{\partial E}{\partial T_{nm}} = \frac{1}{K} \sum_k \left\{ \frac{1}{N_X} \sum_{l \in S_X} [\, u_l^k - x_l^k \,] \frac{\partial u_l^k}{\partial T_{nm}} \right.$$
$$\left. + \frac{1}{N_Q} \sum_{l \in S_Q} [\, u_l^k - q_l^k + N_Q \sum_r \frac{\partial g_r}{\partial u_l^k} \,] \frac{\partial u_l^k}{\partial T_{nm}} \right\} \qquad (3.1.7)$$

If we define

$$\hat{I}_l^k = \begin{cases} \frac{1}{KN_Q}[\, u_l^k - q_l^k + N_Q \sum_r \frac{\partial q_r}{\partial u_l^k}] & \text{if } l \in S_Q \\ 0 & \text{if } l \in S_H \\ \frac{1}{KN_X}[\, u_l^k - x_l^k \,] & \text{if } l \in S_X \end{cases} \tag{3.1.8}$$

then we can rewrite Eqn. (3.1.7) as

$$\frac{\partial E}{\partial T_{nm}} = \sum_l \sum_k \hat{I}_l^k \frac{\partial u_l^k}{\partial T_{nm}} \tag{3.1.9}$$

where the index l is defined over the entire set of neurons. Eqs. [3.1.3, 3.1.8 and 3.1.9] constitute a dissipative nonlinear dynamical system, the flow of which generally converges to a manifold of lower dimensionality in the phase space. In this paper we focus on network convergence to point attractors, i.e., state-space vector locations corresponding to the presented, joint- and Cartesian-space coordinates. Of crucial importance is to know how stable these attractors are, and, starting from arbitrary network configurations how fast they can be reached. In this vein, we first briefly review a novel mathematical concept in dynamical systems theory, the *terminal attractor*, and its properties that subsequently will enable us to formalize efficient algorithms for learning the inverse kinematics mapping.

Artificial neural networks store memory states or patterns in terms of the fixed points of the network dynamics, such that initial configurations of neurons in some neighborhood, or *basin of attraction*, of that memory state will be attracted to it [9]. But the static attractors considered so far in nonlinear dynamical system formulations in general, and in neural networks in particular, have represented regular solutions to the differential equations of motion. Such solutions can never intersect the transients. The theoretical relaxation time of the system to these "regular attractors" can theoretically be infinite, and they suffer from convergence to spurious states and local minima. The concept of *terminal attractors* in dynamical systems, was initially introduced by Zak [17,18] to obviate some of the above limitations, thereby significantly improving the performance characteristics of associative memory neural network models.

The existence of terminal attractors was established by Zak [17,18] using the following argument: At equilibrium, the fixed points, $\bar{p}$, of an N-dimensional, dissipative dynamical system

$$\dot{u}_i - f_i(u_1, u_2, , \cdots, u_N) = 0 \quad i = 1, 2, \cdots, N \tag{3.1.10}$$

are defined as its constant solutions $\bar{u}^{\infty}(\bar{p})$. If the real parts of the eigenvalues, η_μ, of the matrix $M_{ij} = \left[\frac{\partial f_i}{\partial x_j}(\bar{p})\right]$ are all negative, i.e., Re $\{\eta_\mu\} < 0$, then these points are globally asymptotically stable. Such points are called static attractors since each motion along the phase curve that gets close enough to $\bar{p}$, i.e., enters a so-called basin of attraction, approaches the corresponding constant value as a limit as $t \rightarrow \infty$. An equilibrium point represents a repeller if at least one of the eigenvalues of the matrix $\mathbf{M}$ has a positive real part. Usually, nonlinear neural networks deal only with systems which satisfy the Lipschitz condition, i.e., $\left| \frac{\partial f_i}{\partial u_j} \right| < \infty$. This condition guarantees the existence of a unique solution for each of the initial phase space configurations. That is why a transient solution cannot intersect the corresponding constant solution to which it tends, and therefore the theoretical time of approaching the attractors is always infinite.

In contrast, Zak's notion of terminal attractors is based upon the violation of the Lipschitz condition. As a result of this violation the fixed point becomes a singular solution which envelops the family of regular solutions, while each regular solution approaches the terminal attractor in finite time. To formally exhibit a terminal attractor which is approached by transients in finite time, consider the following one-dimensional example:

$$\dot{u} = -u^{1/3} \tag{3.1.11}$$

This equation has an equilibrium point at $u = 0$ at which the Lipschitz uniqueness condition is violated, since

$$\frac{d\dot{u}}{du} = -\frac{1}{3}u^{-2/3} \longrightarrow -\infty \;\; at \;\; u \longrightarrow 0 \tag{3.1.12}$$

Since here $\mathrm{Re}\{\eta\} \rightarrow -\infty < 0$, this point is an attractor with "infinite" local stability. As a consequence, the dynamical system is bestowed with "infinite attraction power", enabling rapid clamping of neuronal potentials to the fixed points; in this case the above phenomena imply immediate relaxation to the desired attractor coordinates, x_l and q_l. Also the relaxation time for the solution corresponding to initial conditions $u = u_0$ of this attractor is finite. It is given by

$$t_0 = - \int_{u_0}^{u \to 0} \frac{du}{u^{1/3}} = \frac{3}{2} u_0^{2/3} < \infty \tag{3.1.13}$$

i.e., this attractor becomes terminal. It represents a singular solution which is intersected by all the attracted transients. In particular static terminal attractors occur for $k = (2n+1)^{-1}$ and $n \geq 1$, while for $k = 2n+1$ all attractors are regular. It has been shown that incorporation of terminal attractor dynamics leads to the elimination of all spurious states. This property is critical to providing an accurate generalization ability during the operational phase. It ensures that interpolations / extrapolations of joint configurations are not based on false attractors, i.e., attractor coordinates not obtainable by the forward kinematics mapping. In our proposed neuromorphic framework, terminal attractor dynamics then provides a mechanism that can implicitly exploit the time-bounded terminality of phase trajectories and the locally infinite stability, thereby enabling an efficient and accurate solution to the manipulator inverse kinematics.

3.2. Virtual Attractor Computation

The Hamiltonian defined in Eqs. (3.1.1) specified the functionality of our fully connected neural network, i.e., learn the inverse kinematics mapping. We now need to select the network dynamics for evolving the synaptic elements, such that the latter's convergence to steady state leads towards the above function. So to capture the kinematic invariances consider the following neurodynamics.

$$\tau_u \dot{u}_l^k + u_l^k = \varphi_\gamma \, [\sum_{l'} T_{ll'} u_{l'}^k \,] - I_l^k \tag{3.2.1}$$

Here u_l represents the mean soma potential of the l-th neuron (u_l^k is the neuron's activity when processing the k-th training sample), $T_{ll'}$ denotes the synaptic coupling from the l'-th to the l-th neuron, and I_l^k captures the varying input/output contribution in a terminal attractor formalism. Though I_l^k influences the degree of stability of the system and the convergence to fixed points in finite time, it does not further affect the location of existing static attractors. And, $\varphi_\gamma(\cdot)$ denotes the sigmoidal neural response function with gain γ; typically, $\varphi_\gamma(z) = \tanh(\gamma \cdot z)$. In topographic maps N_T neurons are generally used to compute a single value of interest in terms of spatially-coded response strengths. Here we use the simplest possible model (where $N_T = 1$), but encode the information through terminal attractors. Thus, the topographic map is given by

$$I_l^k = \begin{cases} (\ u_l^k - x_l^k \ )^{1/3} & \text{if } l \in S_X \\ 0 & \text{if } l \in S_H \\ (\ u_l^k - q_l^k \ )^{1/3} & \text{if } l \in S_Q \end{cases} \tag{3.2.2}$$

where x_l^k and q_l^k are the attractor coordinates provided by the training sample, to be denoted for brevity as a_l^k. Our basic operating assumption for the dynamical system defined by (3.2.1) is that at equilibrium

$$\dot{u}_n \longrightarrow 0 \quad \text{and} \quad u_n \longrightarrow a_n$$

This yields the fixed point equations :

$$a_n = \varphi_\gamma \left[\sum_m T_{nm} \, a_m \right] \tag{3.2.3}$$

In associative memory applications, these equations can in principle be used to determine the synaptic coupling matrix T, resulting in each memory pattern being stored as a fixed point. The key issue is that some of these fixed points may actually be repellers. The terminal attractors are thus used to guarantee that each fixed point becomes an attractor, i.e., spurious states are suppressed. In this case however, we are in the process of learning a continuous mapping between two spaces and attractor coordinates have been defined for only two of the three topographic regions of the network, i.e., the input set S_X, and the output set S_Q. Consequently, the fixed point equation $\bar{a} = \varphi(T\bar{a})$ may not necessarily be defined, since for $| S_H | > 0$, $\{ a_n \mid n \in S_H \}$ are not defined, and cannot be used for directly computing T.

This necessitates a strategy whereby ***virtual attractor*** coordinates are first determined for the hidden units. These coordinates are virtual since they correspond to a current estimate $\hat{T}$ of the synaptic connectivity matrix. This is achieved by considering the fixed point equations as ***adaptive conservation equations*** which use the extra degrees of freedom made available by the hidden neurons in S_H. Let $\{ \hat{u}_j = a_j \mid j \in S_H \}$ denote the virtual attractors to which the unknowns, $\{ u_j \mid j \in S_H \}$, are expected to converge. Then at equilibrium Eqn. (3.2.3) yields

$$\begin{aligned}
\varphi^{-1}(x_i) &= \sum_{i' \in S_X} \hat{T}_{ii'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{ij'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{il'} q_{l'} \qquad \forall i \in S_X \\
\varphi^{-1}(\hat{u}_j) &= \sum_{i' \in S_X} \hat{T}_{ji'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{jj'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{jl'} q_{l'} \qquad \forall j \in S_H \\
\varphi^{-1}(q_l) &= \sum_{i' \in S_X} \hat{T}_{li'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{lj'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{ll'} q_{l'} \qquad \forall l \in S_Q
\end{aligned} \tag{3.2.4}$$

where $\hat{T}_{jl}$ denotes the current estimate of synaptic coupling from l-th neuron to the j-th neuron, and $\hat{u}_j$ represents a virtual attractor whose value is isomorphic to the current level of knowledge in the network. Now define

$$\begin{aligned}
\psi_i &= \varphi^{-1}(x_i) - \sum_{i'} \hat{T}_{ii'} x_{i'} - \sum_{l'} \hat{T}_{il'} q_{l'} \qquad \forall\, i \in S_X \\
\psi_j &= \sum_{i'} \hat{T}_{ji'} x_{i'} + \sum_{l'} \hat{T}_{jl'} q_{l'} \qquad \forall\, j \in S_H \\
\psi_l &= \varphi^{-1}(x_l) - \sum_{i'} \hat{T}_{li'} x_{i'} - \sum_{l'} \hat{T}_{ll'} q_{l'} \qquad \forall\, l \in S_Q.
\end{aligned} \tag{3.2.5}$$

Then consistency with the terminal attractor dynamics assumptions requires that $\{ \hat{u}_j \mid j \in S_H \}$ be simultaneous solutions to the following "conservation" equations

$$\begin{aligned}
\sum_{j' \in S_H} \hat{T}_{ij'} \hat{u}_{j'} &= \psi_i \qquad \forall\, i \in S_X \\
\varphi^{-1}(\hat{u}_j) - \sum_{j' \in S_H} \hat{T}_{jj'} \hat{u}_{j'} &= \psi_j \qquad \forall\, j \in S_H \\
\sum_{j' \in S_H} \hat{T}_{lj'} \hat{u}_{j'} &= \psi_l \qquad \forall\, l \in S_Q
\end{aligned} \tag{3.2.6}$$

The above system of equations for $\hat{u}$ is generally overdetermined. A number of standard algorithms exist to obtain a good approximate solution to such a system. In our implementation we use an iterative approach (e.g. conjugate gradient descent) to minimize the function

$$\begin{aligned}
\hat{E} &= \frac{1}{2N_X} \sum_i \Big(\psi_i - \sum_{j'} \hat{T}_{ij'} \hat{u}_{j'} \Big)^2 + \frac{1}{2N_H} \sum_j \Big(\hat{u}_j - \varphi[\sum_{j'} \hat{T}_{jj'} \hat{u}_{j'} + \psi_j] \Big)^2 \\
&+ \frac{1}{2N_Q} \sum_l \Big(\psi_l - \sum_{j'} \hat{T}_{lj'} \hat{u}_{j'} \Big)^2
\end{aligned} \tag{3.2.7}$$

We can now return to the computation of $\partial u_l^k \;/\; \partial T_{nm}$ in Eq. (3.1.9). Let us define $z_l^k \;=\; \sum_{l'} T_{ll'}\; u_{l'}$ and denote $\varphi'_{lk} \;=\; \frac{\partial \varphi(\cdot)}{\partial z_l^k}$. Then at equilibrium, as $\dot{u}_l^k \longrightarrow 0$ and $I_l^k \longrightarrow 0$, we have

$$\frac{\partial u_l^k}{\partial T_{nm}} \;=\; \varphi'_{lk} \left[\sum_{l'} \frac{\partial T_{ll'}}{\partial T_{nm}}\, u_{l'}^k \;+\; \sum_{l'} T_{ll'}\, \frac{\partial u_{l'}^k}{\partial T_{nm}} \right] \tag{3.2.8}$$

which can be rewritten as

$$\sum_{l'} [\; \delta_{ll'} \;-\; \varphi'_{lk}\; T_{ll'} \;]\; \frac{\partial u_{l'}^k}{\partial T_{nm}} \;=\; \varphi'_{lk} \delta_{ln} u_m^k \tag{3.2.9}$$

In the above expression δ_{ij} denotes the Kronecker symbol. We now define, following [12], a weighted coupling matrix $A_{ll'}^k \;=\; \delta_{ll'} \;-\; \varphi'_{lk}\; T_{ll'}$. Then substituting $A_{ll'}^k$ in (3.2.9), and premultiplying both sides with $[A^{-1}]_{ln}^k$ and summing over l yields

$$\sum_l [A^{-1}]_{pl}^k\; A_{ll'}^k\; \frac{\partial u_{l'}^k}{\partial T_{nm}} \;=\; \sum_l [A^{-1}]_{pl}^k\; \varphi'_{lk}\; \delta_{ln}\; u_m^k. \tag{3.2.10}$$

Carrying out the algebra and relabeling the dummy indices results in

$$\frac{\partial u_l^k}{\partial T_{nm}} \;=\; [A^{-1}]_{ln}^k\; \varphi'_{nk}\; u_m^k. \tag{3.2.11}$$

The above expression can now be substituted in Eq. (3.2.10); the learning equation thus takes the form

$$\tau_T \dot{T}_{nm} \;=\; -\sum_l \sum_k \hat{I}_l^k\; [\; A^{-1}\;]_{ln}^k\; \varphi'_{nk}\; u_m^k \tag{3.2.12}$$

where the indices l and k run over the complete sets of neurons and training samples.

3.3. Adjoint Network Dynamics

A computation of the synaptic interconnection matrix as suggested by Eq. (3.2.12) would involve a matrix inversion. Since direct matrix inversion is typically nonlocal, we adopt the relaxation procedure suggested by Pineda [12] to compute the synaptic updates defined by (3.2.12). Consider the following change of variable

$$v_n^k \;=\; \sum_l [\; A^{-1}\;]_{ln}^k\; \hat{I}_l^k\; \varphi'_{nk} \tag{3.3.1}$$

Then substituting (3.3.1) in (3.2.12) we have

$$\sum_n A_{np}^k\; \frac{v_n^k}{\varphi'_{nk}} \;=\; \hat{I}_p^k \tag{3.3.2}$$

One can also use the explicit form of A_{np}^k and by substitution in (3.2.12), we obtain

$$\sum_n A_{np}^k\; \frac{v_n^k}{\varphi'_{nk}} \;=\; \frac{v_p^k}{\varphi'_{pk}} \;-\; \sum_n T_{np}\; v_n^k \tag{3.3.3}$$

Regrouping the previous equations and (3.3.2), and relabeling the dummy indices yields

$$v_n^k \;=\; \varphi'_{nk} \;\cdot\; [\; \sum_p T_{pn} v_p^k \;+\; \hat{I}_n^k \;]. \tag{3.3.4}$$

where τ_v denotes the relaxation constant. We see that v_n^k represents a fixed point solution of an "adjoint" neural network having the following coupled dynamics:

$$\tau_v \dot{v}_n^k + v_n^k = \varphi'_{nk} \cdot [\sum_p T_{pn} v_p^k + \hat{I}_n^k] \tag{3.3.5}$$

By comparing Eqs. (3.2.12, 3.3.1 and 3.3.5) we see that the resulting neural learning equations couple the terminal attractor dynamics for u_m^k with the adjoint dynamics for v_n^k, i.e.,

$$\tau_T \dot{T}_{nm} = - \sum_k v_n^k u_m^k \tag{3.3.6}$$

During run-time, i.e., after the kinematic invariances have been learned, the above neurodynamics can be used to generate joint-configurations corresponding to arbitrary task end-effector positions, with two primary modifications. Firstly, in the operational phase, terminal attractor coordinates are specified only on the input neurons. Secondly, adaptive virtual attractor computation is no longer required. The pseudo-code for the complete neural learning algorithm, criteria for selecting different time-scales and the results of our simulation on 3-dof and 7-dof redundant manipulators are reported in [3,19].

4. CONCLUSIONS

In this paper we have attempted to address a complex problem in robotics research, which enables the enhancement of manipulative capability and reliability. Our novel learning paradigm for neural network models, based on the terminal attractor concept, is shown [19] to be computationally competitive with iterative methods currently used in robotics to solve the inverse kinematics of redundant manipulators. In addition, this strategy does not appear to suffer from non-cyclicity of motion, as encountered in the pseudo-inverse resolution techniques, or the algorithmic singularities common to augmented task approaches. Furthermore, unlike the feed forward, backpropagation neural learning approaches, the adaptive dynamical system formulation presented here provides the flexibility for incorporating arbitrary combinations of kinematic optimization criteria, without imposing high computational overheads. Two options are available for including the redundancy resolution criteria in the algorithm to resolve the nonuniqueness of joint configurations that may satisfy a given end-effector configuration. The constraints may either be included *a priori* , i.e., while generating the training samples themselves, thereby forcing the network to learn only limited aspects of inverse kinematics mapping with a bias towards a particular criterion [2,3]; or they could be selectively applied in real-time to an operational version of the network (trained to encode the emergent invariants of the inverse kinematic mapping), to regularize the solutions (i.e. provide unique best answers) [3].

Despite the emphasis on real-time performance, the dexterous nature of applications envisaged for the next-generation robots imposes uncompromising demands on the resultant end-effector trajectory. Consequently, this entails the generation of intermediate joint angles with a high degree of precision, currently achievable only through off-line programming techniques (e.g., acceptable error tolerances are below 0.05%). In this context, our future directions for research include development of true neural topographic map techniques, enabling the much higher resolution needed to achieve the desired precision in interpolated joint angles.

ACKNOWLEDGEMENTS

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The authors wish to acknowledge fruitful discussions with Michail Zak, Ken Kreutz and Joel Burdick.

REFERENCES

[1] J. Baillieul, "Kinematic Programming Alternatives for Redundant Manipulators ", in *Proc. IEEE Int'l Conf. on Robotics and Automation* , San Francisco, April 1986, pp. 1698-1704.

[2] J. Barhen, S. Gulati and M. Zak, "Neural Learning Algorithm for Inverse Kinematics of Redundant Manipulators in Unstructured Environments", *IEEE Computer* , Vol. 22, June 1989 (in press).

[3] J. Barhen and S. Gulati, "Neuromorphic Formalisms for Resolution of Kinematic Redundancy" submitted for publication.

[4] J.W. Burdick IV, " Kinematic Analysis and Design of Redundant Robot Manipulators," *Ph. D. Thesis,* Dept. of Mech. Eng., Stanford Univ., 1988.

[5] A.A. Goldenberg, B. Benhabib and R.G. Fenton, "A Complete Generalized Solution to the Inverse Kinematics of Robots," *IEEE Journal of Robotics and Automation* , Vol. RA-1, No. 1, March 1985, pp. 14-20.

[6] A. Guez, "Solution to the Inverse Kinematics Problem in Robotics by Neural Networks," in *Proc. of 2nd Int'l Conf. on Neural Networks* , San Diego, 1988, Vol. 2, pp. 617-624.

[7] J.M. Hollerbach and K.C. Suh, " Redundancy Resolution of Manipulators Through Torque Optimization," in *Proc. of IEEE Int'l Conf. on Robotics and Automation* , St. Louis, 1985, pp. 1016-1021.

[8] G. Josin, " Integrating Neural Networks with Robots ", *AI Expert* , Vol. 3, 8, 1988, pp. 50-60.

[9] J.D. Keeler, " Basins of Attraction of Neural Network Models ", *Proc. of AIP Conference* , Vol. 151, Neural Networks for Computing, Snowbird, UT, 1986, pp. 259-265.

[10] A.A. Maciejewski and C.A. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", *Int'l Journal on Robotics Research* , Vol. 4, No. 3 , 1985, pp. 109-117.

[11] R.P. Paul, *Robot Manipulators : Mathematics, Programming and Control* , Cambridge, Mass., MIT Press, 1981.

[12] F.J. Pineda, "Generalization of Back-Propagation to Recurrent Neural Networks," *Physical Review Letters* , Vol. 59, No. 19, 1987, pp. 2229-2232.

[13] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing* , Vol. I, Cambridge, Mass.: MIT/Bradford Book, 1986.

[14] R. Tawel, S. Eberhardt and A.P. Thakoor, "Neural Networks For Robotic Control," in *Proc. Conf. on Neural Networks for Computing* , Snowbird, Utah, 1988.

[15] D.E. Whitney, " Resolved Motion Rate Control of Manipulators and Human Prosthesis," *IEEE Trans. on Man-Machine Systems* , Vol. MMS-10, 1969, pp. 47-53.

[16] T. Yoshikawa, " Analysis and Control of Robot Manipulators with Redundancy ", *Robotics Research: First Int'l Symp.*, ed. M. Brady and R. Paul, Cambridge, Massachusetts: MIT Press, 1984, pp. 735-748.

[17] M. Zak, " Terminal Attractors for Addressable Memory in Neural Networks," *Physics Letters A*, Vol. 133, No. 1,2, 1988.

[18] M. Zak, " Terminal Attractors in Neural Networks", *Int'l Journal on Neural Networks*, Vol. 2, 1989 (in press).

[19] J. Barhen and S. Gulati, "Self-Organizing Neuromorphic Architecture for Manipulator Inverse Kinematics", *NATO-ASI Series*, Vol. F44, 1989 (in press).

N90-29031

MULTI-LAYER NEURAL NETWORKS FOR ROBOT CONTROL

Farzad Pourboghrat

Department of Electrical Engineering
Southern Illinois University
Carbondale, IL 62901-6603

Abstract

Two neural learning controller designs for manipulators are considered. The first design is based on a neural inverse-dynamics system. The second is the combination of the first one with a neural adaptive state feedback system. Both types of controllers enable the manipulator to perform any given task very well after a period of training, and to do other untrained tasks satisfactorily. The second design also enables the manipulator to compensate for unpredictable perturbations.

1. Introduction

The design of advanced control systems for robot manipulators has been a very active area of research in recent years. Inadequacy of current control strategies suggests that there is a need for a newer and faster control architecture which will account for both learning and control of robotic manipulators.

In classical systems theory, input-output descriptions are based on some assumed or predetermined mathematical structures, normally a set of linear differential equations. Replacement of these predetermined structures by learned associative memory mappings of stimulus-response leads to more general, normally non-linear, representations of the connections between inputs and outputs. This procedure can be implemented by neural networks [1]. The best example of a system with such an architecture is the human brain, which performs many complex functions superbly.

In the problem of motor control, obtaining an input function $u(t)$ to generate a desired motion $y(t)$ is directly related to finding the inverse-dynamics of the controlled system. Let the operator $\mathfrak{G}$ denote the dynamics relation of the system, where $\mathfrak{G}(u)=y$. Then the inverse-dynamics of the system is the operator $\mathfrak{L}=\mathfrak{G}^{-1}$ such that $\mathfrak{L}(y)=u$. Knowing the inverse-dynamics relation $\mathfrak{L}=\mathfrak{G}^{-1}$, for a given desired motion trajectory y_d, the required input u_d can be found from $u_d=\mathfrak{L}(y_d)$. This is because the motion corresponding to u_d is equal to $y=\mathfrak{G}(u_d)=\mathfrak{G}(\mathfrak{L}(y_d))=\mathfrak{G}(\mathfrak{G}^{-1}(y_d))=y_d$.

It has been shown that multi-layer neural networks with sigmoidal functions are able to map any measurable function to another with an arbitrary degree of accuracy, provided that there are enough units in their hidden layers. Therefore, such networks can be used for approximating the model of the inverse of the dynamics of a system [2-10]. In this paper the development of neuromorphic learning controllers is considered. First, a recurrent neural network learning controller $\mathfrak{C}$ is designed. The design has a neural inverse-dynamics block $\mathfrak{L}$ and a PD-type feedback block $\mathfrak{H}$. Next, the learning controller $\mathfrak{C}$ is modified, where its PD-type feedback block is replaced by a neural adaptive state feedback block $\mathfrak{H}$, which is to optimally compensate for unpredictable perturbations. The architectures of these learning controllers are similar to those in [10], which are inspired by the model of the cerebellum given by Kawato [5-6].

2. Robot Dynamics

The dynamics of a robot manipulator can be represented by an operator $\mathcal{G}$ which corresponds to a set of n coupled nonlinear differential equations, given by

$$M(q)\,q'' + N(q,q') + Q(q) = u \qquad (1a)$$

or $$\mathcal{G}(u) = q \qquad (1b)$$

where q, q', and q" are n-dimensional vectors of the positions, velocities, and accelerations of the joints, respectively, where "prime" denotes the time-derivative. M(q) is the nxn inertia matrix of the arm, which is symmetric and positive definite. N(q,q') is the n-dimensional vector of coriolis, centrifugal, and frictional forces. Q(q) is the n-dimensional vector representing the torques due to gravitational forces, and u is the n-dimensional vector of the generalized input torques applied to the robot.

3. Learning Controller Design

There are a variety of algorithms which can be used for multi-layer neural networks to learn the mapping between two patterns [1]. However, the state of the art learning algorithms are most effective when the input-output patterns are fixed. This condition, in general, is not satisfied when the objective error function is not identical to the error function at the neural network's output layer. To satisfy this condition we observe the following.

Lemma 1

Consider a stable system given by the operator $\mathcal{G}$, as in Figure 1, where its output q is desired to follow a reference function q_r. Let the high gain feedback block given by the linear operator $\mathcal{H}$ be such that the closed-loop $(I+\mathcal{GH})^{-1}\mathcal{G}$ is stable and that $\|\mathcal{GH}\| >> 1$. Then for bounded input v the output error $e=q_r-q$ is bounded and is given by $e=[(I+\mathcal{GH})^{-1}\mathcal{G}](\delta v)\approx\mathcal{H}^{-1}(\delta v)$, where $\delta v=r-v$. Moreover, the feedback signal $\delta u=\mathcal{H}(e)\approx\delta v$.

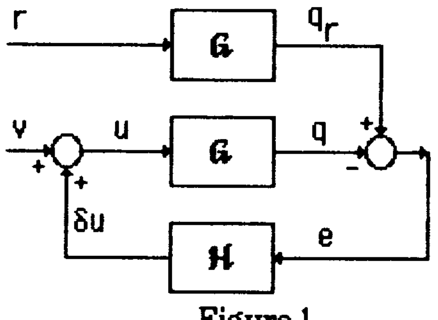


Figure 1

Proof

From Figure 1, by some block manipulation, it is easy to see that $e=[(I+\mathcal{GH})^{-1}\mathcal{G}](\delta v)$, where $\delta v=r-v$. Now let v be bounded. Then, since r exists, δv is also bounded. But since the closed-loop system $(I+\mathcal{GH})^{-1}\mathcal{G}$ is stable, the error signal $e=q_r-q$ is also bounded. Now since $\|\mathcal{GH}\| >> 1$, we get $e\approx\mathcal{H}^{-1}(\delta v)$. On the other hand, since $\mathcal{H}$ is linear, we have $\delta u=[(I+\mathcal{GH})^{-1}\mathcal{GH}](\delta v)$. But again, since $\|\mathcal{GH}\|>>1$, it is easy to see that $\delta u=\mathcal{H}(e)\approx\delta v$. □

4. Neural Inverse-Dynamics Model for Learning Control

The learning controller $\mathfrak{C}$, shown in Figure 2, has only one neural network block $\mathfrak{L}$ to approximate the inverse-dynamics model. There is also a feedback block $\mathfrak{H}$, of the PD-type, which is used for both the neural learning and the error compensation, and is given by

$$\mathfrak{H}(e) = \delta u = K_p e + K_d e' . \qquad (2)$$

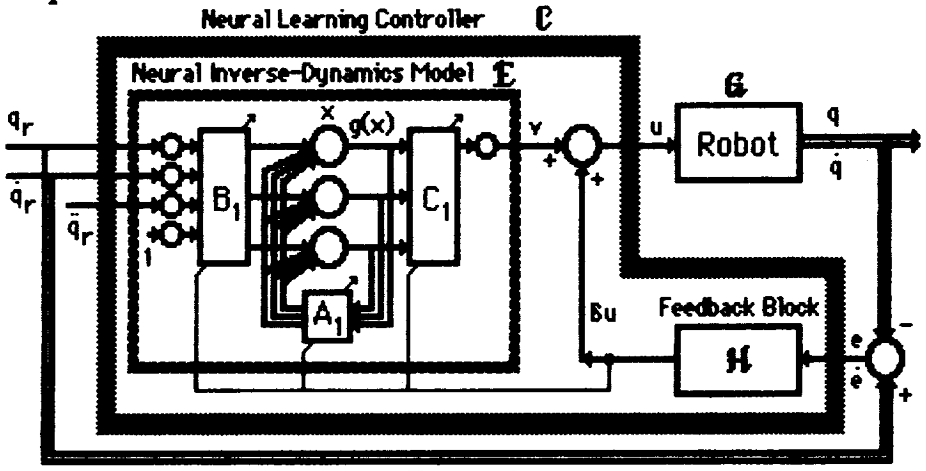


Figure 2

Network's Architecture

The neural block $\mathfrak{L}$ used here is essentially a recurrent multi-layer neural network. The input-output relation of the neural network $\mathfrak{L}$ is given by

$$\begin{aligned} x' &= A_1\, g(x) + B_1\, \theta \\ v &= C_1\, g(x) \end{aligned} \qquad (3)$$

where $\theta = [q_r^T, q'_r{}^T, q''_r{}^T, 1]^T \in \mathfrak{R}^{3n+1}$, $x \in \mathfrak{R}^N$, and $v \in \mathfrak{R}^n$ are respectively the vectors of the network's input, states, and outputs. A_1, B_1, and C_1 are respectively the matrices of the network's state recurrence, input, and output connection weights, and g is the sigmoidal function given by $g(x)=\tanh(x)$. The unity input in vector θ is added to allow for the automatic adjustment of the bias term.

Network's Learning Rule

The learning algorithm used for the network is a modification of the delta rule [1], and is given by [11]

$$\begin{aligned} a'_{1,ij} &= \alpha_1\ \delta u^T\ \nabla g(x)\ C_1\ \eta_{1,ij} \\ b'_{1,ik} &= \beta_1\ \delta u^T\ \nabla g(x)\ C_1\ \zeta_{1,ik} \\ c'_{1,pj} &= \gamma_1\ \delta u_p\, g(x_j) \\ \eta'_{1,ij} &= A_1 \nabla g(x)\ \eta_{1,ij} + I_i\, g(x_j) \\ \zeta'_{1,ik} &= A_1 \nabla g(x)\ \zeta_{1,ik} + I_i\, \theta_k \end{aligned} \qquad (4)$$

where I_i is the ith column of the identity matrix, δu is the feedback torque which is also the network's output error, $\nabla g(x)=\partial g(x)/\partial x$ is the Jacobian matrix, and α_1, β_1, and γ_1 are the learning rate constants which are small positive numbers. The initial values of matrices A_1, B_1, and C_1 are selected randomly between -0.2 and 0.2, and $\eta_{1,ij}(0)=\zeta_{1,ik}(0)=0$.

The objective of the learning controller $\mathfrak{C}$ is to force the system's output error to zero through repeated trials of the desired task. During trials, when the reference input is repeatedly applied to the system, the system's output error is used to adjust the controller parameters, which are the connection weights of the neural network block $\mathfrak{L}$. Therefore, the feedforward block $\mathfrak{L}$ is modified in such a way to force the feedback torque to vanish, which indirectly decreases the robot's output error. When the error becomes small, learning has been accomplished and the neural network block $\mathfrak{L}$ is said to have acquired the model of the inverse-dynamics of the robot. But for this, the corresponding learning algorithm must be convergent, or, the dynamics of the learning system must be asymptotically stable.

Result 1

Consider the robotic manipulator given by the operator $\mathfrak{G}$, as in equation (1). Let the neural learning controller $\mathfrak{C}$ given by equations (2) and (3) be applied to the system, as shown in Figure 2. Let the feedback block $\mathfrak{H}$ be such that the closed-loop system $(I+\mathfrak{G}\mathfrak{H})^{-1}\mathfrak{G}$ is stable and that $\|\mathfrak{G}\mathfrak{H}\| >> 1$. Then the neural learning controller $\mathfrak{C}$, together with the learning rule (4) is asymptotically stable. That is, the proposed learning controller forces the manipulator's trajectory q, q', to follow the desired trajectory q_r, q'_r, after a sufficiently long period of time.

Proof

Let $\eta_{1,ij}=\partial x/\partial a_{1,ij}$ and $\zeta_{1,ik}=\partial x/\partial b_{1,ik}$, then from equation (3), we get [11]

$$x = \int_0^t [A_1\, g(x) + B_1\, \theta]\, d\tau$$

$$\eta_{1,ij} = \int_0^t \partial/\partial a_{1,ij}\, [A_1 g(x)+B_1\, \theta]\, d\tau = \int_0^t [A_1 \nabla g(x)\eta_{1,ij}+I_i\, g(x_j)]\, d\tau$$

$$\zeta_{1,ik} = \int_0^t \partial/\partial b_{1,ik}\, [A_1 g(x)+B_1\, \theta]\, d\tau = \int_0^t [A_1 \nabla g(x)\zeta_{1,ik}+I_i\, \theta_k]\, d\tau.$$

Differentiating the above two relationships, we get

$$\eta'_{1,ij} = A_1 \nabla g(x)\, \eta_{1,ij} + I_i\, g(x_j) \tag{5}$$

$$\zeta'_{1,ik} = A_1 \nabla g(x)\, \zeta_{1,ik} + I_i\, \theta_k.$$

Now, without loss of generality, we asume that there exists an input function r(t) to the manipulator such that $q_r=\mathfrak{G}(r)$. Let a performance function for the learning process of the neural inverse-dynamics network be defined by

$$J_1(t) = 0.5\, [r(t)-v(t)]^T\, [r(t)-v(t)] = 0.5\, \delta r^2\ . \tag{6}$$

Since $J_1(t)$ is positive definite and monotonically increasing, for asymptotic stability, $J'_1(t)$

must be negative definite. But, the time derivative of $J_1(t)$ is given by

$$\begin{aligned} J'_1(t) &= \delta r^T \partial(\delta r)/\partial t \\ &= -\delta r^T [(\partial(\delta r)/\partial a_{1,ij})\, a'_{1,ij} + (\partial(\delta r)/\partial b_{1,ik})\, b'_{1,ik} + (\partial(\delta r)/\partial c_{1,pj})\, c'_{1,pj}] \\ &= -\delta r^T [C_1 \nabla g(x)\, \eta_{1,ij}\, a'_{1,ij} + C_1 \nabla g(x)\, \zeta_{1,ik}\, b'_{1,ik} + I_p\; g(x_j)\, c'_{1,pj}]. \end{aligned} \tag{7}$$

On the other hand, since $\|\mathfrak{G}\mathfrak{H}\| >> 1$, from Lemma 1 we have $\delta u = \delta r$. Therefore we have

$$J'_1(t) = -\delta u^T [C_1 \nabla g(x)\, \eta_{1,ij}\, a'_{1,ij} + C_1 \nabla g(x)\, \zeta_{1,ik}\, b'_{1,ik}] - \delta u_p\, g(x_j)\, c'_{1,pi}\,. \tag{8}$$

However, for $a'_{1,ij}$, $b'_{1,ik}$, and $c'_{1,pj}$ given by equation (4), we get

$$J'_1(t) = -\alpha_1 [\delta u^T C_1 \nabla g(x)\, \eta_{1,ij}]^2 - \beta_1 [\delta u^T C_1 \nabla g(x)\, \zeta_{1,ik}]^2 - \gamma_1 [\delta u_p\, g(x_j)]^2 \tag{9}$$

which is a negative definite scalar function, except when we have $\delta u=0$ where the learning is complete. Therefore, from the second method of Liapunov, the learning controller $\mathfrak{C}$ with the weight adjustments given by equation (4), is asymptotically stable (i.e., it is convergent). That is, the connection weight matrices A_1, B_1, and C_1 in the neural inverse-dynamics block $\mathfrak{E}$ will be adjusted until $J_1(t)=0$, that is when $\delta u=u-v=0$ or equivalently when $\delta r=r-v=0$. However, since the feedback operator is linear, $\delta u=K_p e+K_d e'=0$ implies that $e=e'=0$, since e and e' are linearly independent. Therefore, $q=q_r$, and $q'=q'_r$ as time t approaches infinity (i.e., the manipulator's trajectory q, q' follow the desired trajectory q_r, q'_r). □

The neural network $\mathfrak{E}$ part of the controller $\mathfrak{C}$ is able to acquire the model of the inverse-dynamics of the manipulator after a sufficiently long period of training. After this, the robot with the inverse-dynamics block $\mathfrak{E}$ alone (i.e., without the error feedback block $\mathfrak{H}$), is able to perform the trained tasks very well. In addition, the robot is able to perform some new tasks satisfactorily. However, without the feedback block $\mathfrak{H}$, the robot is not quite able to compensate for unpredictable perturbations. It is easily seen, however, that leaving block $\mathfrak{H}$ in the controller loop after the period of training greatly improves the ability of the controller to compensate for perturbations. This is the motivation for the next design.

5. Neural Adaptive State Feedback Model for Learning Control

The learning controller $\mathfrak{C}$ in this section contains both a feedforward and a feedback neural network block. The feedback neural block $\mathfrak{H}$ in this design has substituted for the PD-type error feedback block, as in Figure 3.

The neural adaptive state feedback block $\mathfrak{H}$ is intended as an optimal state feedback controller, and contains two sub-networks. One is the dynamics identifier $\mathfrak{D}$, which realizes the dynamics model of the system's perturbation about the nominal operating point. The other is the state feedback $\mathfrak{F}$, which generates an optimal state feedback for disturbance compensation. The overall feedback network $\mathfrak{H}$ learns to generate the optimal state feedback torques to eliminate perturbations. From the Linear Quadratic Control Theory, this network is equivalent to an optimal state feedback which continuously identifies the parameters of the perturbation dynamics of the manipulator, and from these, produces the optimum compensating torques.

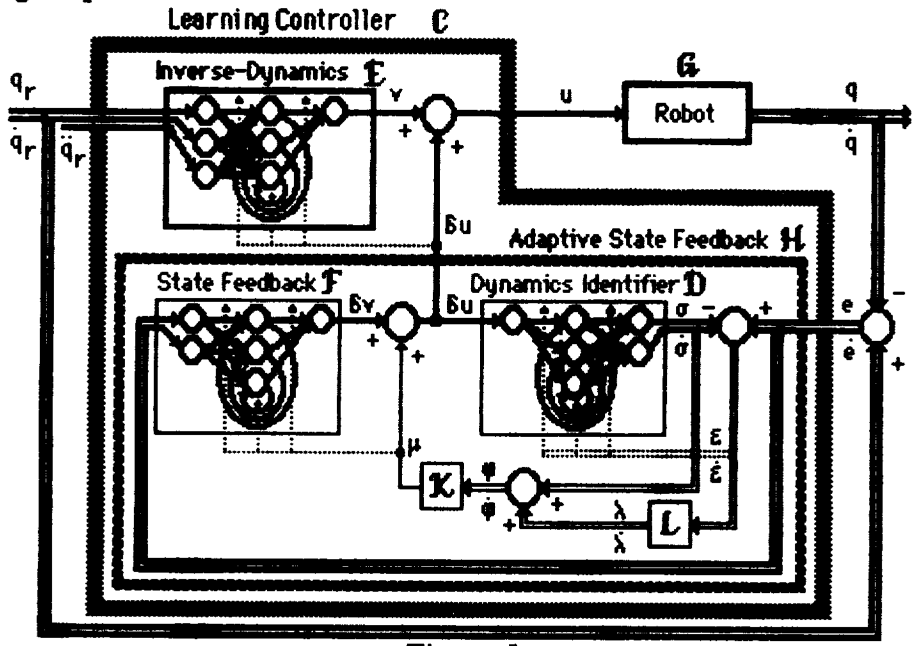


Figure 3

Feedback Network's Architecture

The input to neural block $\mathfrak{H}$ is a 2n-vector of the angular position velocity errors e and e'. The outputs of the network are the n compensating torque signals δu.

The input-output relationship for the dynamics identifier network $\mathfrak{D}$ is given by

$$y' = A_2\, g(y) + B_2\, \upsilon \qquad (10)$$
$$\underline{\sigma} = C_2\, g(y)$$

where $\upsilon = [\delta u^T, 1]^T \varepsilon\, \mathfrak{R}^{n+1}$, $y \,\varepsilon\, \mathfrak{R}^M$, $\underline{\sigma} = [\sigma^T, \sigma'^T]^T \varepsilon\, \mathfrak{R}^{2n}$ are respectively the input, state, and the output of the network. A_2, B_2, and C_2 are respectively the matrices of the network's state recurrence, input, and output connection weights.

The input-output relationship for the state feedback network $\mathfrak{F}$ is given by

$$z' = A_3\, g(z) + B_3\, \underline{e} \qquad (11)$$
$$\delta v = C_3\, g(z)$$

where $\underline{e} = [e^T, e'^T, 1]^T \varepsilon\, \mathfrak{R}^{2n+1}$, $z \,\varepsilon\, \mathfrak{R}^L$, $\delta v \,\varepsilon\, \mathfrak{R}^n$ are respectively the input, state, and the output of the network, and A_3, B_3, C_3 are respectively the matrices of the network's state recurrence, input, and output connection weights. As shown in Figure 3, there are some internal feedback blocks $\mathfrak{K}$ and $\mathfrak{L}$ within the neural adaptive state feedback block $\mathfrak{H}$ which are used primarily to provide a performance function for the networks' learning algorithm.

That is

$$\delta u = \delta v + \mu \qquad (12)$$
$$\mu = \mathfrak{K}(\underline{\varphi}) = K_p \varphi + K_d \varphi'$$
$$\underline{\varphi} = \underline{\sigma} + \underline{\lambda}$$
$$\underline{\lambda} = \mathfrak{L}(\underline{\varepsilon}) = L_p \varepsilon + L_d \varepsilon'$$

where $\underline{\varphi}=[\varphi^T,\varphi'^T]^T$, $\underline{\sigma}=[\sigma^T,\sigma'^T]^T$, $\underline{\lambda}=[\lambda^T,\lambda'^T]^T$, $\underline{\varepsilon}=[\varepsilon^T,\varepsilon'^T]^T=[(e-\sigma)^T,(e'-\sigma')^T]^T$, $\mathfrak{K}$ is a linear high gain feedback operator, and $\mathfrak{L}$ is a linear feedback gain block.

In the feedback block $\mathfrak{H}$, the neural dynamics identifier $\mathfrak{D}$ approximates the input-output relationship of the dynamics of perturbations by forcing its outputs to follow the system errors e and e'. The neural state feedback $\mathfrak{F}$, on the other hand, approximates the input-output relationship of an optimal state error feedback system by forcing its output to follow the input of the neural dynamics identifier $\mathfrak{D}$. This, in effect, adjusts block $\mathfrak{F}$ to approximate the inverse $\mathfrak{D}^{-1}$ of the neural dynamics identifier $\mathfrak{D}$.

Networks' Learning Rules

The learning algorithm used for the neural dynamics identifier network $\mathfrak{D}$ is similar to that of inverse-dynamics network $\mathfrak{E}$, i.e., the time derivative of the connection weight matrices A_2, B_2, and C_2 are given by [11]

$$a'_{2,ij} = \alpha_2 \underline{\varepsilon}^T \nabla g(y)\ C_2\ \eta_{2,ij} \qquad (13)$$
$$b'_{2,ik} = \beta_2 \underline{\varepsilon}^T \nabla g(y)\ C_2\ \zeta_{2,ik}$$
$$c'_{2,pj} = \gamma_2 \underline{\varepsilon}_p\, g(y_j)$$
$$\eta'_{2,ij} = A_2 \nabla g(y)\ \eta_{2,ij} + I_i\, g(y_j)$$
$$\zeta'_{2,ik} = A_2 \nabla g(y)\ \zeta_{2,ik} + I_i\, v_k$$

where $\underline{\varepsilon}=[\varepsilon^T,\varepsilon^T]^T$ is the network's output error, $\nabla g(y)=\partial g(y)/\partial y$ is the Jacobian matrix, and α_2, β_2, and γ_2 are small positive learning rate constants. The initial values of matrices A_2, B_2, and C_2 are selected randomly between -0.2 and 0.2, and $\eta_{2,ij}(0)=\zeta_{2,ik}(0)=0$.

The learning scheme for the neural state feedback network block $\mathfrak{F}$ is similar; i.e., the time derivative of the connection weight matrices A_3, B_3, C_3, are given by

$$a'_{3,ij} = \alpha_3 \mu^T \nabla g(z)\ C_3\ \eta_{3,ij} \qquad (14)$$
$$b'_{3,ik} = \beta_3 \mu^T \nabla g(z)\ C_3\ \zeta_{3,ik}$$
$$c'_{3,pj} = \gamma_3 \mu_p\, g(z_j)$$
$$\eta'_{3,ij} = A_3 \nabla g(z)\ \eta_{3,ij} + I_i\, g(z_j)$$
$$\zeta'_{3,ik} = A_3 \nabla g(z)\ \zeta_{3,ik} + I_i\, \underline{e}_k$$

where $\mu=\delta u-\delta v$ is the network's output error, $\nabla g(z)=\partial g(z)/\partial z$ is the Jacobian matrix, and α_3, β_3, and γ_3 are small positive learning rate constants. The initial values of matrices A_3, B_3, and C_3 are selected randomly between -0.2 and 0.2, and $\eta_{3,ij}(0)=\zeta_{3,ik}(0)=0$.

From Result 1, the inverse-dynamics neural network $\mathfrak{E}$ with its learning rule is able to realize the model of the inverse-dynamics $\mathfrak{G}^{-1}$ of the robot and to generate the required robot torque corresponding to the desired trajectory q_r and q'_r. From the Linear Quadratic Control Theory, in order to generate the compensating torque corresponding to the dynamics perturbations about the nominal trajectory of the robot, the adaptive state feedback neural network $\mathfrak{H}$ must identify the dynamics relation of the perturbations and correspondingly generate the optimal feedback according to some performance criterion. But for this the corresponding learning algorithm must be convergent (i.e., the dynamics of the learning system must be asymptotically stable).

Result 2

Consider the robotic manipulator given by the operator $\mathfrak{G}$, as in equation (1). Assume that the neural learning controller $\mathfrak{C}$, given by equations (3) and (10-12), is applied to the system, as shown in Figure 3. Let the feedback operator $\mathfrak{L}$ be a unity gain. Also let the high gain feedback block $\mathfrak{K}$ be such that the closed-loop system $(I+\mathfrak{G}\mathfrak{K})^{-1}\mathfrak{G}$ is stable and that $\|\mathfrak{G}\mathfrak{K}\| >>1$. Then the neural learning controller $\mathfrak{C}$, together with the learning rules (4) and (13-14) is asymptotically stable. That is, the learning controller $\mathfrak{C}$ forces the manipulator's trajectory q and q' to follow the desired trajectory q_r and q'_r after a sufficiently long time.

Proof

From Result 1, since $\mathfrak{L}$ is a unity gain, $\mathfrak{K}$ is such that $(I+\mathfrak{G}\mathfrak{K})^{-1}\mathfrak{G}$ is stable, and $\|\mathfrak{G}\mathfrak{K}\| >>1$, the learning process for the neural inverse-dynamics network $\mathfrak{E}$ is asymptotically stable.

Now let $\eta_{2,ij}=\partial y/\partial a_{2,ij}$, $\zeta_{2,ik}=\partial y/\partial b_{2,ik}$, $\eta_{3,ij}=\partial z/\partial a3,ij$, and $\zeta_{3,ij}=\partial z/\partial b3,ik$. Then, similar to the proof of Result 1, from the neural network's dynamics equations (10-11), we get

$$\eta'_{2,ij} = A_2\nabla g(y)\,\eta_{2,ij} + I_i\,g(y_j) \tag{15}$$
$$\zeta'_{2,ik} = A_2\nabla g(y)\,\zeta_{2,ik} + I_i\,\nu_k.$$
$$\eta'_{3,ij} = A_3\nabla g(z)\,\eta_{3,ij} + I_i\,g(z_j)$$
$$\zeta'_{3,ik} = A_3\nabla g(z)\,\zeta_{3,ik} + I_i\,\nu_k.$$

Now considering the convergence of the feedback block $\mathfrak{H}$, let a performance function for the learning process of the dynamics identifier sub-network $\mathfrak{D}$ be defined by

$$J_2(t) = 0.5\,\underline{\varepsilon}(t)^T\underline{\varepsilon}(t) \tag{16}$$

where $\underline{\varepsilon}=[\varepsilon^T,\varepsilon'^T]^T$ and $\varepsilon=e-\sigma$. Since $J_2(t)$ is positive definite and monotonically increasing, for asymptotic stability, $J'_2(t)$ must be negative definite. Using the chain rule, we get

$$J'_2(t) = \underline{\varepsilon}^T \partial(\underline{\varepsilon}')/\partial t \tag{17}$$
$$= -\underline{\varepsilon}^T [C_2 \nabla g(y)\, \eta_{2,ij}\, a'_{2,ij} + C_2 \nabla g(y)\, \zeta_{2,ik}\, b'_{2,ik} + I_p\, g(y_j)\, c'_{2,pj}].$$

However, for the weight adjustments given by equation (13), we have

$$J'_2(t) = -\alpha_2 [\, \underline{\varepsilon}^T C_2 \nabla g(y)\, \eta_{2,ij}]^2 - \beta_2 [\, \underline{\varepsilon}^T\ C_2 \nabla g(y)\, \zeta_{2,ik}]^2 - \gamma_2 [\, \underline{\varepsilon}_p\, g(y_j)]^2 \tag{18}$$

which is a negative definite scalar function, except when $\underline{\varepsilon}=0$ where learning is complete.

Similarly, let the learning performance function for neural block $\mathfrak{F}$ be defined by

$$J_3(t) = 0.5\, \zeta(t)^T\, \zeta(t) \tag{19}$$

where $\zeta = \delta r - \delta v$ is the output error of the network, $\delta r = r - v$, and r is such that $\mathfrak{G}(r) = q_r$. Again, since $J_3(t)$ is positive definite and monotonically increasing, for asymptotic stability, $J'_3(t)$ must be negative definite. Similar to the earlier case, by the chain rule, we have

$$J'_3(t) = \zeta^T \partial(\zeta)/\partial t \tag{20}$$
$$= -\zeta^T [C_3 \nabla g(z)\, \eta_{3,ij}\, a'_{3,ij} + C_3 \nabla g(z)\, \zeta_{3,ik}\, b'_{3,ik} + I_p\, g(z_j)\, c'_{3,pj}].$$

On the other hand, since $\|\mathfrak{G}\mathfrak{K}\| >> 1$, from Lemma 1 we have $\delta u = \delta r$ and hence $\mu = \zeta$. Therefore, for the weight adjustments given by equation (14), we get

$$J'_3(t) = -\alpha_3 [\, \mu^T C_3 \nabla g(z)\, \eta_{3,ij}]^2 - \beta_3 [\, \mu^T\ C_3 \nabla g(z)\, \zeta_{3,ik}]^2 - \gamma_3 [\, \mu_p\, g(z_j)]^2 \tag{21}$$

which is a negative definite scalar function, except when $\mu=0$ where learning is complete. Therefore from the second method of Liapunov, this learning system is asymptotically stable. This means that the connection weights in the networks will be adjusted until $J_2(t)=0$ and $J_3(t)=0$, or equivalently $\varepsilon = e - \sigma = 0$, $\varepsilon' = e' - \sigma' = 0$, and $\mu = \delta u - \delta v = 0$. However, these imply that $e = e' = 0$, and that the dynamics identifier sub-network $\mathfrak{D}$ acquires the model of the dynamics of the perturbation system. Also, the optimal state feedback sub-network $\mathfrak{F}$ becomes identical to the inverse $\mathfrak{D}^{-1}$ of the dynamics identifier sub-network $\mathfrak{D}$, which generates the compensating torque corresponding to the trajectory perturbation e and e'. Therefore, $q = q_r$ and $q' = q'_r$ as time t approaches infinity (i.e., the manipulator's trajectory q,q' follow the desired trajectory q_r, q'_r). □

6. Conclusion

In this paper, two neural learning controller designs have been considered. They mimic the functions of the cerebellum for the learning and control of voluntary movements and they have parallel processing capabilities which make them fast and adaptable. The designs have several promising attributes that make them very feasible solutions to current problems in Robotics. Most importantly, such controllers are able to approximate the model of the inverse-dynamics of the robot, during the training period. This allows the robot to learn repetitive motions almost perfectly. But even above that, it can perform tasks that it has not been trained to do yet, and to perform them well. In addition, the second design has a good adaptation capability which allows the controller to compensate for unexpected disturbances.

Another advantage of these designs is that they do not require knowledge of the system parameters, and they are robust with respect to parameter variation and disturbances under a variety of tasks. Finally, the parallel processing property of these architectures makes them highly suitable for the integration of a multitude of sensory information into the motion controller networks.

7. References

[1] Rumelhart, D.E., McClelland, J.L., and the PDP Research Group, Parallel Distributed Processing, Volume 1, MIT Press, 1987.

[2] Raibert, M.H., "A Model for Sensorimotor Control and Learning," Biol. Cybern. 29, 1978.

[3] Eckmiller, R., "Neural Network Mechanisms for Generation and Learning of Motor Programs," ICNN, 1987.

[4] Psaltis, D., Sideris, A., and Yamamura, A., "Neural Controllers," 1st IEEE ICNN, 1987.

[5] Kawato, M., Uno, Y., Isobe, M. and Suzuki, R., "A Hierarchical Model for Voluntary Movement and Its Application to Robotics," ICNN, 1987.

[6] Kawato, M., Furukawa, K., and Suzuki, R., "A Hierarchical Neural Network Model for Control and Learning of Voluntary Movement," Biol. Cybern., 57, 1987.

[7] Pourboghrat, F., "Neuronal Controller for Robotic Manipulators," IASTED Int. Symp. Robotics Autom., 1988.

[8] Pourboghrat, F. and M.R. Sayeh, "Neural Network Learning Controller for Manipulators," INNS Conf. in Boston, MA, Sept. 1988.

[9] Pourboghrat, F. and M.R. Sayeh, "Neural Network Models for the Learning Control of Dynamical Systems with Application to Robotics," Springer-Verlag Lecture Notes in Control and Information Sciences (to appear). Also in Int. Conf. Advances in Comm. & Cont. Syst., Oct. 1988.

[10] Pourboghrat, F., "Neural Learning Controllers for Manipulators," IEEE Trans. Syst. Man Cybernetics (submitted).

[11] Pourboghrat, F., "A Learning Algorithm for Temporal Pattern Recognition," IEEE Trans. Syst. Man Cybernetics (submitted).

N90-29032

A Hybrid Architecture for the Implementation of the *Athena* Neural Net Model

by

C. Koutsougeras
Computer Science Department
Tulane University
New Orleans, LA 70118

and

C. Papachristou
Computer Science Department
Case Western Reserve University
Cleveland, OH 44106

Abstract

In this paper the implementation of an earlier introduced neural net model for pattern classification is considered. Data Flow principles are employed in the development of a machine that efficiently implements the model and can be useful for real time classification tasks. Further enhancement with optical computing structures is also considered here.

1. Introduction

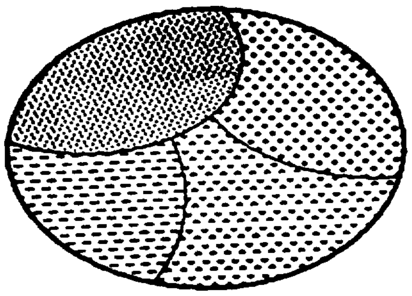
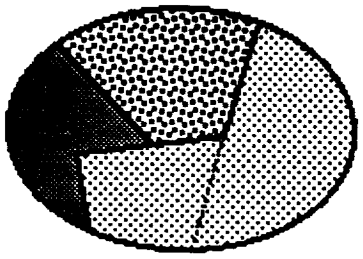
Present day computers depend for their performance on programming. Before any specific task can be carried out by a computer, a programmer has to understand the nature of both the task and the domain of reference. He/she must determine those features of the referenced domain which are pertinent to the task. He/she must also define the basic steps which carry out the task, and the "data structures" which are appropriate for representing the relevant information. Therefore targets of conventional computers were mainly well defined tasks for which complete information can be encoded into the explicit steps of a program. A number of interesting applications however, would require machines to operate with incomplete or without explicit information. In some engineering applications for example, computers are used as controllers to carry out the necessary decision making. Current computers can perform well if the decision process is well understood. Yet in many cases explicit description of a decision process is not available because the relation between the pattern of the environmental variables (input) and the required action (output) is very complex or it is not well understood. Of interest in such cases are machines which can deduce descriptions of the input-output relation from an abstract specification such as a typical set of input-output examples.

Neural nets are machine models which have been developed in the effort to meet this challenge. These models are able to automatically develop internal representations of domain information which is presented to them in terms of domain examples, thus exhibiting true learning by example capabilities. Neural nets operate in two phases. In an initial "training phase" they are presented with factual information consisting of input-output examples typical of a certain desired behavior. During this phase the network's function is adapted so that it becomes consistent with the examples. In the second phase - the normal processing phase - the network produces responses for inputs on the basis of its adapted function or in other words on the basis of its experience. Responses are produced even for inputs which the network may have never encountered before.

There are two fundamental problems associated with some of the most widely known neural net models [Hopf82] [Rume86]. The quality of the internal representations which can be developed with a given net depends on the degree of nonlinearity inherent with the net. The precision of the adaptation therefore depends on the nets size and topology which is an apriori choice. There is no known systematic way to go about this choice. Also these models assume a tremendous number of interconnections which make their implementation with existing VLSI technology very difficult. We have developed an alternative model named Athena which does not face these problems. The network as suggested with this model expands during training and therefore an apriori choice of topology is not required. The structure is tree-like feed-forward and its simplicity allows implementation with conventional VLSI technology. The analysis and the foundations of Athena have been described in [Kout88]. The purpose of this discussion is to present machine architectures considered for implementation. The model is briefly discussed here for reasons of cohesiveness. A machine architecture based on Data Flow models is then analyzed. The proposed machine can be implemented with conventional technology. Another advantage of Athena is that the required computations are such that existing optical processing structures can be efficiently employed. It is explained here how such optical processing structures can be embedded in the original architecture for a significant improvement in speed and throughput.

2. The basis of the internal representations

The subject of this work is the class of stimulus-response relations which can be formulated as a mapping M: V $\rightarrow$ Z where V is a discrete set and Z is a finite set. The elements of Z can be viewed as the classes into which the elements of V can be classified, and M is by definition (and without loss of generality) a many-to-one mapping. Given such a mapping M : V$\rightarrow$ Z, an equivalence relation θ can be defined on V as follows: For any pair of distinct objects X and Y (elements of V), we will say that X relates to Y (X θ Y) if and only if M(X)=M(Y). The relation θ therefore defines a partition of the objects (elements of V) into a number of object classes (equivalence classes) C_i, i=1,2,..k. Since the object classes uniquely identify M, they can be used as a representation for M. In turn the object classes are uniquely defined by a generalized hypersurface which consists of the envelopes of the classes as shown in figure 1a. In [Kout88] we describe a feed forward network model consisting

A mapping M: V->Z clusters V into classes. Lines correspond to a hypersurface on the basis of which M and the classes can be reconstructed.
Figure 1a Figure 1b

of hard thresholding elements which can internally represent an approximation of a hypersurface by means of hyperplane segments. The function of the model can adapt to any given mapping M by internally representing the corresponding hypersurface which partitions V into the object classes defined by M. The approximation is exact if the input space V is discrete (figure 1b). The hypersurface is incrementally formed on the basis of the available factual information consisting of a collection of objects (elements of V) of known classification. This incremental process is guided by an entropy measure. The hypersurface partitions V into an expectedly minimal number of convex regions (sets) each of which contains objects of a single class. Given an object X of unknown classification, that is one which was not encountered in the training set, assume that it belongs to a certain convex region S_k of the final partition. If all the instances of the training set which fall within S_k belong to the same object class, let's say C_k, then within S_k the classification decision is predominantly C_k and therefore with a high degree of confidence X's classification should also be C_k. For the reasons of completeness and reference the model is also briefly discussed in the following.

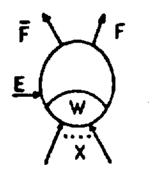
3. Structure and training

The building element of this model is a new type of threshold unit which is described in figure 2. Each unit has a set X of n "data" input lines, a control or enable input E, and two outputs F and F'. With each unit a weight vector W and threshold T are associated, representing some hyperplane $P = \{X | X \in R^n \text{ and } W^t X = T\}$. Then the functions F and F' represent the upper P^u and lower P^L half-spaces respectively. While the E input of a unit is not activated (E=0), both the outputs F and F' of that unit are inactive (F=F=0) regardless of the input X. When a unit is fed with an input vector $X \in R^n$ and its E input is activated, then F and F' are complementary assuming the values :

$$F=1 \text{ and } F'=0 \text{ if } W^tX \geq T$$
$$F=0 \text{ and } F'=1 \text{ if } W^tX < T \qquad (1)$$

where W^tX is the scalar product of W and X. For an enabled unit the meaning of the above equations (1), is that the unit's F output is active if the input X is on a certain side of the hyperplane that unit represents. If X is on the other side of the hyperplane then F' is active.

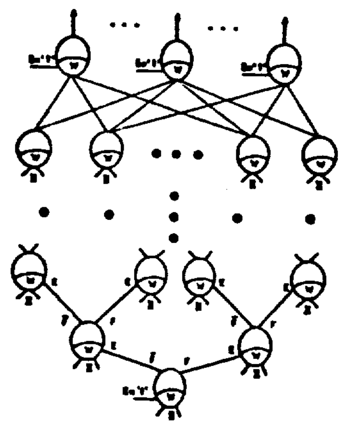
Each of the outputs of every unit is connected to the E input of another unit in the next higher layer. In this way, a set of units is connected to form a D_tree hereafter referred to as a D_tree. The enable input of the unit at the root of the D_tree is set to be always active. Any input vector presented to the network, is routed to the data inputs of every unit in the D_tree. During the presentation of an input vector X, the following operation takes place in the D_tree. The input X is broadcasted to all the units and for each unit at most one output is activated. Thus, in the whole network at most one path is activated for each input vector. If, given the input X, unit u_k is enabled (its E input is activated), then necessarily all of its ancestors are also enabled. If the ancestors of u_k represent the hyperplanes $P_1, P_2, ...P_j$, then u_k is enabled when X lies within one of the convex sets into which $P_1, P_2, ...P_j$ subdivide the input space V. Thus, a certain unit u_k is enabled only when the network's input lies within a certain



$$F = \begin{cases} E \text{ and } (W,X) \geq T \\ 0 \quad \text{Otherwise} \end{cases}$$

$$\bar{F} = \begin{cases} E \text{ and } (W,X) < T \\ 0 \quad \text{Otherwise} \end{cases}$$

A single unit
Figure 2



The network structure
Figure 3

bounded subset of the input space, hereafter referred to as the space of activity of u_k, and further, u_k divides that subset in two parts. In like manner, each output F_i of a unit at a leaf of the D_tree represents a convex subset V_i of the input space.

A final layer of output units of the same type completes the network. Each output unit corresponds to a single object class. The purpose of each output unit is to perform the logical OR function among a selected number of the tree's outputs. This operation is conceptually equivalent to the formation of the union of a selected number of convex sets V_i of those corresponding to the outputs of the D_tree. A selected set of outputs from the leaf units of the D_tree are used as data inputs for an output unit. The output unit corresponding to the class C_i collects those outputs of the D_tree representing the convex sets which form C_i. The E input of the output unit is set to be always active. The OR function is performed by an output unit by setting its weight and threshold values as follows. The threshold value is set to $1 \in R$. The components of the weight vector which correspond to the selected outputs of the D_tree are set to 1 and all others are set to 0. The complete network structure is illustrated in figure 3.

Adjustment of the network's weights (W_i's) and thresholds (T_i's) takes place incrementally starting from the lowest layer (consisting of only the root of the D_tree) and proceeding layer by layer towards the leaf units of the D_tree. This adjustment is based on a collection of input-output examples (the training set) which is a typical sample set of a target mapping M as earlier explained. The training set is presented a number of times and at the n-th iteration the W's and T's of all the units at the n-th layer are determined in parallel but independently of each other. The rationale of this adaptation process can be briefly explained as follows.

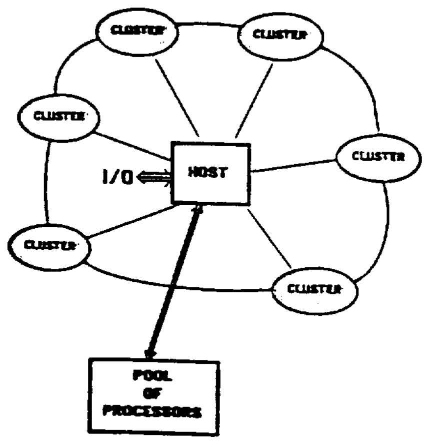
After the training set has been presented n-1 times all units from the root of the D_tree up to (and including) layer n-1 have been assigned W and T values defining the hyperplanes of an intermediate partition. Thus the activity space of each unit in the n-th layer is determined. During the n-th presentation each unit in the n-th layer is allowed to "observe" only those instances of the training set (that is those of the presented examples) which are pertinent to its own activity space. On the basis of the "observed set" then each unit determines a hyperplane which further partitions the region corresponding to its activity space in two parts. This hyperplane must be the one which is "most useful" in discriminating among instances of different classes. The estimator used in this model to measure the goodness of a hyperplane is the entropy. In [Kout88] we describe in detail a constrained optimization process for the hyperplane selection which is based on the optimization of entropy. The exact optimization of the entropy is a laborious process which is plaqued by the dimensionality of the input space and for this reason the constrained optimization process further employs heuristics based on discriminant analysis techniques. The greediest computation of those required in the constrained optimization process is the inversion of a matrix and therefore its complexity is that of the matrix inversion.

The hyperplanes are the means by which decision information is internally represented in the network. In keeping the system's internal representations as simple as possible (as few hyperplanes as possible), it is expected that these representations capture the structure or regularities which are possibly exhibited by the input space. The reason simply is that if the system operates properly by having acquired a minimal amount of information, then necessarily that information must be of high quality, reflecting the essential characteristics (e.g. structure, regularities) of the input space.

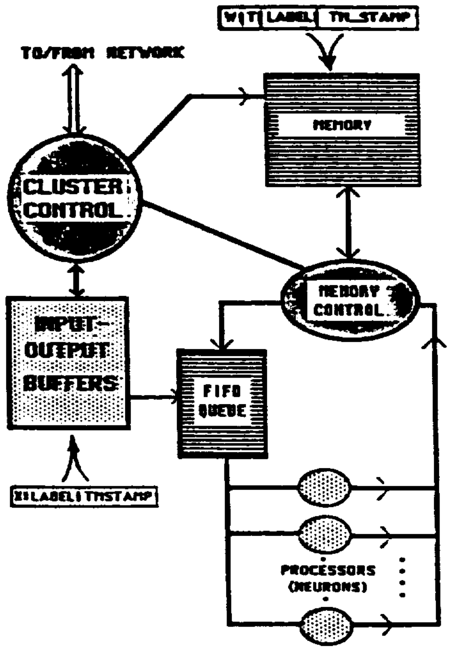
4. The model's implementation

In the following we outline our approach towards the development of a machine architecture. The net model (Athena) we have developed has a particularly simple structure. In contrast with other models the number of the required interconnections is small and therefore it is possible to implement with current VLSI technology. The acyclic nature of the model makes the use of parallel Data Flow architectures [Wats82], [Denn80], [Arv83], particularly efficient for its implementation. Such an architecture is presented in this section.

The overall architecture consists of a host I/O processor and a number of clusters interconnected in a ring structure as shown in figure 4. This machine is intended to simulate concurrently a large number of networks of the Athena type described earlier in this paper. We will hereafter refer to the stored representation of an Athena type of network simulated on this machine as a *virtual network*. The D_tree of a virtual network is implemented in one or more of the clusters. The functions of the units of the output layer of a virtual network are performed by the host. All inputs are entered in the system through the host. When an input intended for a particular virtual network is entered, the host attaches to the input a timestamp and a label identifying the virtual network and distributes it to the clusters.



Overal structure
Figure 4



Architecture of a single cluster
Figure 5

The basic architecture of a single cluster is shown in figure 5. It consists of a memory module which is accessible by a memory controller. The controller communicates with a pool of independent (but identical) processors through a unidirectional loop link. These processors are dedicated to the cluster and will be called hereafter loop processors. The memory stores representations of many networks of the Athena type. A network is represented in the memory as a linked list of records. Each record represents a unit of a network. Each record therefore consists of a number of fields containing information about the weight vector and threshold associated with the corresponding unit, as well two "destination fields" identifying the records corresponding to the children units. Additionally, there is a label field and a timestamp field associated with each record. The label field identifies the virtual network to which the record belongs. The use of the latter two fields will be explained in the following.

The loop processors of a cluster are all identical and can operate independently of each other in parallel. Their purpose is to simulate the functions of the units in an Athena network at the normal processing mode. Therefore these processors are only required to perform the scalar product of two vectors and the threshold function.

An independent pool of processors is utilized exclusively for matrix and other computations needed during the training phase. These processors are not associated with any particular cluster. These processors are used to run the training algorithm by which the weight vector and threshold are determined for each unit of a virtual network. The computed weights and thresholds are then communicated to the appropriate cluster where they are assigned as values to the appropriate fields of the corresponding records. These computations are considerably more sophisticated compared to the scalar product and threshold function needed during the normal processing mode. Furthermore, training takes place only once for each virtual net. For this reason it would be inefficient to require that the loop processors within each cluster be sophisticated enough to carry out the computations required by the training algorithm. The simple threshold test is the only function needed during the normal processing phase of a virtual net and it is therefore the most frequently executed one.

* In this machine the development of a virtual network is carried out as follows:

During the first presentation of the training set, the weight vector and threshold of the root of the virtual tree must be determined. One processor of the pool is assigned to gather the presented examples and compute the W and T values. These values are then sent over to a cluster's memory controller which forms a record and stores it in the memory. The destination fields of this record contain the addresses of two other records with null values for their weight, threshold and destination fields. The labels of the new records are set to the same value as that of their parent.

A partially developed virtual net is recursively expanded as follows. During a new presentation of the training set each example input is processed by the existing partial virtual net. In this way it is determined how the training set should be partitioned for the computation of the weights and thresholds of the leafs (records with null fields) of the current virtual net. Each subset of the partitioned training set is then assigned to a processor and the computed W and T values are subsequently passed on to the corresponding records (in the appropriate cluster). If a certain subset contains examples of only one class then no W or T values are computed and the corresponding record's fields remain null except that the record is marked with a label identifying that class and further, no children are linked to it.

As shown in figure 4 the basic architecture consists of a number of clusters which are interconnected by a unidirectional ring communication network. All clusters are architecturally the same having the structure shown in figure 5. In this architecture a virtual network can expand over more than one clusters. If the available capacity of the memory module in a cluster does not allow completion of the development of a virtual network in that cluster, then free space is seeked in other clusters and development of the virtual network continues there. The process of developing the virtual network within a cluster is carried out as described above. Suppose now that the W and T values for a record are computed but no children can be linked to this record due to unavailability of memory space. Then the W and T values just computed, are not assigned to this record. Instead, a request seeking free space in another cluster is submitted over the communication network interconnecting the clusters. When one is found, a new record is formed in the new cluster, the computed W and T values are assigned to it and the record in the original cluster obtains a pointer address (link) to it. The original record maintains null W and T fields and it is only used for binding purposes. Such records used for binding the parts of a decision D_tree which is distributed over a number of clusters will be called "dummy" records. The function of dummy records is not any part of determining an output, it rather is to designate the fact that the signal they receive must be communicated outside the environment of the cluster.

* During the normal processing mode the function of a virtual net is simulated as follows :

Activation of a unit is simulated by "firing" its corresponding record in the following way. The complete information contained in the record representing that unit is duplicated in a *packet* which is then sent to the processing FIFO queue. If there is an idle processing unit, it gets the packet removing it from the queue. This processing unit then computes the scalar product and performs the threshold comparison. The result of this computation determines which of the "destination" units should be activated. The processing unit finds the address of the corresponding destination record from the destination fields of the packet it has acquired, and simply sends this address to the memory controller. When the controller receives the address, it fires the appropriate destination record, that is, it forms a new packet corresponding to the destination record and sends it to the processing queue repeating the above cycle.

Firing of a dummy record does not produce a packet to be send to the processing queue. An address is only obtained and sent either to the host or to another cluster via the cluster's local control which interfaces the cluster with the (ring) communication network. If the dummy packet represents an output of the corresponding D_tree (exit), then the address is sent to the host. If it represents binding (continuation of the D_tree) to another cluster, then the address is sent to that cluster. In the destination cluster then, the binded root will be fired continuing the process there.

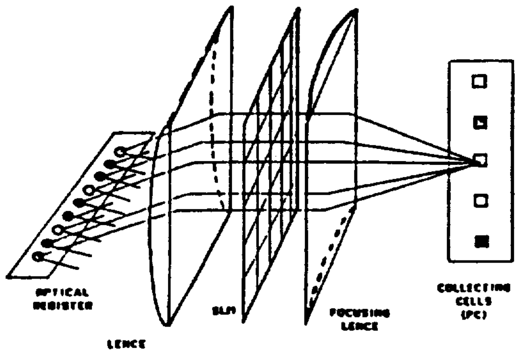
Each packet send to the queue carries a timestamp and a label field identifying the decision tree (D_tree) to which it belongs. Such a label field is needed for the following reasons: In the whole system a lot of virtual networks are stored. When in the normal processing mode the host receives an input intended for a particular virtual network, the host attaches to the input the label of the corresponding decision tree and a timestamp and distributes the labeled input to the clusters. Each cluster containing a part of that decision tree stores a copy of this input in the input buffer. When a processing unit receives a packet, it obtains the input intended for use with this packet, by accessing the input buffer for that input whose label matches that of the received packet. The computations implied by each conceptual decision tree stored in the local memory, are carried out in parallel without affecting each other. Thus parallelism is only limited by the physical delay characteristics and the number of the processing elements. Not only inputs for different decision trees can be processed concurrently, but different inputs intended for the same decision tree can also be processed in parallel as follows:

Each input received by the host is timestamped in addition to being labeled before being distributed to the clusters. The input buffer at each cluster may contain many different inputs intended for the same decision tree, but each input carries a different timestamp. As earlier mentioned, when a processing unit processes an input, the result is a destination address which is sent to the memory controller. The destination address also carries the timestamp of the processed input. The same timestamp is also passed on to the copy of the destination packet which is sent back to the processing units. When the new packet is processed, the input buffer will be accessed for an input whose label and timestamp matches those of the packet being processed. Therefore, if a certain input is used in processing a packet, the same input exactly will be used in the processing of the corresponding destination packet.

The timestamps must also be used when firing a dummy record. If the dummy record represents binding then its timestamp along with the address of the corresponding (binded) root of the virtual network's subtree is obtained and submitted to the appropriate cluster. If the dummy record represents an output leaf unit of the virtual network then its label and timestamp are submitted to the host which furnishes the output.

5. Use of optical processing

We now consider the processing units within a cluster. These units are required to compute the scalar product of two vectors and the threshold function. For this purpose the loop processors can be implemented as simple pipeline processors using conventional VLSI technology. However, the functional requirements for these



Optical processing structure from [Casasent88]
Figure 6

processors are such that advantage can be taken of optical processing structures already proposed. The structure of figure 6 has been proposed by Casasent [Casasent88] for the simultaneous computation of the scalar products of a vector X and a set of vectors Y_j, j=1,2,3, ... The vector X is held in an optical register consisting of an array of laser LED's which is shown in figure 6 positioned on the focus line of the first lens. The vectors Y_j are stored in the spatial light modulator (SLM). The beams on a horizontal plane coming out of the first lens constitute a copy of the optical register. One copy is therefore available for each row of the SLM and so all the partial products $X_i Y_{ji}$ are available in parallel, encoded by the intensity of the beams coming out of the SLM. All beams coming out of a single row of SLM cells are focused by the second lens on a single point. An array of photosensitive cells (one for each row of the SLM) is positioned on the focus line of the lens. If r weight vectors can be stored in the SLM, then with this structure r scalar products can be computed concurrently and extremely fast.

The output of each PC cell must be compared against a corresponding threshold. This comparison can automatically be performed if a bias proportional to the threshold is used on the PC cell. The reason that full advantage of this structure can be taken for implementing Athena is that it is fit for the kind of computations required in Athena, that is, all the units in a D_tree require the same input vector for the scalar product computations. However, if this structure is employed to implement the processing elements of a cluster, then firing one packet at a time would not be the most efficient way. Rather than firing a single packet at a time, a whole virtual subtree of packets is fired as as follows. When the memory controller receives the address of a packet to be fired, it extracts that packet and r-1 of its successors in a Breadth First manner, where r is the capacity of the SLM in any of the (optical) processing units. All of these packets are sent as a group to the processing queue. An idle processing unit gets this group. The W vectors are stored in the SLM and the corresponding thresholds are used as biases on the PC cells. The appropriate input X is obtained from the input buffer after matching the group's label and timestamp to those of X. All of the outputs of the subtree corresponding to the group are computed in parallel. A binary search through these outputs yields the single output which the subtree produces when processing X and which determines which packet should consequently be fired. The processing unit then sends the address of the packet to be fired along with the virtual network label and the timestamp (these are the same as those of the group processed) to the memory controller. If the packet specified to the controller by this address is a dummy packet, the controller submits the received address to the network via the local control unit. Otherwise a new subtree will then be fired having this packet as a root. The advantage of using this optical processing structure over conventional processors is obvious.

6. Conclusion

In this paper we reviewed the operation of an earlier introduced neural net model. Targets of this model are general classification tasks and more specificaly the automatic development of representations for the necessary classification rules or class descriptions on the basis of example information. The purpose of this paper was to outline a machine architecture that can implement this model. It was shown here that it is possible to efficiently implement the model in current technology using Data Flow architecture principles. For the purposes of real time applications the ability to physically implement is a particular advantage of this model over other existing ones. It was also shown here that advantage can be taken of optical processing structures proposed by other researchers. A particular optical structure which meets the model's computational needs can be effectively embedded to further enhance significantly the efficiency and throughput of the basic machine.

References

Arvind et.al., "The Tagged Token Data Flow Architecture", MIT, August 1983

D. Casasent and E. Baranoski, "Directed graph for adaptive organization and learning of a knowledge base", in Applied Optics Vol.27, No. 3, February 1988

J. B. Dennis, "Data Flow Supercomputers", in Computer Vol.13, No.11, Nov. 1980

R. Duda and P. Hart "Pattern Recognition and Scene Analysis", John Wiley & Sons 1973

R. A. Fisher "The use of multiple measurements in taxonomic problems", Ann.Eugenics, 7, Part II, 179-188 (1936); also in Contributions to Mathematical Statistics (John Wiley, New York, 1950)

J. Ghosh and K. Hwang, "Critical Issues in Mapping Neural Networks on Message-Passing Multicomputers", in Proc. 15th Annual Internat. Symposium on Computer Architecture May 1988

J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proc. Natl. Acad. Sci. USA, 79, 2554-2558 1982

D.A. Huffman "The Synthesis of Linear Sequential Coding Networks", Proc. Symposium on Information Theory, London, 1955

D.A. Huffman "A method for the construction of minimum redundancy codes", Proc. IRE, September 1952

T. Kohonen "Self Organization and Associative Memory", Springer-Verlag 1987 (second edition)
C. Koutsougeras and C.A. Papachristou, "A Neural Network Model for Propositional Logic Functions", Computational Intelligence Internat. Conf. Elsevier-North Holland, Amsterdam, September 1988

C. Koutsougeras and C.A. Papachristou, "Training of a Neural Network Model for Pattern Classification Based on an Entropy Measure", in Proc. IEEE Internat. Conf. on Neural Networks (ICNN '88), New York: IEEE, July 1988.

C. Koutsougeras and C.A. Papachristou, "A Neural Network Model for Discrete Mappings", in Proc. IEEE Internat. Conf. on Languages for Automation (LFA '88), New York: IEEE, October 1988.

D. Psaltis and M. Neifeld, "The Emergence of Generalization in Networks with Constrained Representations", in Proc. IEEE Internat. Conf. on Neural Networks (ICNN '88), New York: IEEE, July 1988.

D.E. Rumelhart, G.E. Hinton and R.J. Williams "Learning Internal Representations by Error Propagation", in Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol1: Foundations, D.E. Rumelhart and J.L. McClelland (editors), MIT Press, Cambridge, MA (1986)

I. Watson and J. Gurd, "A Prototype Data Flow Computer with Token Labeling", in AFIPS Conf. Proc. Nat'l Comput. Conf. June 1979

I. Watson and J. Gurd, "A practical Data Flow Computer", in Computer, Vol.15, No.2, Feb. 1982

N90-29033

A DESIGN PHILOSOPHY FOR MULTI-LAYER NEURAL NETWORKS WITH APPLICATIONS TO ROBOT CONTROL

Nader Vadiee and Mo Jamshidi

CAD Laboratory Systems/Robotics
Department of Electrical and Computer Engineering
The University of New Mexico
Albuquerque, New Mexico 87131
USA

ABSTRACT

A system is proposed which receives input information from many sensors that may have diverse scaling, dimension, and data representations. The proposed system tolerates sensory information with faults. The proposed self-adaptive processing technique has great promise in integrating the techniques of artificial intelligence and neural networks in an attempt to build a more intelligent computing environment. The proposed architecture can provide a detailed decision tree based on the input information, information stored in a long-term memory, and the adapted rule-based knowledge. A mathematical model for analysis will be obtained to validate the cited hypotheses . An extensive software program will be developed to simulate a typical example of pattern recognition problem. It is shown that the proposed model displays attention, expectation, spatio-temporal, and predictory behavior which are specific to the human brain. The anticipated results of this research project are (1) creation of a new dynamic neural network structure, (2) applications to and comparison with conventional multi-layer neural network stuctures such as multi-layer perceptron, neo-cognitron, etc. The anticipated benefits from this research are vast. The model can be used in a neuro-computer architecture as a building block which can perform complicated, nonlinear, time-varying mapping from a multitude of input excitory classes to an output or decision environment. It can be used for coordinating different sensory inputs and past experience of a dynamic system and actuating signals. The commercial applications of this project can be the creation of a special-purpose neuro-computer hardware which can be used in spatio-temporal pattern recognitions in such areas as air defense systems , e.g. target tracking, and recognition. Potential robotics-related applications are trajectory planning, inverse dynamics computations, hierarchical control, task-oriented control, and collision avoidance.

1. INTRODUCTION

The problem and opportunity that is discussed in this paper is the subject of current research in the field of neurocomputing. The neural networks that are in current use lack the dynamic behavior, i.e. attention, expectation, and temporal trends recognition abilities.

Adaptive Resonance Technique models such as ART 1 and ART2 , proposed by Carpenter and Grossberg [1,2] are two typical examples of neural nets with dynamic behavior. According to Rumelhart, et.al. [3] internal representations can be learned using error back propagation. Lippmann [4] gives a thorough discussion of different neural networks and their potentials. We will analyse those nets using our new approach discussed in Part 3. Self-organizing Kohonen nets [5-8] will be tested for their pattern classification capabilities for our purpose. The neural network called "neocognitron" [9-11] is used to test the ideas of clustering in the hidden layers and temporal trend analysis. One of the investigators'interest areas lies in robot systems. There has been some research undertaken in applying neural nets to the control of robot manipulators [12-14]. Bavarian [15] gives examples of the applications of neural nets to fault tolerant systems.

The investigators believe that the proposed neurocomputer and the approach unifies all these efforts and provides a general scheme for analysis and design of special-purpose neurocomputers. The type of operation that almost all living organisms or man-made engineering systems as information processors perform can be thought of as a mapping from some element in an input set χ to an element in some output set Ψ, i.e χ -- M --> Ψ.

The set χ contains elements of input information that have to be represented by some appropriate representational system, M is the process of mapping or decision making, and Ψ is the set containing possible decisions, and/or commands to be reached. The process of mapping , M, can be very complex and is based on input information, past experiences and/or a set of rules, beliefs, values, expectations criteria and constraints.

If the objects or elements in the input set can be represented by n-tuple vectors X and elements in the output set be represented by m-tuple vectors Y using some sort of sensing, coding or representational system, then M becomes a complex , time varying, nonlinear and adaptive mapping function f. Then the engineering problem in its most general form is defined by f : X(t) --> Y(t). Figure 1 shows a pictorial description of the general dynamic mapping problem. Here, sensing and preprocessing of the input information provides the system with r-classes of input excitory, i.e., X1, X2, ..., Xr. The STM (short-term memory) blocks give a dynamism to the system that enables it to derive the temporal patterns and trends by storing the recent history of the input classes Xi(t), i=1,2,...,r. This history is represented by Xi(t-ti), Xi(t-2ti), ..., Xi(t-kti) values.

Each Xi can have its own representational system and in general Xi's are vectors of ni dimension that each component is discretized by pi levels, and hence there are pi ^ ni different possible Xi points in the input subspace Si.

The block called LTM (long- term memory) may act as a rule-based data structure which can modify and facilitate the mapping function f. Y1, Y2,......, Ys represent different output classes that can be different motor signals and/or commands with some desired time sequence. Thus, the function f may depend on patterns found in input, temporal patterns in past experienced inputs, patterns in LTM, and a system of rules. The Xi's vectors belonging to different input classes may be corrupted by noise and/or damaged as a result of: ***Translation, Rotation, Distortion, Scaling, Styling, Partial occlusion and marring*** that may happen to the input sensory information. The dimension of the input vectors Xi can be excessively high.

The output Yj can be an input Xi exemplar when the above mentioned noise is filtered out, or the above mentioned damages are compensated for. The output Yj can be a set of exemplar classes, category numbers, or decisions made. The input vectors Xi can be sensory input from vision, touch, accoustic, and state-space sensors in a robotic system, and Yj's can be different actuation signals sent to actuators. Moreover, f is the required coordination between input and output. The mapping function f is the information processing operation that can be:

- Mathematical mapping approximation, developed for a funcion f : X CR^n ---> Y C R^m
- Extraction of relational knowledge from binary data bases
- Probability density function estimation
- Pattern classification
- Categorization of data
- Process of decision making
- High, medium, and low level control law executions

As far as the on-line information processing capabilities of the current analog electronic circuits and systems and the off-line information processing capabilities of modern serial computers are concerned, it seems that they have certain limits in a variety of applications. In modern conventional computers the information processing operation f has to be explicitly known, procedural, and programmable. The process is done in steps and in serial. The input information cannot undergo a great deal of distortion and damage due to noise and/or other previously mentioned causes.

A look-up table fashioned for f may not be feasible when the input vectors are of high dimension. The most important of all, the information processing operation f simply is not well known and has to be learned through experience and examples.

In summary, in cases where the input is of a very high dimension, is corrupted by noise, damaged by distortion and the function f is not explicitly known and has to be learned, adapted, or is a complex nonlinear function, then the conventional computers, analog or digital, are incapable and a new computing model and philosophy is needed.

The specific technical problem or opportunity addressed is that we believe that all the information processing carried out in most neural nets is based on this paradigm that in order to map X to Y the neural net first maps the vector X to a set of vectors ei, i = 1,2,....,r, in feature spaces F1, F2,...., Fr by feature extraction abilities of the processing nodes,. Then the feature vectors ei's are mapped to the appropriate output yi's. The idea is this : if Xi --> Yi and Xj-->Yj and we have ||yi-yj|| < d, where ||.|| is some defined norm, and d is some small positive real value, then we have : **Xi is similar to Xj**. Also we believe that if we continue the feature extraction operation by mapping X's to a set of feature spaces, then there exists a certain feature space in which two similar inputs Xi and Xj are mapped to two close or neighboring points such that $||e_{ki} - e_{kj}||$ **< epsilon** for some 1<k<r and a small positive value epsilon.

Similarity is a relative concept that is defined with respect to a set of features found in a feature space. So we can say that similar inputs produce clusters of points in certain feature space. These clusters can be detected using self-orgainizing neural networks.

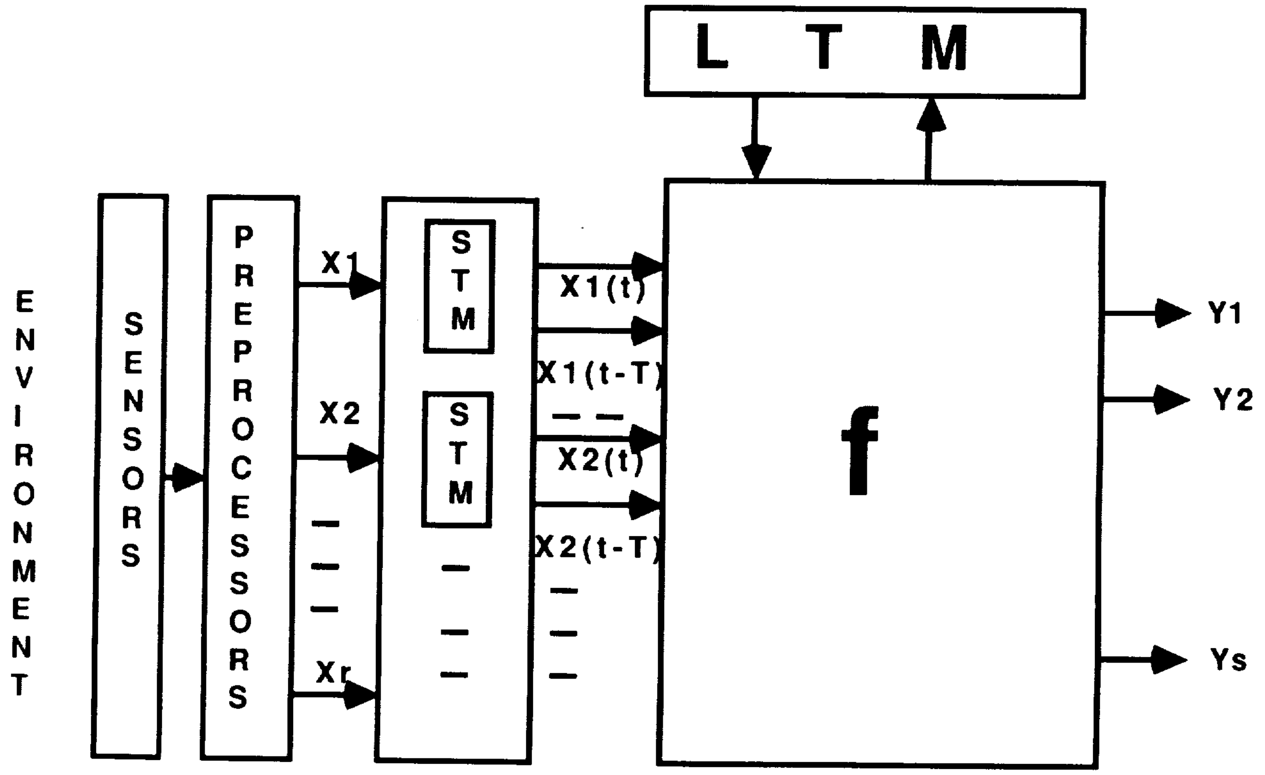


Figure 1. A Pictorial Description of the General Mapping Problem

2. PROBLEM STATEMENTS

The statement of the problems addressed here can be briefly summarized as follows :

1) Is there any clustering phenomenon occurring in the hidden layers of a multi-layer feed-forward neural network ? If so, is it possible to detect these clusterings by using some self-organizing neural nets, for example, a Kohonen map ?

2) Is it possible to recognize the temporal patterns and trends in the features using short-term memory units at the output of the above-mentioned self-organizing layers ?

3) Is it possible to continue the feature extraction and pattern classification action using the self-organizing maps' outputs augmented and made into a new input layer, and thus forming a categorizing tree describing the input ?

4) It is desirable to build neural nets in a bidirectional way to investigate the effect of priming of the former layers'nodes using the feedback from the latter layers' nodes.

5) It is interesting to prove the idea that if two inputs cluster in some hidden layer Fi, then they would cluster in all other layers Fj for any j >= i.

6) By simulating the visual pattern recognition net, i.e. "neocognitron", we would like to detect the dynamism in features. For example, to recognize the temporal changes in some specific features. It is possible to detect the motion of a hand-written number across the input field providing that all other features are fixed. Thus, it is possible to build temporal pattern classifiers.

7) In currently used neural networks, we start by knowing almost nothing about the mapping function f and begin the process of adaptation using a set of training points. We want to investigate a new idea that allows us to add the knowledge, if any, that we have about the mapping process to the system. This knowledge can be a set of rules from an expert. In other words, we would like to investigate a method to integrate the traditional techniques of artificial intelligence with those of artificial neural networks.

3. PROPOSED NEUROCOMPUTER ARCHITECTURE

We can simulate any unconventional transfer function from multiple inputs to multiple outputs using neural networks. The interesting point is that the transfer function can be learned by examples and experience. The inputs can be corrupted by noise and the system still remain to a high degree fault tolerant.

Traditional engineering systems extract analytical well defined features. For example, the first derivative, average, time integral, etc. In neural networks we can extract or even discover quite complicated features that are not explicitly known. Every object in the world has some specific features that distinguish it from the rest of the objects and any two objects are related from certain feature points.

Neural networks use a combination of state space transformation, feature extraction, and curvilinear transformation to reach to a canonical form for an otherwise complex mapping function. A set of objects can be classified in an infinite number of ways depending on the common defined features.

In the feed-forword multilayer perception, during the process of training by using the back propagation learning algorithm, the hidden layers are forced to form those features that would highlight the similarily between the inputs. In these nets, once the process of training is finished no further changes in the network will occur. By providing a type of feedback loop in the neural net, we are able to give the net some kind of attention and expectation abilities usually seen in natural intelligent systems. Sometimes the trend in input / output is such that certain priming of the sensory nodes or feature detection nodes can be achieved.

A network that has attention capability is able to tune on certain input patterns and ignore the others. It is able to prime the sensory nodes to make them provide more information of the kind that is most necessary for classification or mapping action. Sometimes the time sequence of output is associated with some form of time sequence of input, and certain expectancy state is aroused in the neural system. If it becomes strong enough, it may ignore and cut off the input and provide the output through the expectation mechanism. The priming action can be achieved through the inhibitory inputs of "instar" nodes or through their threshold level values.

The attention and expectation capabilities provide the net with a sort of dynamism that increases the speed of processing and saves time in computations. As far as the application to the control of robotic systems is concerned the neural nets can be used wherever a system identification or input / output approximation is required. The neural net gives the mapping between the sensory input and the controlled outputs. One can distinguish the following subsystem in a neuro-computer structure: ***1. Input layers , 2. Feature layers, 3. Kohonen layers, 4. Output layers, 5. Expectation FBK loops, 6. Attention FBK loops, 7. Temporal features detection mechanisms, 8. Short-and long-term memory blocks***.

If the nodes in the Kohonen layer, when excited, show a kind of persistence with an exponential decay function, this will provide means of studying the temporal patterns of the input sensory signals and extract the temporal features and store them in a new layer for further processing.

Computational algorithms and codes will be developed to simulate the input /output function of a single artificial neural node in all its functionalities. It is believed that by too much simplification we lose a lot of capabilities that a single processing node can have.

Typical multilayer feed-forward artificial neural nets are simulated and the idea of clustering in each layer is investigated by using Kohonen self-organizing layers, complex decision boundaries can be formed using line segments, planes or hyperplanes. The idea of using curvilinear mapping is investigated and is used in reducing the complexity of decision boundaries.

Bidirectional associative memories are implemented to study the effect of feedback paths on the performance of multilayer nets. It is believed that by using feedback paths and controlling the threshold levels in each layer we are able to give a kind of expectation and attention capabilities to neural processors. This will increase the speed of processing and save computational time . Hierarchical structures provide a kind of tree that classify the input data. The notion that Kohonen layers are the best candidate which can integrate neuro - computers with other more traditional types of computers will be investigated.

If we build the basic processing nodes in their full input/output characteristics, the decaying output will provide us means to analyze the temporal patterns. New temporal features can be extracted using Kohonen layers retain previous inputs. Relative intensily of nodes provides us a sense or means of measuring time. The training set provides stable points in each layer that make the centers of clusterings. These clusterings can be surfaced by using Kohonen layers.

If two input vectors cluster in a feature layer ej , they will cluster in feature ei for $j < i < r$. The best training set is the set providing the largest number of classes in the hidden layers. Each input vector Xi is stored in r feature vectors eij, $j = 1,\ldots,r$ and in r Kohonen layer vectors Kij, $j = 1,\ldots, r$, and so on. By exciting any of these vectors we should be able to excite all others. It means that this structure can have mental images within the context of psychology.

The neuro - computer architecture proposed consists of successive layers of **1. Feature layers , 2. Class layers (Kohonen), (pattern classifiers), 3. Motor output layers (integrating layers).** These nets are completely bidirectional and the output of Kohonen layers can be interfaced to digital computers .

If there are n layers and p input exemplars, there will be n^p stable vectors in the layers. A new input should cluster around one of these n^p vectors. These p input exemplars should not be correlated and there should be some differences in feature. The success in convergence depends on how well these n ^ p vectors are representative of all possible future input data.

The end result of a supervised learning is producing clustering regions in the one to the last layer. The efficiency of training algorithms is studied as applicable to producing these cluster regions around the examplars. Kohonen layers, if proved useful for our purpose, can provide a means of

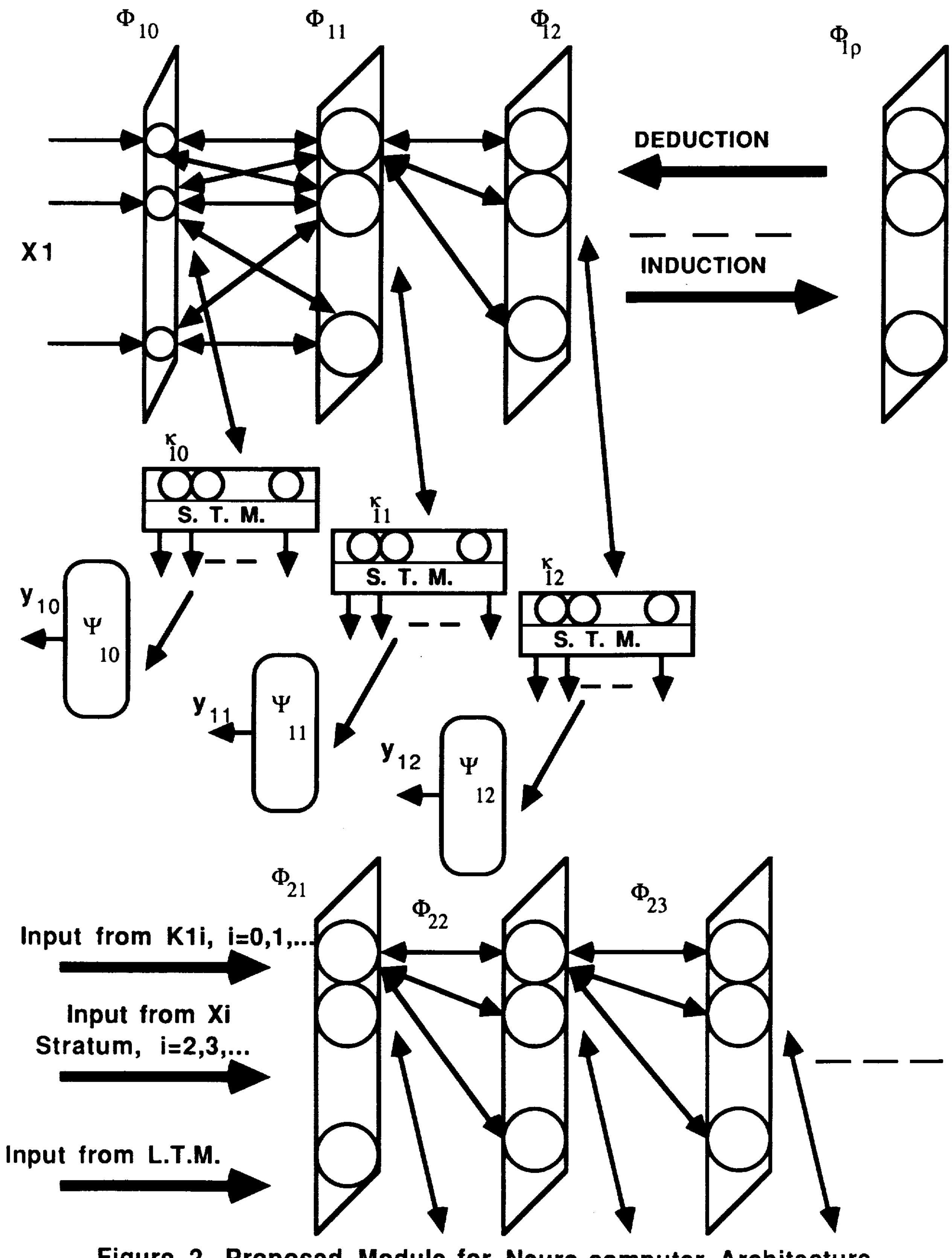


Figure 2. Proposed Module for Neuro-computer Architecture

integrating neural computers. The possibility to analyse temporal patterns provides ways for robotic system's dynamic trajectory planning.

Attention and expectation capabilities reduce the processing time in input data classifications. Figure 2 illustrates the proposed stucture for a generic subsystem for neurocomputers. In order to simplify the illustration, we concentrate on input class X1 only. Similar arguments can be applied to other input classes not shown. X1 is the input layer. O_{1j} 's are feature layers for the input class 1, k_{1j} are self-organizing layers, Y_{1j}'s are different branching outputs. The outputs from self-organizing layers can be augmented with outputs from other input classes , and outputs from LTM constitute a new higher level input layer.

Continued feature extraction in successive layers results in the formation of disjoint decision regions which are linearly discriminant. These disjoint decision regions can be detected by clustering methods using self-organizing neural networks. The clustering layers form a decision tree. This tree can be used to classify and categorize input information for future decision making.

4. FUTURE TRENDS AND POTENTIAL APPLICATIONS

The results obtained from this proposal will help understand the internal process going on in a neural network. It shows how the optimum set of distinctive features is extracted in the process of learning and how these features are used to classify the patterns. Attention, expectation, use of short-term memory, and long-term memory give a sense of dynamism that the current neural networks are lacking.

The initial phase of the research will provide the mathematical model for further design and implementation of adaptive dynamic controllers for different applications. The mathematical model and software programs developed will be used to design task-oriented high level controllers for such systems as a robot manipulator. The hardware implementation of the proposed neurocomputer will be postponed until a second phase of the project.

A neurocomputer with dynamic transfer function equipped with attention, expectation, and temporal trend analysis can be used as a special-purpose computer that can be integrated with conventional digital computers. The techniques from artificial intelligence can be used to equip the neurocomputer with rule-based intelligence. These special neurocomputers can be used in aerospace avionics, aircraft control, low-cost visual inspection systems in manufacturing, power plant fault detection, just to name a few areas of applications. This special-purpose computer can be used for many nonlinear and time-varying problems associated with a robot control system such as inverse kinematics, trajectory planning, vision, and control. These applications are being simulated in C language using a MACII computer.

5. BIBLIOGRAPHY

1. Carpenter and Grossberg, " ART 2 Self Organization of Stable Category Recognition Codes for Analog Input Patterns," Applied Optics, Dec., 1987.

2. Carpenter, G. and Grossberg, G. " The ART of Adaptive Pattern Recognition by a Self Organizing Neural Network," IEEE Computer Magazine, March 1988, pp. 77.

3. Rumelhart, D.E., Hinton, and Williams, "Learning Internal Representations by Error Propagation, " in Parallel Distributed Processing : Exploration in the Microstructure of Cognition : Foundations, D.E. Rumelhart and J. L. McClelland (eds.), Vol. 1, MIT Press, 1986.

4. Lippmann, R., "An Introduction to Computing with Neural Nets, " IEEE ASSP Magazine, pp. 4-22, April, 1987.

5. Kohenen, T., "An Introduction to Neural Computing, " Neural Networks, Vol. 1, pp. 3-16, 1988.

6. ----- , Self-organization and Associate Memory, Springer-Verlag, Berlin, 1984.

7. ----- , "Analysis of Self-organizing Process, "Biological Cybernetics, Vol. 44, pp. 135-140, 1982.

8. -----, Self-organization and Associative Memory, 2nd Edition, Springer -Verlag, Berlin, 1987.

9. Fukushima, K., and Miyake, S. , "Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position," Pattern Recognition, Vol. 15, pp. 455, 1982. (computer simulations that demonstrate first order invariant in a pattern set.)

10. Fukushima, K., "A neural network model for selective attention in visual pattern recognition," Biological Cybernetics, Vol. 55, pp.5-15, 1986.

11. Fukushima, K., "A Neural Nerwork for Visual Pattern Recognition," IEEE Computer, March 1988, pp. 65.

12. Guez, A., Eilbert, J., Kam, M., "Neuromorphic Architecture for Fast Adaptive Robot Control," IEEE International Conference on Neural Networks , San Diego, CA, June, 1987.

13. Guez, A., Eilbert, J., Kam, M., "Neural Network Architecture for Control," IEEE Control Systems Magazine, April 1988, pp.22.

14. Psaltis, D., Sideris, A., and Yamamura, A., "Neural Controllers," Proc. IEEE ICNN, San Diego, CA, 1987.

15. Bavarian, B., "Introduction to Neural Networks for Intelligent Control," IEEE Control Systems Magazine, Vol. 8, No. 2, April 1988, pp. 3-7.

N90-29034

A NEURAL NETWORK FOR CONTROLLING THE CONFIGURATION OF FRAME STRUCTURE WITH ELASTIC MEMBERS

Kazuyoshi Tsutsumi

Division of System Science
The Graduate School of Science and Technology
Kobe University, Rokkodai, Kobe 657, JAPAN

Abstract

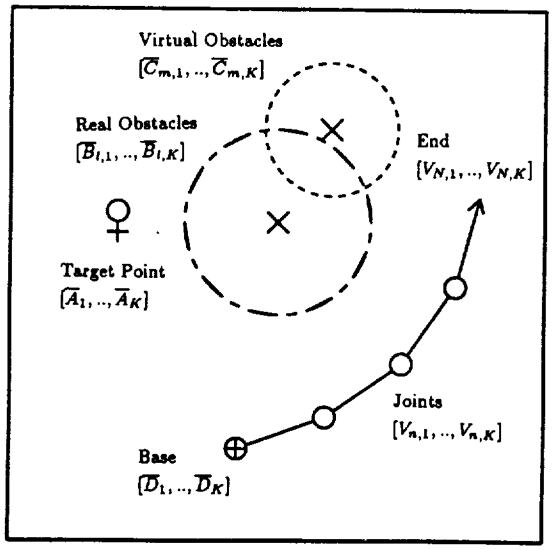
A neural network for controlling the configuration of frame structure with elastic members is proposed. In the present network, the structure is modeled not by using the relative angles of the members but by using the distances between the joint locations alone. The relationship between the environment and the joints is also defined by their mutual distances. The analog neural network attains the reaching motion of the manipulator as a minimization problem of the energy constructed by the distances between the joints, the target, the obstacles, etc. The network can generate not only the final but also the transient configurations and the trajectry. This framework with flexibility and parallelism is very suitable for controlling the Space Telerobotic systems with many degrees of freedom.

1. Introduction

In the field of Space Telerobotics, the frame structures with elastic members are paid attention as new types of robot manipulators, space cranes, etc[1]. They can be transformed to various configurations with ease. And also they can be easily disassembled / assembled and folded / unfolded. However, in general, it is very difficult to control such structure with excessive degrees of freedom. On the other hand, the applicability of neural networks to Robotics attracts a lot of researchers who produce interesting results. The possibility of parallel computation and/or the learning capability with generalization are especially emphasized as excellent characteristics of neural networks. Recently this field is expected to be in harmony with Space Telerobotics.

Tsutsumi et al. proposed a new method for controlling the configuration of the robot manipulator based on the Hopfield's neural network model[2]. According to this framework, the structure of the manipulator is modeled not by using the polar coordinate but by using the distances between the joint locations alone. The relationship between the environment and the joints is also defined by their mutual distances. The constraints, for instance, 'to keep the link lengths constant', 'to keep the safe distances between the farther joints', 'to avoid the obstacles', and 'to reach the target point', can be defined as energy terms based on the distances. By virtue of this formulation, the problem for the control resolves itself into how to find out the parameters so as to minimize the total energy. The neural network derived from the energy can control not only the ordinary manipulator with rigid links but also the elastic arm by weighting the energy terms adequately. And the processing speed does not depend on the number of degrees of freedom when the parallel analog hardware is employed.

In Sect.2, we describe the mathematical framework of the proposed method. In Sect.3, we apply it to 3-dimentional frame structure with elastic members and show the simulation results. The problems for the obstacles etc. and the future courses are discussed in the last section.


Virtual Obstacles
$[\overline{C}_{m,1},..,\overline{C}_{m,K}]$
Real Obstacles
$[\overline{B}_{l,1},..,\overline{B}_{l,K}]$
End
$[V_{N,1},..,V_{N,K}]$
Target Point
$[\overline{A}_1,..,\overline{A}_K]$
Joints
$[V_{n,1},..,V_{n,K}]$
Base
$[\overline{D}_1,..,\overline{D}_K]$

[Fig.1] The structure of the manipulator in the workspace with real and virtual obstacles.

2. Formulation and Neural Network

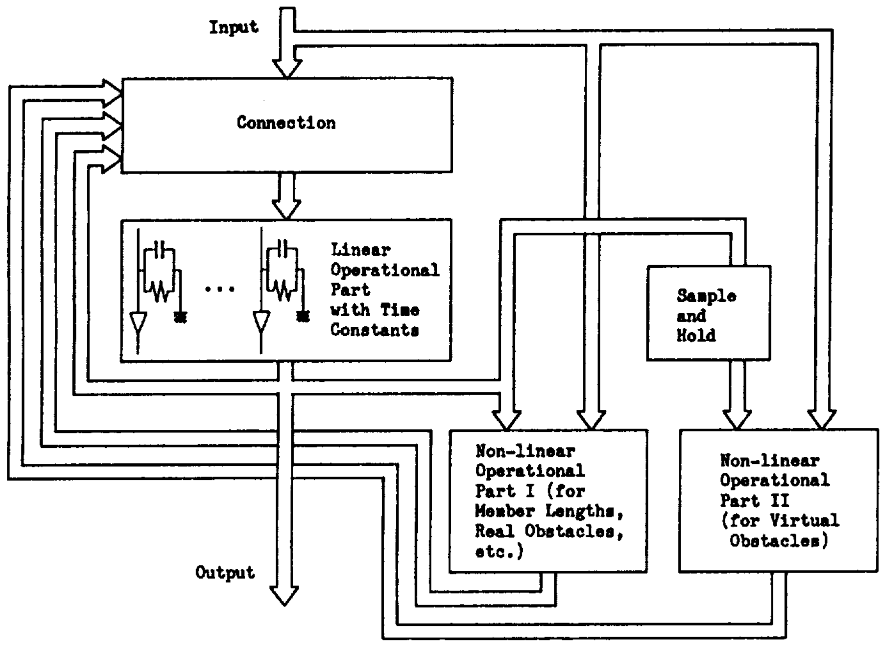
We consider an ideal robot manipulator as shown in Fig.1. The positions of the base, the movable joints, and the end are respectively given by $\overline{D}_k$, $V_{n,k}$ $(n = 1,2,..,N-1)$, and $V_{N,k}$ $(k = 1,2,..,K)$. The target point is $\overline{A}_k$ $(k = 1,2,..,K)$. L real and M virtual obstacles are assumed to be located on the position $\overline{B}_{l,k}$, $\overline{C}_{m,k}$ $(l = 1,2,..,L,\ m = 1,2,..,M,\ k = 1,2,..,K)$. The joints and the end have sensors which can get the information about the following distances:

$$\begin{aligned}
S_{n,1} &= [\textstyle\sum_{k=1}^{K}(V_{n,k}-\overline{D}_k)^2\,]^{1/2} && (n=1,2,..,N)\\
S_{n,n'} &= [\textstyle\sum_{k=1}^{K}(V_{n,k}-V_{n'-1,k})^2\,]^{1/2} && (n=2,3,..,N,\ n'=2,3,..,n,\ n\geq n')\\
P_n &= [\textstyle\sum_{k=1}^{K}(V_{n,k}-\overline{A}_k)^2\,]^{1/2} && (n=1,2,..,N)\\
Q_{n,l} &= [\textstyle\sum_{k=1}^{K}(V_{n,k}-\overline{B}_{l,k})^2\,]^{1/2} && (n=1,2,..,N,\ l=1,2,..,L)\\
R_{n,m} &= [\textstyle\sum_{k=1}^{K}(V_{n,k}-\overline{C}_{m,k})^2\,]^{1/2} && (n=1,2,..,N,\ m=1,2,..,M)
\end{aligned} \tag{1}$$

If one can find out the set of $V_{n,k}$ so that $P_N \Longrightarrow 0$ conserving the following constrains,

$$\begin{aligned}
S_{n,n} &= \overline{S}_{n,n} && (n=1,2,..,N)\\
S_{n,n'} &\geq \overline{S}_{n,n'} && (n=2,3,..,N,\ n'=1,2,..,n-1)\\
P_n &\geq \overline{P}_n && (n=1,2,..,N-1)\\
Q_{n,l} &\geq \overline{Q}_{n,l} && (n=1,2,...,N,\ l=1,2,...,L)\\
R_{n,m} &\geq \overline{R}_{n,m} && (n=1,2,...,N,\ m=1,2,...,M)
\end{aligned} \tag{2}$$

then the behaviour of the solution corresponds to the action of the robot manipulator. Here $\overline{S}_{n,n}$ represent the desirable link lengths. $\overline{S}_{n,n'}$ $(n > n')$, $\overline{P}_n$, $\overline{Q}_{n,l}$, and $\overline{R}_{n,m}$ are radii of the keep-off


Input
Connection
Linear Operational Part with Time Constants
Sample and Hold
Non-linear Operational Part I (for Member Lengths, Real Obstacles, etc.)
Non-linear Operational Part II (for Virtual Obstacles)
Output

[Fig.2] Block diagram of the analog neural network employed here.

regions characterized by the cone-type potential. Based on the constraints given by Eq.(2), we define the following energies:

$$\begin{aligned}
ES_{n,n} &= (1/2) * (S_{n,n} - \overline{S}_{n,n})^2 \\
&\qquad (n = 1, 2, .., N) \\
ES_{n,n'} &= (1/2) * F(S_{n,n'} - \overline{S}_{n,n'}) \\
&\qquad (n = 2, 3, .., N,\ n' = 1, 2, .., n-1) \\
EP_n &= (1/2) * F(P_n - \overline{P}_n) \\
&\qquad (n = 1, 2, .., N-1) \\
EP_N &= (1/2) * P_N^2 \\
EQ_{n,l} &= (1/2) * F(Q_{n,l} - \overline{Q}_{n,l}) \\
&\qquad (n = 1, 2, .., N,\ l = 1, 2, .., L) \\
ER_{n,m} &= (1/2) * F(R_{n,m} - \overline{R}_{n,m}) \\
&\qquad (n = 1, 2, .., N,\ m = 1, 2, .., M) \\
EG_{n,k} &= (1/r_{n,k}) * \int_0^{V_{n,k}} g^{-1}(V) dV \\
&\qquad (n = 1, 2, .., N,\ k = 1, 2, .., K)
\end{aligned} \tag{3}$$

$$\text{where} \quad g(x) = x \quad \text{and} \quad F(z) = \begin{cases} 0 & \text{if } z \geq 0 \\ z^2 & \text{otherwise} \end{cases}$$

Adding up these energy terms to get the total energy given by Eq.(4),

$$E = \sum_{n=1}^{N}\sum_{n'=1}^{N} ES_{n,n'} + \sum_{n=1}^{N} EP_n + \sum_{n=1}^{N}\sum_{l=1}^{L} EQ_{n,l} + \sum_{n=1}^{N}\sum_{m=1}^{M} ER_{n,m} + \sum_{n=1}^{N}\sum_{k=1}^{K} EG_{n,k} \tag{4}$$

differentiating it with respect to time, and then employing the capacitances $c_{n,k}$, we can obtain the network equations as follows:

$$\begin{aligned}
-c_{1,k}\frac{d}{dt}U_{1,k} &= (V_{1,k}-\overline{D}_k)*(1-\frac{\overline{S}_{1,1}}{S_{1,1}}) - (V_{2,k}-V_{1,k})*(1-\frac{\overline{S}_{2,2}}{S_{2,2}}) \\
&+ \sum_{n'=2}^{N}(V_{n',k}-V_{1,k})*f(1-\frac{\overline{S}_{n',1}}{S_{n',1}}) \\
&+ (V_{1,k}-\overline{A}_k)*f(1-\frac{\overline{P}_1}{P_1}) \\
&+ \sum_{l=1}^{L}(V_{1,k}-\overline{B}_{l,k})*f(1-\frac{\overline{Q}_{1,l}}{Q_{1,l}}) + \sum_{m=1}^{M}(V_{1,k}-\overline{C}_{m,k})*f(1-\frac{\overline{R}_{1,m}}{R_{1,m}}) + \frac{1}{r_{1,k}}U_{1,k}
\end{aligned}$$

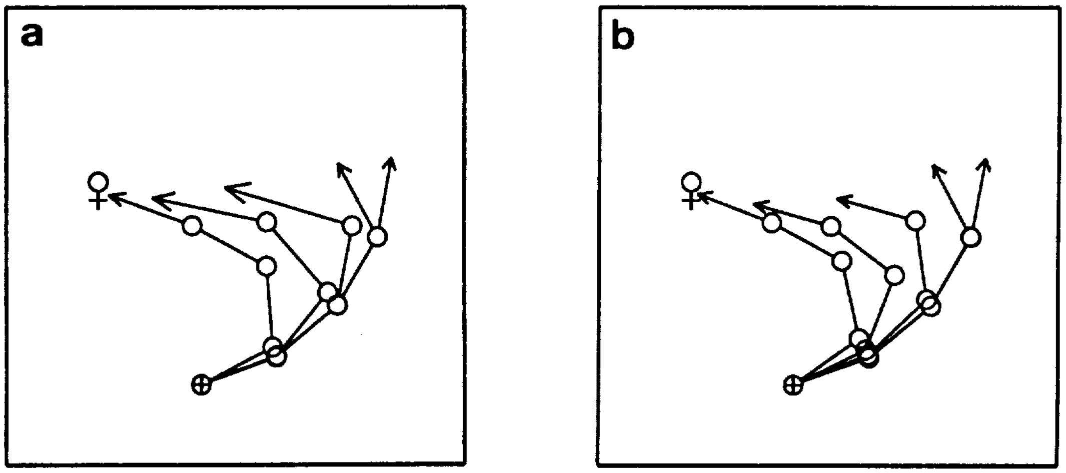
$$\begin{aligned}
-c_{n,k}\frac{d}{dt}U_{n,k} &= (V_{n,k}-V_{n-1,k})*(1-\frac{\overline{S}_{n,n}}{S_{n,n}}) - (V_{n+1,k}-V_{n,k})*(1-\frac{\overline{S}_{n+1,n+1}}{S_{n+1,n+1}}) \\
&+ \sum_{n'=1}^{n-1}(V_{n,k}-V_{n',k})*f(1-\frac{\overline{S}_{n,n'}}{S_{n,n'}}) + \sum_{n'=n+1}^{N}(V_{n',k}-V_{n,k})*f(1-\frac{\overline{S}_{n',n}}{S_{n',n}}) \\
&+ (V_{n,k}-\overline{A}_k)*f(1-\frac{\overline{P}_n}{P_n}) \\
&+ \sum_{l=1}^{L}(V_{n,k}-\overline{B}_{l,k})*f(1-\frac{\overline{Q}_{n,l}}{Q_{n,l}}) + \sum_{m=1}^{M}(V_{n,k}-\overline{C}_{m,k})*f(1-\frac{\overline{R}_{n,m}}{R_{n,m}}) + \frac{1}{r_{n,k}}U_{n,k} \\
&\qquad (n = 2, 3, ..., N-1)
\end{aligned} \tag{5}$$

$$\begin{aligned}
-c_{N,k}\frac{d}{dt}U_{N,k} &= (V_{N,k}-V_{N-1,k})*(1-\frac{\overline{S}_{N,N}}{S_{N,N}}) \\
&+ \sum_{n'=1}^{N-1}(V_{N,k}-V_{n',k})*f(1-\frac{\overline{S}_{N,n'}}{S_{N,n'}}) \\
&+ (V_{N,k}-\overline{A}_k) \\
&+ \sum_{l=1}^{L}(V_{N,k}-\overline{B}_{l,k})*f(1-\frac{\overline{Q}_{N,l}}{Q_{N,l}}) + \sum_{m=1}^{M}(V_{N,k}-\overline{C}_{m,k})*f(1-\frac{\overline{R}_{N,m}}{R_{N,m}}) + \frac{1}{r_{N,k}}U_{N,k}
\end{aligned}$$

$$\text{where} \quad U_{n,k} = V_{n,k} \quad \text{and} \quad f(z) = \begin{cases} 0 & \text{if } z \geq 0 \\ z & \text{otherwise} \end{cases}$$

Figure 2 shows the block diagram of the analog neural network represented by Eq.(5). The network has a direct feedback loop and indirect ones via non-linear operational parts. According to the same procedure as Hopfield proved, it is assured that the network output converges to a certain set of values[3].

Figure 3a shows an example of the network behaviour. The end of the manipulator can reach the target point retaining a natural posture. The links are of the same lengths in the final stage. However they lenghten in the trangent state. This is because the energy for link lengths is relatively



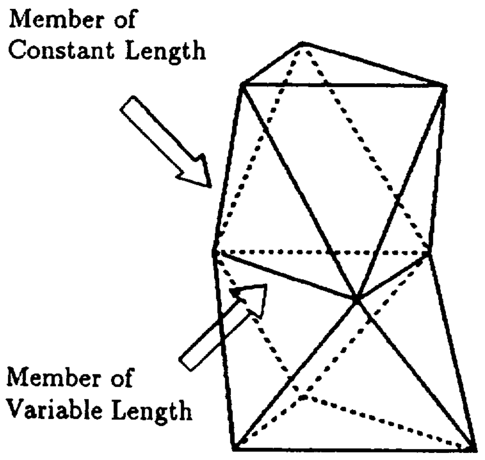
[Fig.3ab] Examples of simulation. (a) Case in which the energy terms are summed up without any weighting. (b) Case in which the adequately weighted energy terms are employed.

set smaller than the others. Figure 3b shows the case in which the energy term for link lengths is weighted severely. The link lengths in the trangent state are kept constant. Varieties of actions are caused by adjusting the balance of the energy terms alone. This means that not only rigid but also elastic types of robot manipulators can be easily represented.

3. Application to 3-Dimensional Frame Structure with Elastic Members

Tsutsumi et al. enhanced the proposed framework so as to control the configuration of the 2-dimensional truss effectively[4]. The simulation studies show that the motions for 'Rotation' and 'Stretching/Translation' are well-controlled, but 'Shrinking/ Translation' can not be attained. That is, the behaviour of the truss structure under this control method is very similar to that of spiral spring. This is because quadratic energy function is employed for the distances between the joints. Therefore the truss with curved configuration has less total energy than that with homogeneously shrinked members. In order to improve 'Shrinking/Translation', preliminarily shrinked 'Natural Position' was considered. Then introducing a 'Virtual Target Point' and stretching the truss to it, the control was assumed to start from this stretched position every time. It was named 'Initial Position'. The introduction of both short 'Natural Position' and tall 'Initial Position' gets rid of the practical shrinking motion from all modes of the behaviour and makes varieties of configurations possible.

In the present study, we further apply this framework to the variable geometry truss (VG-truss) which is a type of 3-dimensional frame structures with elastic members. Figure 4 shows the component of VG-truss conceived by Miura et al[5]. Here the lengths of the vertical members are assumed to be contant. The configuration of the truss can be changed by stretching or shrinking the horizontal members. We can formulate this structure according to the same procedure as described in Sect.2. That is, we first define the distances between the joints and those between the joints and the environment. Then we construct the energy terms based on the constraints necessary for desirable



[Fig.4] The structure of the 3-dimensional variable geometry truss.

behaviour. Differentiating the sum of the energy terms with respect to time, we can obtain the analog neural network whose block diagram is represented by Fig.2.

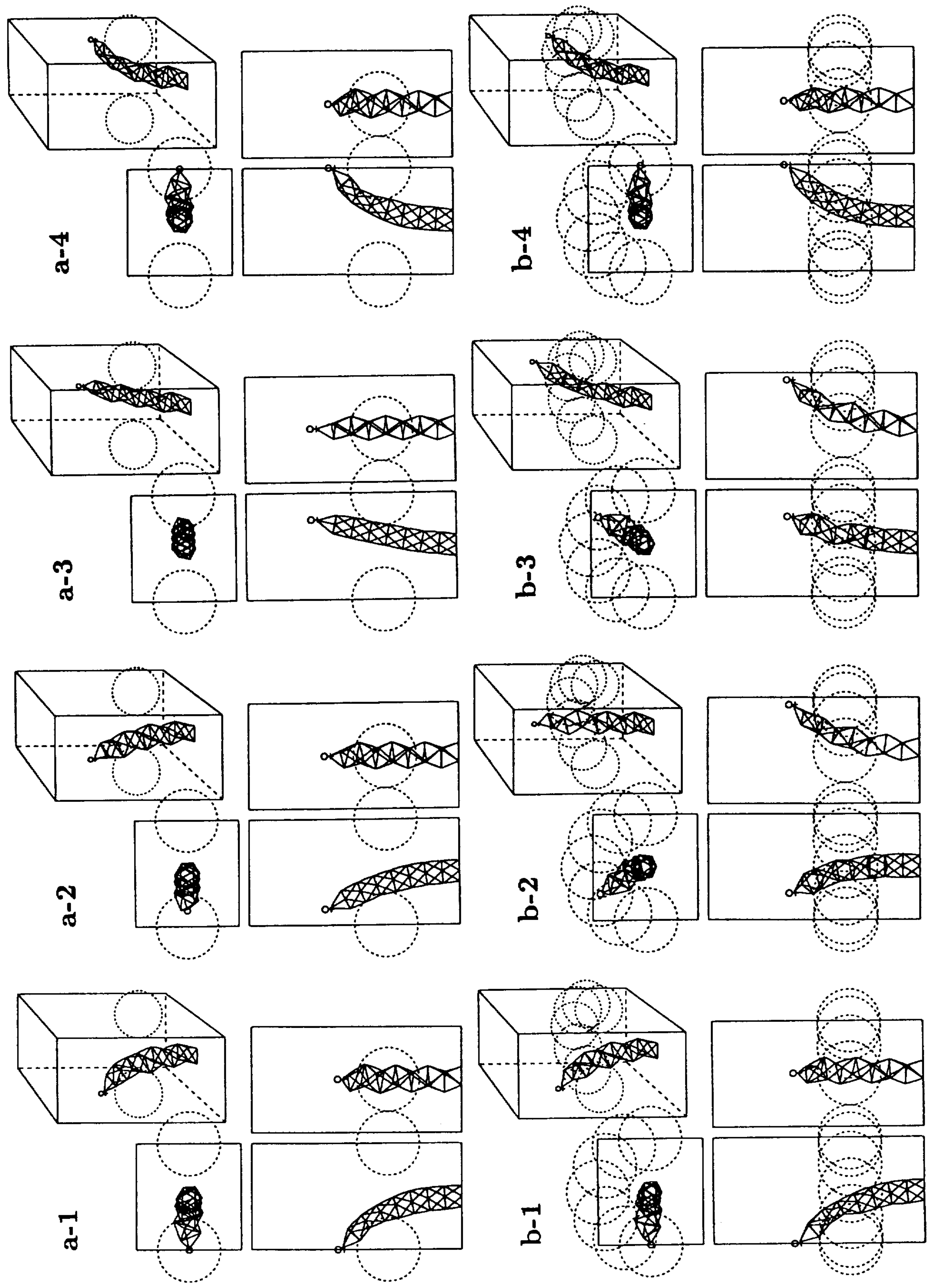
It is effective for 'Rotation / Stretching / Shrinking' motions to employ the above-mentioned concepts named 'Natural Position', 'Virtual Target Point', and 'Initial Position'. When the target point is placed at the side of the truss, the truss starts to curve and its end reaches the target point keeping the overall length of 'Natural Positon'. Figure 5a shows an example of 'Rotation' using this nature. Here we introduce virtual obstacles to improve the configurations. It is further effective to divide the action into some parts by shifting the location of 'Virtual Target Point' in turn. Figure 5b shows another example of 'Rotation'. Multiple virtual obstacles are introduced to keep the curved configuration.

Figure 6a shows the case in which the target point is placed above the truss. The end reaches to the target point. In this case, the variable members near the end tend to expand and contract more, since the energy terms are summed up without any weighting. Therefore the transient configurations are irregular as shown in the figure. The behaviour of the structure is improved sharply by employing the sum of the adequately weighted energy terms. Figure 6b shows the example. Here we change the weighting factors a few times on the way for obtaining the more desirable transient configurations.

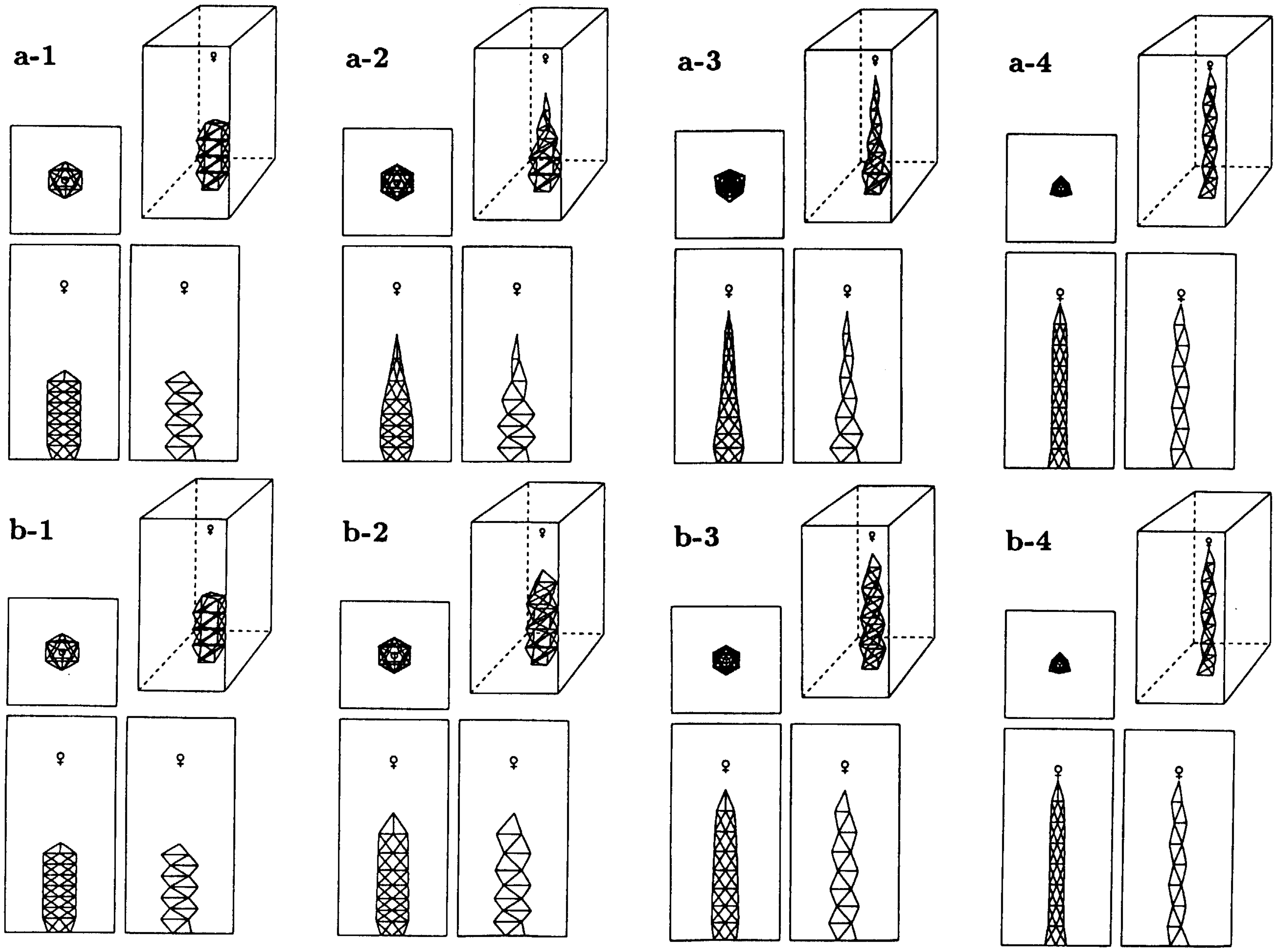
Figure 7a shows a simulation result when the target point is placed inside the truss. As already mentioned, the frame structure under this control method is poor at shrinking its member lengths homogeneously. Therefore the short 'Natural Position' is introduced here. The end can reach the target point shrinking its configuration, but the end of the truss sinks into the inside in the transient state. This is because the constraints for the distances between the farther joints are not taken into account here. It is clear that the truss can shrink its configuration successfully if such constraints are considered. Another method for better 'Shrinking/Translation' is to remove the target point temporarily. Since short 'Natural Position' is employed, the truss shrinks its overall length and it moves to 'Natural Position' unaffectedly. Figure 7b demonstrates the exmaple.

4. Discussion

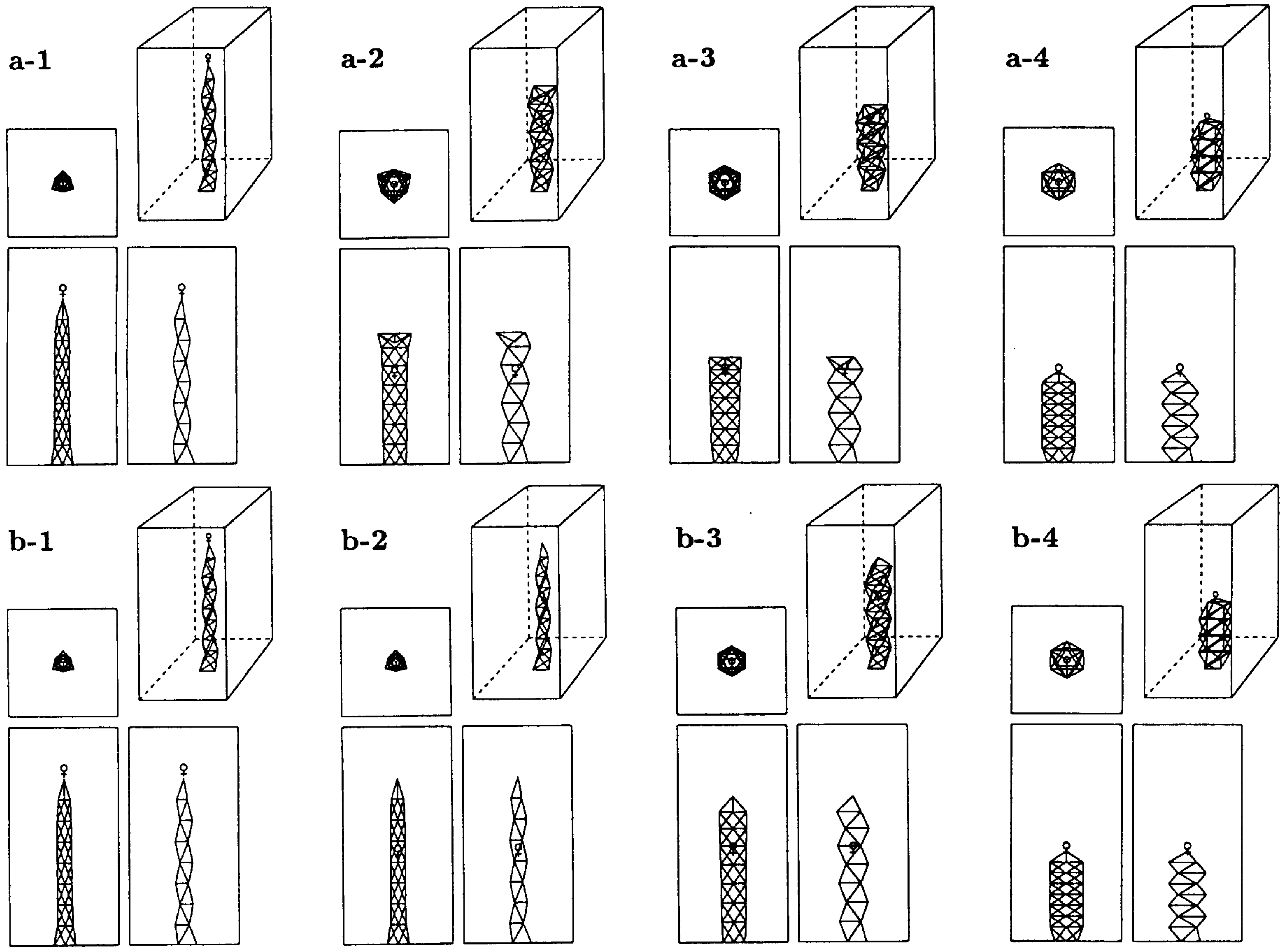
As already mentioned, the frame structure under this control method behaves just like a spiral spring. In order to solve some problems caused by this nature, we introduced the concepts named



[Fig.5ab] Rotation. (a) Case I. (b) Case II.



[Fig.6ab] Stretching/Translation. (a) Case I. (b) Case II.



[Fig.7ab] Shrinking/Translation. (a) Case I. (b) Case II.

'Natural Position', 'Virtual Target Point', 'Initial Position', and 'Virtual Obstacles'. Simulation studies demonstrate that the neural network can successfully realize the basic motions of frame structure with elastic members including 'Translation' and 'Rotation'. Although we did not exemplify 'Revolution' for want of space, it can be easily realized by using 'Virtual Target Point'. As shown in Fig.8, it is sometimes necessary for better configurations to manually change the weighting factors on the way. This is because the constraints for overall structure were not considered. We should take into account the higher-level constraints for more desirable configurations.

One of the severe problems to be solved is for the deadlock state caused by the obstacles in the workspace. Tsutsumi et al. proposed a learning strategy in which virtual obstacles are put on the deadlocked location of the end. It is very compatible with the network. However it is not a type of algorithm by which the synaptic connections are modified little by little repeating trial and error. It appears to be reasonable to divide the network into two parts, namely the one for the structure itself and the other for the environment. In this case, it will be effective to employ the mapping network with learning capability such as Backpropagation Network in order to acquire and utilize the information about the environment.

There remain a lot of future courses to be solved. However neural networks have high potentialities and they will be indispensable to Space Telerobotics. At the same time, the complicated problems in Space Telerobotics will give us good hints for modelling the neural networks.

Acknowledgement

The author wishes to thank Prof. Haruya Matsumoto and Prof. Shinzo Kitamura for their encouragement and Mr. Koichi Katayama for his assistance.

References

[1] Y.Seguchi, M.Tanaka, Y.Sasabe, H.Tsuji, and K.Okamura, "Evolution of Mast-type Truss to Robot Arm", Preprints of Japan-U.S.A. Symposium on Flexible Automation, pp.251-259, 1986

[2] K.Tsutsumi and H.Matsumoto, "Neural Computation and Learning Strategy for Manipulator Position Control", Proc. of IEEE First Annual International Conference on Neural Networks (ICNN-87), Vol.IV, pp.525-534, 1987

[3] D.W.Tank and J.J.Hopfield, "Simple 'Neural' Optimization Networks: A A/D Converter, Signal Decision Circuit, and Linear Programming Circuit", IEEE Trans. of CAS, Vol.33, No.5, pp.533-541, 1986

[4] K.Tsutsumi, K.Katayama, and H.Matsumoto, "Neural Computation for Controlling the Configuration of 2-Dimensional Truss Structure", Proc. of IEEE ICNN-88, Vol.II, pp.575-586, 1988

[5] K.Miura, H.Furuya, and K.Suzuki, "Variable Geometry Truss and its Application to Deployable Truss and Space Crane Arm", Acta Astronautica, Vol.12, No.7/8, pp.599-607, 1985

FUNDAMENTAL AI RESEARCH

N90-29035

Coordinating the Activities of a Planner and an Execution Agent

Austin Tate

Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom

Abstract

We have defined a research programme that will explore the link between planning and execution systems. A simple scenario has been defined in which we have a very capable off-line planning system interacting with the user and a smaller, less capable, on-line real-time system executing plans and reacting to faults. However, the on-line execution system may have a more flexible representation of the plans it is executing. This imbalance in the capabilities of the two "agents" involved should clarify some of the research objectives and give us an experimental framework for our work. Our task is to investigate the knowledge representations and communication protocols needed to link a user stating some requirements for a task to be carried out through a planning system to the (remote) execution agent that can carry out the user's wishes.

We are starting from the notion that a single representation can encapsulate the expression of the user's requirements, the capabilities for action, the communication to the execution agent, the successful or faulty response from the execution agent and the means of keeping the user informed. This is based on our work on *Goal Structure*, which captures the intent of plan steps; *Task Formalism*, a declarative input language for the planners we have built at Edinburgh; and on the definition of a *Plan State*.

Methods of creating *plan patches* to update the plans separately held by each of the parties involved to keep them in step as they each react to changing circumstances in real-time will be investigated. This will involve the specification of plan patch *attachment points* that can be understood by the recipient. We will also investigate transaction based methods for coordinating the activities of the planner with those of the execution agent and user.

The trial application area for the research is in the command and control of an advanced Earth Observation Space Platform.

1. Introduction

We have just embarked on a new phase in our research concerned with the application of Artificial Intelligence (AI) planning techniques to the area of spacecraft command and control. In particular, the research addresses the issue of closing the loop between plan generation and execution monitoring in the face of simple plan failures.

1.1. Planning and Executing: Closing the Loop

In this project, we intend to make use of our experiences in dealing with applications of AI planning techniques to practical projects (with O-Plan, Currie and Tate, 1985) and to develop a planning system that closes the loop between planning and executing. There have been some successes with previous attempts at closing the loop (Fikes, Hart and Nilsson (1972), Wilkins (1985), Malcolm and Smithers (1988), Drabble (1988)), but often the plans generated were rather limited and not very flexible. In general, the complexities of the individual tasks of plan representation, generation, execution monitoring and repair has led to research into each of these issues separately. In particular, there is now a mismatch between the scale and capabilities of plan representations proposed for real-time execution systems (Georgeff and Lansky (1986), Nilsson (1988), Rosenschien and Kaelbling (1987)) and those that can be generated by today's AI planners.

However, the demand is for a system that can take a command request, generate a plan, execute it and react to simple failures of that plan, either by repairing it or by re-planning. Explicit knowledge about the structure of the plan, the contribution of the actions involved and the reasons for performing plan modifications at various stages of the plan construction process, provides us with much of the information required for dealing with plan failures. Such knowledge is also essential for further planning and re-planning by identifying generalisations or contingencies that can be introduced into the plan in order to avoid similar failures.

1.2. Planning with semi-autonomous agents

Most planners to date have constructed their plans with full knowledge of the capabilities of the devices under their control. Thus, executing such plans involves the direct application of the operators within the plan by an execution agent which has no planning capability. Invariably, unforseen events will occur causing failure of the current plan and a request for repair of the plan or re-planning directed at the planning system. Building into the execution agent some ability to repair plans and to perform re-planning would improve the problem solving performance of the execution agent, especially when it is remote from the central planning system.

The scenario we intend to investigate is as follows. A central planner plans to achieve a task described at a high-level of abstraction. The central planner has knowledge of the general capabilities of a semi-autonomous execution agent but does not need to know about the actual operators that execute the actions required to carry out the desired task. The execution agent executes the plan by choosing the appropriate operators to achieve the various sub-tasks within the plan, using its knowledge about the particular devices under its control. Thus, the central planner communicates a general plan to achieve a particular task, and responds to failures fed back from the execution agent which are in the form of flaws in the plan. The execution agent communicates with the real world by executing the operators within the plan and responding to failures fed back from the real world. Such failures may be due to the inappropriateness of a particular operator, or because the desired effect of an operator was not achieved due to an unforseen event. The reason for the failure dictates whether the same operator should be re-applied, replaced with other operators or whether re-planning should take place.

1.3. The Role of Dependencies in Plans

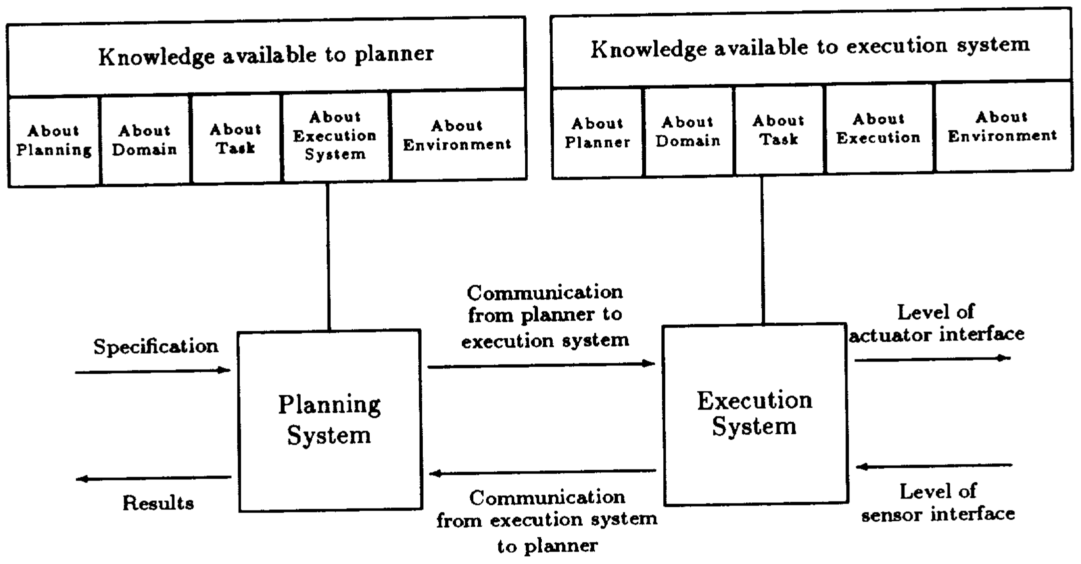
The use of dependencies within planning promise great benefits for the overall performance of a planning system particularly for plan representation, generation, execution and repair.

Early work on Decision Graphs by Hayes (1975) at Edinburgh has shown how the explicit recording of the decisions involved in the planning process could be used for suggesting where and how much re-planning should take place when unforeseen situations invalidate the success of the current plan. Some work to link these ideas with a non-linear AI planner was undertaken during the mid 1970s by Daniel (1983).

The notion of the teleology of a plan, which we call the Goal Structure (Tate, 1977), refers to the dependencies between the preconditions and postconditions of operators involved in the plan. Although, such dependencies have been shown to be useful for describing the internal structure of the plan and for monitoring its execution (Fikes, Hart and Nilsson (1972), Tate (1984)), there has been no comprehensive discussion of their use in all aspects of plan generation, execution monitoring and plan repair.

2. Planning and Execution Architecture

Recently, we have been promoting a common representation for the input/output requirements and capabilities of a planner and execution agent. This supports the representation of the communication between a user, requesting the plan, and the real world, in which the plan is being executed. Such communication may take place either directly through a planner or indirectly via a central planner and a dumb or semi-autonomous execution agent. In the latter case, the communication between the central planner and the execution agent becomes an interesting research issue.


Knowledge available to planner
About Planning
About Domain
About Task
About Execution System
About Environment
Knowledge available to execution system
About Planner
About Domain
About Task
About Execution
About Environment
Specification
Results
Planning System
Communication from planner to execution system
Communication from execution system to planner
Execution System
Level of actuator interface
Level of sensor interface

The common representation includes knowledge about the capabilities of the planner and execution agent, the requirements of the plan and the plan itself either with or without flaws. See the diagram above. Thus, a planner will respond to the requirements of a user. Based on the knowledge of its own capabilities and that of the execution environment, it will generate a plan. This plan may then be executed directly in the real world, or, indirectly via an execution agent. The execution agent executes this plan in the real world and monitors the execution, responding to failures in one of two ways. If it does not have knowledge of its own capabilities, it simply returns knowledge of the failure to the central planner and awaits a revised plan to be sent. In this case, the execution agent is dumb. If it does have knowledge of its own capabilities, it may attempt to repair the plan and then continue with execution. On the other hand, if a repair is beyond the capabilities of the execution agent, then this knowledge is fed back to the central planner and again a revised plan is expected. In this case, the execution agent is semi-autonomous. When failures during the application of the plan are fed back to the planner, these may be acted upon by it and a repair of the plan made or total re-planning instigated. This may, in turn, involve the user in reformulating the task requirement. A revised or new plan is then executed. Finally, success of the execution or partial execution of the plan is fed back to the user.

Other issues relating to the choice of the common representation and communication protocols include:

- when to repair the plan or seek re-planning,
- continuing execution of parts of a plan, not affected by the failure,
- continuing to maintain a safe execution state even while awaiting initial commands or the correction of faults in earlier plans,
- maintaining integrity and synchronisation of communicated plans and flaws.

3. Plan States

The O-Plan (Currie and Tate, 1985) Plan State holds a complete description of a plan at some level of abstraction. The Plan State contains a list of the current *flaws* in the plan. Such flaws could relate to abstract actions that still must be expanded before the plan is considered valid for passing on for execution, unsatisfied conditions, unresolved interactions, overcommitments of resource, time constraint faults, etc. The Plan State can thus stand alone from the control structure of the AI planner in that it can be saved and restored, passed to another agent, etc.

At any stage, a Plan State represents an abstract view of a set of actual plans that could be generated within the constraints it contains. Alternative lower level actions, alternative action orderings and object selections, and so on are aggregated within a high level Plan State description.

The O-Plan *Task Formalism (TF)* is a declarative language for expressing action schemata, for describing task requests and for representing the final plan. Our design intention for O-Plan is that a Plan State can be created from a TF description and vice versa. This has not quite been achieved in the existing O-Plan prototype (Currie and Tate, 1989), but this remains our goal. The aim is that the AI planner can take a Plan State as a requirement (created by a TF Compiler from the user provided task specification in TF) and can use a library of action schemata or generic plan state fragments (themselves created by the TF Compiler from a domain description provided by the user) to transform the initial Plan State into one considered suitable for termination. This final Plan State could itself be decompiled back into a

TF description if required.

In practice, the O-Plan architecture is designed for operation in an environment where the ultimate aim of termination will not be achieved. There will be new command requests arriving and earlier ones being modified, parts of plans will be under execution as other parts are being elaborated, execution faults are being handled, etc.

The Plan State cannot contain arbitrary data elements. The AI planner is made up of code that can interpret the Plan State data structure and interpret the lists of flaws in such a way that it can select from amongst its computational capabilities and its library of domain specific information to seek to transform the current Plan State it is given into something that is desired by the overall architecture. This is defined as the reduction of the list of flaws known to the planner. The O-Plan architecture associates a Knowledge Source with each flaw type that can be processed (Currie and Tate, 1985). An agenda of outstanding flaws is maintained in a Plan State and appropriate Knowledge Sources are scheduled on the basis of this.

We believe that the basic notions described above can serve us well as a basis for an attack on the problem of coordinated command, planning and execution in continuously operating domains. We will explore the new properties that we must seek from our basic notions in the following sections.

4. Plan Patches

The requirement for asynchronously operating planners and execution agents (and indeed users and the real world) means that it is not appropriate to consider that a plan requirement is set, passed on for elaboration to the planner and then communicated to a waiting execution agent which will seek to perform the actions involved. Instead, all components must be considered to be operating already and maintaining themselves in some stable mode where they are responsive to requests for action from the other components. For example, the execution agent may have quite elaborate local mechanisms and instructions to enable it to maintain a device (say a spacecraft or a manufacturing cell) in a safe, healthy, responsive state. The task then is to communicate some change that is requested from one component to another and to insert an appropriate alteration in the receiver such that the tasks required are carried out.

We propose to define a *Plan Patch* as a modified version of the type of Plan State used in O-Plan. It would have some similarity to an operator or action schema given to an AI planning system in that it would be an abstracted or high level representation of a part of the task that is required of the receiver using terminology relevant to the receiver's capabilities. This would provide a simplified or black-box view of possibly quite detailed instructions needed to actually perform the action (possibly involving iterators and conditionals, etc). Complex execution agent representational and programming languages could be handled by using this abstracted view (e.g., Georgeff and Lansky (1986), Nilsson (1988)). For example, reliable task achieving *behaviours* which included contingencies and safe state paths to deal with unforseen events could be hidden from the planner by communication in terms of a simplified and more robust model of the execution operations (Malcolm and Smithers, 1988).

Outstanding flaws in the Plan Patch would be communicated along with the patch itself. However, these flaws must be those that can be handled by the receiver.

It can be seen that the arrangement above (mostly assumed to refer to the communication between a planner and execution agent) also reflects the communication that takes place between a user and the planner in an O-Plan type AI planner. Requiring rather more effort will be the investigation of suitable Plan Patch constructs to allow execution errors to be passed back to the planner or information to be passed back to the user, but we believe that this is a viable objective.

5. Plan Patch Attachment Points

There is a need to communicate the points at which the Plan Patch should be attached into the full Plan State in the receiver. The sender and receiver will be operating asynchronously and one side must not make unreasonable assumptions about the internal state of the other.

We intend to endow all the components with a real-time clock that can be assumed to be fully synchronised. We will also make simplifying assumptions about delays in communication to keep to the immeditate problem we are seeking to tackle (while fully believing that extension to environments where communication delay is involved will be possible). Therefore, metric time will be the "back-stop" as a means of attaching a Plan Patch into the internal Plan State of the receiver. Metric time will also be important to start things off and to ensure a common reference point when necessary (e.g., in cases of loss of control).

However, the use of metric time as an attachment point lacks flexibility. It gives the receiver little information about the real intentions behind the orderings placed on the components of the Plan Patch. It will, in some cases, be better to communicate in a relative or qualified way to give the receiver more flexibility. Suitable forms of flexible Plan Patch Attachment Point description will be investigated. Initial work will centre on descriptions relative to the expected Goal Structure (Tate, 1977) of the receiver.

6. Incremental Plan States

Our approach will be to combine the ideas above to define an *Incremental Plan State* with three components:

1. a plan patch,
2. plan patch flaws as an agenda of pending tasks,
3. plan patch attachment points.

Such Incremental Plan States will be used for two way communication between the user and the planner and between the planner and the execution agent. Our current Plan State structures and flaw repertoire will be extended to cope, initially, with a dumb execution agent that can simply dispatch actions to be carried out and receive fault reports against a nominated set of condituions to be explicitly monitored (as described in Tate, 1984). Later in the research programme, the Plan State data structures and flaw repertoire will be extended again to cope with a semi-autonomous execution agent with some capability to further elaborate the Incremental Plan States and to deal locally with re-planning requirements.

A means to compile an Incremental Plan State from a modified type of Task Formalism (TF) declarative description (and vice versa) will be retained.

7. Plan Transactions

The overall architecture must ensure that an Incremental Plan State can be understood by the receiver and is accepted by it for processing. This means that all the following are understood by the receiver:

1. plan patch description is clear,
2. plan patch flaws can be handled by receiver's Knowledge Sources,
3. attachment points understood.

It is important that the sender and receiver (whether they are the user and the AI planner, the planner and the execution agent, or one of the reverse paths) can coordinate to send and accept a proposed Incremental Plan State which the receiver must assimilate into their own Plan State. We proprose to use *transaction processing* methods to ensure that such coordination is achieved.

We expect to create some specific flaw types and Knowledge Sources in the various components (user interface, AI planner and execution agent) to handle the extraction and dispatch (as an Incremental Plan State) of a part of an internal Plan State in one component, and the editing of such an Incrememtal Plan State into the internal Plan State of the receiver. The "extraction" Knowledge Sources must be supplied with information on the Plan Patch description, flaw types and attachment points that the receiver will accept. This constitutes the primary source of information about the capabilities of the receiver that the sender has available and its representation will be an important part of the research.

Communication guards will ensure that the *a priori* criteria for acceptance of an Incremental Plan State for processing by the receiver's Knowledge Sources are checked as part of the Plan Transaction. It may also be the case that initial information about urgency will be able to be deduced from this acceptance check to prioritise the ordering of the new flaws with respect to the existing entries on the agenda in the receiver.

8. Application to Spacecraft Command and Control

Spacecraft command and control provides a realistic target domain for looking at planning and execution, in particular, for planning with semi-autonomous agents. The desire to improve the autonomous capabilities of a spacecraft is apparent, especially for spacecraft in communication with an earth-based segment, such as for a deep space probe in communication with its mission control or for a satellite with its ground station, or, indeed, for a space station that is controlling various on-board devices such as robot arms and orbital manoeuvring vehicles.

The NASA Jet Propulsion Laboratory Deviser planner which was applied to the task of generating command sequences for the Voyager spacecraft (Vere, 1983) was based on our earlier work on the Nonlin plannner (Tate, 1977). Recently, AIAI has had a team of people who have worked on the application of Knowledge-Based Planning and other Knowledge-Based Systems techniques to the area of spacecraft command and control. This has been funded by the European Space Agency for ERS-1 scheduling (Fuchs et al., 1988) and by the UK Science and Engineering Research Council for work on a technology proving satellite T-SAT (Drummond, Currie and Tate (1987, 1988), Fraser et al (1988)).

The investigation of planning and execution using the approach described above will be carried out in the context of an application to spacecraft command and control using data from a system such as ERS-1 or the Polar Platform segment of the International Space Station. We are alert to the possibility of viewing our techniques as being relevant to the process of *Task Amplification* whereby a user's commands can be interpreted via a planner and an intelligent execution monitor in a teleoperations environment.

9. Next Steps

The description in this paper of our approach to the integration of planning and execution is based on ideas and techniques that have been developed over a significant period of time. A number of important building blocks are now seen to be in place for a concerted effort to construct such an integrated command and control system able to operate in a realistic application domain.

We believe that the simplifications we have made and the differentiation of the nature of the experimental planning and execution environments we will explore will assist in clarifying our research approach. However, we believe that the architecture should be quite general if we are successful. We expect that one advantage of the line of attack we propose will be the ability to deal with large scale realistic execution environments of the type now being developed by other researchers.

10. Acknowledgements

The work of the Knowledge Based Planning group at AIAI in Edinburgh has is supported by the Science and Engineering Research Council (Alvey Directorate grant GR/D/5898-7 An Architecture for Knowledge Based Planning and Control and SERC grant GR/E/98058 Knowledge Based Command and Control of a Spacecraft). Members of the Knowledge Based Planning Group at AIAI (especially Ken Currie and Roberto Desimone) helped formulate the approach described in this paper and assisted with its preparation. The support of SD-Scicon and the United States Air Force for the work of the group is gratefully acknowledged.

11. References

Fuchs, J., Guldberg, J., Olalainty, B. and Currie, K.W. (1988) "Expert Planning System for a Space Application" Workshop on AI Applications for Space Projects, ESTEC, European Space Agency.

Currie, K.W. and Tate, A. (1985) "O-Plan: Control in the Open Planning Architecture", Proceedings of the BCS Expert Systems 85 Conference, Warwick, UK, Cambridge University Press.

Currie, K.W. and Tate, A. (1989) "O-Plan: the Open Planning Architecture", to appear.

Daniel, L. (1983) "Planning and Operations Research" in Artificial Intelligence: Tools, Techniques and Applications (eds.) O'Shea and Eisenstadt, Harper and Row, New York.

Drabble, B. (1988) "Intelligent Execution Monitoring and Error Analysis in Planning involving Processes", Ph.D. Thesis, University of Aston in Birmingham.

Drummond, M.E., Currie, K.W. and Tate, A. (1987) "Contingent Plan Structures for Spacecraft" Proceedings of the NASA Workshop on Space Station Telerobotics, JPL Publication 87-13 Vol III, pp. 17-25 (ed.) G. Rodriguez. Jet Propulsion Laboratory, California, USA, January 1987.

Drummond, M.E., Currie, K.W. and Tate, A. (1988) "O-Plan meets T-SAT: First results from the application of an AI Planner to spacecraft mission sequencing", AIAI-PR-27, AIAI, University of Edinburgh.

Fikes, R.E., Hart, P.E. and Nilsson, N.J. (1972) "Learning and Executing Generalized Robot Plans", Artificial Intelligence Vol. 3.

Fraser, J., Conway, S.M., Currie, K.W., Drummond, M.E., Lockwood, R.M., Macintosh, A.L. and Tate, A. (1988) "Using on-board AI to increase spacecraft autonomy", International Conference on Human-Machine Interaction and Artificial Intelligence in Aeronautics and Space, Toulouse, France, September 1988.

Georgeff, M. P. and A. L. Lansky (1986) "Procedural Knowledge", in Proceedings of the IEEE Special Issue on Knowledge Representation, Vol. 74, pp 1383-1398.

Hayes, P.J. (1975) "A representation for robot plans", IJCAI-75, Proceedings of the International Joint Conference on Artificial Intelligence, Tbilisi, USSR.

Malcolm, C. and Smithers, T. (1988) "Programming Assembly Robots in terms of Task Achieving Behavioural Modules: First Experimental Results", in Proceedings of the Second Workshop on Manipulators, Sensors and Steps towards Mobility as part of the International Advanced Robotics Programme, Salford, UK.

Nilsson, N.J. (1988) "Activity Networks", Proceedings of the Rochester Planning Workshop, October 1988.

Rosenschein, S.J., and Kaelbling, L.P. (1987) "The Synthesis of Digital Machines with Provable Epistemic Properties" SRI AI Center Technical Note 412.

Tate, A. (1977) "Generating Project Networks", IJCAI-77, Proceedings of the International Joint Conference on Artificial Intelligence, pp 888-893, Cambridge, Mass., USA.

Tate, A. (1984) "Planning and Condition Monitoring in a FMS", Proceedings of the International Conference on Flexible Automation Systems, Institute of Electrical Engineers, London, UK.

Vere, S.A. (1983) Planning in Time: Windows and Durations for Activities and Goals. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 3, May 1983, pp 246-267.

Wilkins, D.E. (1985) "Recovering from execution errors in SIPE", Computational Intelligence Vol. 1 pp 33-45.

N90-29036

Plan Recognition for Space Telerobotics

Bradley A. Goodman[1]
BBN Systems and Technologies Corporation
10 Moulton Street
Cambridge, MA 02138

Diane J. Litman
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Abstract

Current research on space telerobots has largely focussed on two problem areas: executing remotely controlled actions (the "tele" part of telerobotics) or planning to execute them (the "robot" part). This work (see [Rodriguez, ed. 87]) has largely ignored one of the key aspects of telerobots: the interaction between the machine and its operator. For this interaction to be felicitous, the machine must successfully understand what the operator is trying to accomplish with particular remote-controlled actions. Only with the understanding of the operator's *purpose* for performing these actions can the robot intelligently assist the operator, perhaps by warning of possible errors or taking over part of the task. We motivate the need for such an understanding in the telerobotics domain and describe an intelligent interface we are developing in the chemical process design domain that addresses the same issues.

1 Telerobotic Interfaces

Space telerobots are being developed to increase astronaut's productivity by allowing them to remotely perform cost-effective assembly, servicing and maintenance of equipment in outer space [Bejczy 87, Jenkins 87]. The telerobot extends the operator's abilities into the hostile environment of outer space while the operator stays in a non-hostile location. While the long-term goal is for autonomous telerobots that can perform their duties with little human intervention, current research is directed towards an active role of the human operator in the guidance of the telerobot. The operator monitors closely the activity of the robot through video and sensor information and issues instructions directly, through manipulator controls, and indirectly through prestored task plan data.

One problem with such a paradigm is the complexity of the required telerobotic interface. The operator must contend with limited resources and environmental issues such as imprecise video or sensor data, communication time delays, and the parallel coordination of numerous telerobot resources (e.g., the control of a mechanical arm and its numerous degrees of freedom [Bejczy 87]). An interface that can smoothly handle such issues in real time is beyond current technology. An operator instead must break the task down into reasonable self-contained pieces and separately execute actions to achieve each piece. For example, consider a telerobot having two arms with cameras mounted on each hand. Now suppose the operator is manipulating one arm to perform a task but would also like to move the second arm so that its camera can provide a view of the first arm from the side. An operator might find it quite awkward to manipulate both arms and cameras at the same time. If, however, the telerobot system could recognize the operator's goals and, seeing that the second arm was idle, infer that a view from it would be helpful to the operator, it could automatically move the second arm to track the first arm until the operator attempted to manually control it.

We propose that an intelligent telerobotics interface could address such problems and provide the operator with a more robust and habitable environment for controlling the robot. Development of such an interface requires adding another layer to the interface control mechanism that attempts to surmise the purpose of the operator's actions and, from those inferences, determine (1) more precise specifications for fine-tuning an operator's action and (2) helpful actions that the telerobotic system can independently perform to assist the operator and the telerobot in the performance of the task. Such an interface requires a mechanism, which we call a "plan recognizer," to infer the operator's intentions, and a response planner, to determine from the operator's intentions, the operator's actual actions, the telerobotic system's knowledge base, and sensor data appropriate ways of responding (e.g., adjusting an

[1]Partial support for this research was provided by the National Science Foundation under grant IRI-8701874.

operator's commands, by filling in more detailed information, or performing on its own initiative non-catastrophic actions). Figure 1 below illustrates the interface we envision. Notice that the plan recognizer has been added to the typical telerobot interface between the operator and the response planner where it can deduce an operator's goals and fine-tune an operator's commands. In this paper we concentrate only on plan recognition; others (e.g., [Dean and Brody 87], [Drummond, Currie, and Tate 87], [Durfee and Lesser 87], [Georgeff, Lansky, and Schoppers 87], and [Wilkins 87]) are considering response planning. Below we describe plan recognition and some of its recent instantiations and then we lay out a more robust formulation, constructive plan recognition, that we are attempting to employ. We describe as a case study how plan recognition has improved a chemical process design interface that we have built. Finally, we close with a brief discussion on how a similar interface could benefit telerobotics.

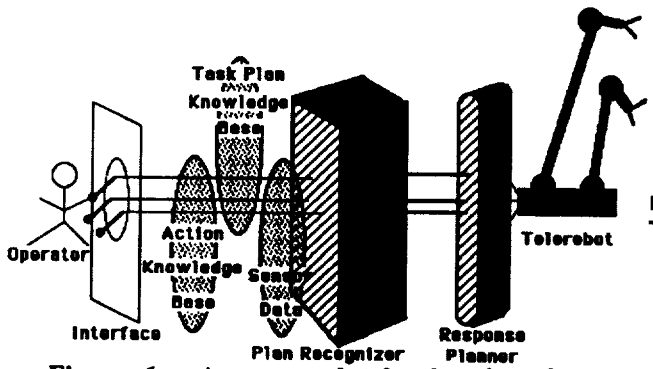


Figure 1: A proposed telerobot interface

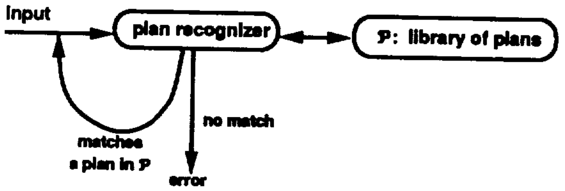


Figure 2: Standard plan recognition

2 Plan Recognition

2.1 Standard Plan Recognition

In the artificial intelligence literature, the issues mentioned in the previous section are addressed by work on plan recognition (e.g., [Allen 83], [Carberry 85], [Carver, Lesser, and McCue 84], [Genesereth 79], [Kautz 87], [Litman and Allen 87], [Pollack 86], [Schank and Reiger 74], [Schmidt, Sridharan, and Goodson 78], [Sidner 85]). Plan recognition is the process of inferring a user's intentions from his or her utterances or commands. In other words, since people don't normally maneuver in the world haphazardly and they usually have a purpose in mind, one wants to recognize the plan that underlies their actions A plan recognition component attempts to understand a sequence of observed input actions as accomplishing some high-level goal. Although these techniques have been largely used to understand the intent underlying linguistic actions (spoken or written utterances), they also lay a solid foundation for analyzing the purpose behind non-linguistic actions, such as teleoperated manipulations.

The first plan recognizers grew out of research on "scripts" [Schank and Reiger 74] for modelling stories or conversations. These early predecessors to plan recognizers were useful only if nothing out of the ordinary occurred. A richer formalism is necessary to infer the proper connections when an unusual or unexpected event occurs. Plan-based models have been employed to do just that. A plan-based system attempts to construct an explanation for the observations. Plan recognition, thus, becomes the process of searching the space of possible plans for an explanation. Since such a search is not tractable in any but the smallest domains, simplifying assumptions must be added to the plan recognition problem [Allen 87]. The typical assumption, called the "completeness assumption," has been to assume complete knowledge of all possible plans.

Several strategies have recently been employed to recognize and track a user's plan (e.g., Allen 83, Sidner 85, Carberry 85, Kautz 87). These plan recognition strategies typically use a plan library: a description of typical plans and actions that might occur in a particular domain. On the basis of the library and description of an action, plan recognition algorithms produce (possibly partial) descriptions of the corresponding plan. Allen's algorithm (Allen 83) uses a heuristic search to choose a preferred plan, while others (such as Sidner's (85) or Kautz's (87)) allow the recognition to occur incrementally. Sidner's and Kautz's plan recognition techniques operate in a "syntactic" framework. They perform a process similar to parsing by attempting to fit observed user actions into an expected user plan (as defined by a joint library of plans or goals). A successful parse of a user's observed actions, thus, entails finding an exactly matching prestored plan in the plan library. This strategy works given the assumption that the library is complete.

There are weaknesses to this approach. First, the observed user's actions may not match any of the plans in the plan library. Current algorithms fail in such a situation (as shown in Figure 2). Second, the parsing algorithms currently employed are slow making them less appropriate for real-time actions.

Our own research has been concerned with extending plan recognition to non-linguistic domains.[2] We have had, among others, two concerns relevant to space telerobots: (1) supporting real-time plan recognition, and (2) allowing for the recognition of novel plans.

2.2 Extended Plan Recognition for Space Telerobotics

The telerobot domain does not fit the current plan recognition paradigm described above. All possible plans of the operator or tasks the telerobot can achieve cannot be prestored ahead of time. Conditions in which tasks are undertaken can vary greatly and thus cannot be prepared for ahead of time; unexpected contingencies can occur. Hence user plans are often novel, created on the fly, or can have (unanticipated) errors in them. Given the hostile environment in which telerobots are intended to work, plan recognition must occur promptly in near real time before conditions change drastically. We discuss below a proposal for extending plan recognition so that it can function in a helpful manner in a telerobot domain. We describe a new view of plan recognition to allow it to respond to novel situations. While we do not discuss how to formulate plan recognition to occur in real time, we anticipate that a formulation based on chart parsing [Ross and Lewis 87, Vilain 88] could provide us with the necessary speed and flexibility.

2.2.1 Constructive Plan Recognition

Our proposal for constructive plan recognition is motivated by some early research in plan recognition. We describe that work briefly in the next section.

Motivating work

Plan recognition has primarily been investigated in the past at three different levels: formal, implementational, and psychological. Recently there have been numerous interesting theoretical results in the formal studies of plan recognition (Cohen and Levesque 85, Kautz 87), just as there have been interesting results about grammars independent of any particular grammar. We cannot, however, simply transfer the plan recognition algorithms developed from the formal studies and use them as implementations without first considering the domain to which they are to be applied. Instead, we propose looking further back to the early pioneering work in plan recognition (i.e., Schmidt, Sridharan, and Goodson 78, Genesereth 79, and Allen 83). Those works typically had particular applications in mind (e.g., Allen (83) wanted to provide helpful responses in a natural language understanding system). They also were not governed by the completeness assumption on the plan library. The analysis by Schmidt et al. is heavily motivated by psychological modelling and uses causality instead of (implicitly) prestored "legal" action sequences (as in the "syntactic" approaches to plan recognition) to tie together observed actions. After each observed action, a small number of alternative plan hypotheses are generated and used along with knowledge about the domain to generate expectations concerning what actions to expect next. Should the expectations not be satisfied, a revision process is employed to reformulate the current hypothesis. Allen developed a plan inference system that also chains together actions via decompositions and effects. His algorithm incorporates a set of plan recognition rules and a heuristic control strategy. The rules determine which inferences are possible. They define a valid chain of inferences from observed actions to plausible goals, while the control strategy considers the likelihood of these inferences in order to commit to a particular plan immediately. Allen's algorithm, as formulated, is intended to handle only single observations. Finally, Genesereth's plan recognition strategy draws heavily on a prestored model of the typical user to detect mistakes and misconceptions by the user. The procedure reconstructs the plan based on underlying beliefs in the user model. It suggests partial parsings of the user's actions and uses the user model to filter those suggestions. Thus, it too differs from the syntactic approaches by using knowledge about the user that is outside the plan library. As in these early approaches, we propose to *construct* valid plans from observed action sequences using first principles. We briefly describe our constructive view of plan recognition (CPR) in the section that follows.

[2]In particular, mouse-oriented interfaces to a chemical plant design system and a naval information database.

Constructive Recognition

In [Goodman and Litman 89], we propose a new formulation of plan recognition called "constructive plan recognition" that can address some of the problems outlined above. We briefly describe that formulation below.

Many of the implementations of plan recognition algorithms that we mentioned earlier have never been embedded inside a complete working application. As a result, crucial issues of robustness, reliability and inherent limitations remain. In particular, most current algorithms make the incorrect assumption that valid and complete plan knowledge is specified and shared by all agents, and they do not consider the fact that users often make mistakes or get sidetracked. Our research attempts to rectify these deficiencies by building on the early more semantic-based work in plan recognition described in the last section.

We wish to remove the assumption that the system's knowledge of the user's plan is complete. Otherwise, action sequences that differ from those stored in the system's plan library, even if only minimally, cannot be recognized. Dropping the completeness assumption is especially necessary in contexts where the user is often creating new plans. If we remove this assumption, however, our plan recognition algorithm cannot be limited to matching into a predefined set of expected plans. It must be more *constructive* in nature and attempt to fill in potential knowledge gaps when presented with a novel plan. It must dynamically construct from the incomplete knowledge new ways of performing high level actions, if such actions can be seen as purposeful. One goal of our work is, thus, to identify the relationship *purposeful* so that our plan recognizer can efficiently conclude the proper inferences.

Many existing plan recognizers define a purposeful action sequence "syntactically" (e.g., Huff and Lesser 82, Carver et al. (84), Sidner 85, Kautz 87). They perform a process similar to parsing by attempting to fit observed user actions into an expected user plan as (implicitly) defined by a complete plan library, not unlike a language that defines all possible sentences. CPR attempts to extend this parsing analogy by developing a "cascaded" parsing algorithm where syntax and semantics work hand in hand (Woods 80). However, while a traditional cascaded parser uses semantics to verify or eliminate existing syntactic choices, we are using semantics to *construct* additions to an incomplete syntactic language.[3] That is, if CPR cannot parse an observed action sequence, it attempts to determine whether or not the sequence could be part of the plan language (i.e., is "purposeful") by applying semantic information to piece together actions. Purposeful actions must be able to be seen as being part of an action sequence on the way towards achieving some goal. In the case of novel sequences, however, that goal is not necessarily known. We propose to define a metric for "purposefulness." Suppose we observe two actions: action A followed by action B. "Purposefulness by entailment" occurs if the effects of A make the preconditions of B satisfied. If we can construct intervening actions between A and B to connect them, then we have "purposefulness by construction." Finally, we have "purposefulness by example" if we can contrast the partial sequence "A B" that was observed against a known "purposeful" action sequence (i.e., finding a partial match).

Similarly to the early work mentioned in the last section, CPR can use plan generation techniques to determine whether novel sequences of actions are potentially purposeful. For example, CPR incrementally examines (using local backward and forward chaining) the effects and preconditions of actions in an action sequence and the propagation of those effects to determine whether or not they fit together well. In other words, an action sequence is (potentially) purposeful if it could have been generated by a planner from first principles.[4] When effects of earlier actions neither violate nor achieve preconditions of later actions, CPR can say nothing definitive about the causal structure of the plan. Instead, the desired interactions are viewed as expectations that need to be satisfied before the plan specification is through in order for the action sequence to remain valid. Because CPR uses bottom up techniques to incrementally *verify* the coherence of action sequences, the search explosion involved in earlier bottom up approaches to plan recognition (where search spaces were generated to *propose* action sequences) can be controlled.

Besides plan recognition by plan generation techniques, we are also investigating the recognition of plans by plan adaptation. What we learn from failed plan parses from the incomplete library, along with techniques of case-based and adaptive reasoning (e.g., Alterman 86, Broverman and Croft 87, Hammond 87, Kolodner et al. 85) can be used to determine if novel actions and goals form reasonable plans. In other words, we attempt to use the incomplete plan library to dynamically construct new ways of performing high-level actions. Finally, we only detect an errorful plan in the case where our semantic plan recognition techniques fail. We can then employ relaxation techniques similar

[3]The plan parser, thus, fills in knowledge gaps in our plan library.

[4]In the early semantic-based recognition algorithms the chaining together of actions was either controlled by expectations once the goal was known, or if totally bottom-up, it exploded.

to those used in natural language (Carberry 85, Goodman 86) to provide diagnostic support to repair the plan. Figure 3 below provides an overview of our proposed constructive plan recognizer.

3 Intelligent Interfaces

We want to provide practical uses of artificial intelligence in human-machine interfaces. Our primary motivation for enhancing the plan recognition procedure is to provide the flexibility required for building such robust interfaces. In this section we discuss intelligent interfaces, citing an example of one that we built. That interface, CHECS (CHemical Engineering CAD System), uses a standard plan recognizer. We explain why a constructive plan recognizer is required to provide the full power necessary to bring about a more robust interface. We end the section with a brief discussion about using the same paradigm for telerobotic interfaces.

3.1 Current Intelligent Interface Technology

Before discussing the CHECS interface, we discuss briefly how others have preceded our efforts to increase the robustness of interfaces and the methodologies they employed. In particular, we mention a couple efforts. User modelling techniques have been employed in an interface to Unix[5] as part of the Unix Consultant (Wilensky et al. 84) to provide help to stereotypical users (from novice to expert users). User models are generally "static" so that the inferences that can be deduced are limited in comparisons to the ones plan recognition can provide. Chin (88) has extended the Unix Consultant to take into account some pragmatic information about a user's interaction with the system. It tracks the user's knowledge over the course of a session. This allows it, for example, to avoid repeating things the user already knows. Fischer and Rathke (88) describe an interface for spreadsheets based on object-oriented knowledge representation. Their enhanced interface is most useful for the developer of the system and less for the actual user. It utilizes constraint based programming to represent knowledge about the application domain. It has a layered architecture to make it easy to modify and extend the system. Using plan recognition instead of constraint propagation should make it possible to bring the full power of such a system to the user as well as the application builder since the system could infer the user's intentions and reflect them dynamically in the system.

3.2 CHECS: An Intelligent Interface for Computer Aided Design[6]

3.2.1 CHECS

We performed an empirical study that involved examining a representative plan recognizer based on standard technology and using it to add more power and flexibility to an intelligent interface. To concretely explore the connections between interfaces and plan recognition, we designed and implemented CHECS, a graphical interface to a process design system. Design appears to be an excellent forum for applying plan recognition. Recent research projects in AI and design have expanded design systems from mere bookkeepers to active participants in the design process (e.g., Brown et al. 86). As we will see, an interactive plan-based interface can add a further layer of sophistication to these approaches. We have chosen chemical process engineering as our design domain because the interface is illustrative of many graphical design tools, and the domain is particularly amenable to standard planning and plan recognition representations. For example, there is a small, well-defined set of unit operations out of which other operations are hierarchically and functionally composed. We also chose design to push on the robustness of plan recognition, that is, since most designs are usually new, the design process is more than simply finding an old design.

The architecture of CHECS (and of plan-based interfaces in general) is shown in Figure 4. Its user interface is a graphical interface, inspired by existing graphical interfaces to chemical design simulators. As shown in Figure 5, the interface allows the user to build up a flow diagram by introducing chemical equipment (e.g., heat exchanger), flow (e.g., pipes connecting equipment), and parameters (e.g., heat-exchanger.input = n-butane). For example, by clicking on one of the icons representing equipment types (in the "CHECS Object" window), the user can build up a

[5]UNIX is a registered trademark of AT&T Bell Laboratories.

[6]A more complete description of this work can be found in [Goodman and Litman 89]. We would also like to acknowledge Kimberly Claffy for help in programming CHECS.

design containing equipment instantiations (in the "Plant Design" window). Parameters of the particular instantiations are then specified via menu interactions (Figure 6).[7]

After each user input, CHECS calls a plan recognizer to infer the chemical reaction (or plan) underlying the design, propagates information through the design to the appropriate points, and checks the consistency of the design step. For example, a portion of the butane isomerization plan recognized from the current design is shown in the lower portion of Figure 5. Currently, CHECS uses a modified implementation of Kautz's covering model of plan recognition (Kautz 87). The algorithm is incremental since it cyclically receives inputs and infers conclusions. For each input, the conclusions of the plan recognizer are further constrained by the conclusions drawn from previous inputs. In CHECS we assume that a single plan explains all observations (i.e., that a user is working on only one design at a time).

A plan is inferred using pre-existing knowledge of actions. This knowledge is expressed in the form of interleaved frame hierarchies, defining abstraction and decomposition relationships between actions. For example, the fact that a *heating-action* "is-a" *change-temperature-action* is an abstraction, while the specification of the steps involved in the performance of a *heating-action* is a decomposition. The CHECS hierarchy contains frames representing primitive chemical actions (e.g., heat, cool), chemical plans composed of primitive actions and/or other plans (e.g., reaction with recycle, as in Figure 5), and a distinguished set of plans called *end* plans, which are not components of any other plans. There is also an interleaved object hierarchy, representing the equipment associated with each primitive action (e.g., a heat exchanger can be used to implement a heating or cooling action). Finally, there is a hierarchy of chemical substances, as well as a database of facts containing particular assertional information. In CHECS, the goal of plan recognition is the inference of an instantiated end plan given the specified configuration.

Finally, just as plan recognition has been used to enhance various systems, CHECS recognizes and then manipulates plans in order to provide new design interface capabilities. As will be discussed below, plan recognition is used to add conceptual design completion, error handling, advice, and plan-dependent system responses.

3.2.2 Plan Recognition Help for CAD Interfaces

Typical graphical front-ends to design simulators provide standard graphics capabilities such as the creation of icons to represent parts, the selection and placement of parts, and editing facilities. Everything is performed manually including the layout. More recent systems help the designer with layout, allow parameter values to flow through the design automatically, provide symbolic simulation, and determine design validity by contrasting the designer's selected values against other pieces of the design. With plan recognition, we were able to implement features in CHECS not found in current systems. These advances allow the system to reflect the context in the system's current state, such as through menus, system advice, error detection and recovery, and bookkeeping. What we achieved in CHECS can be seen as a step towards more intelligent *cooperative* design tools. It is not automatic design, but instead keeps the user in the loop with most but not all of the control.

For example, CHECS is able to constrain icons and parameter values by considering what is legal with respect to what the system infers that the designer is doing. Such constraints are visible in "dynamic menus" which reflect the current context by dimming out entries that are already filled by the designer, that are inferred and propagated by the plan recognizer, or that are illegal given the hypothesized design(s). Figure 6 illustrates the menu generated for specification of an input to a reactor vessel. The input menu is dimmed because an entry, *n*-pentane, has already been constrained as a result of inferences made by the plan recognizer from an earlier observation (e.g., by propagating the output from one part through a pipe to the input of the reactor). The menu could have been only partly dimmed if more than one entry was possible. Dynamic menus, thus, provide salience to their entries.

CHECS, once it infers the designer's goal, can also make inferences that provide shortcuts to the design process or even allow design completion. For example, once CHECS determines a unique plan in the library (or common elements in the remaining "possible" plans), it can complete all (or portions of) the user's design based upon the inferred plans(s). Consider Figure 7 in which the user has added first a feed (a container of a particular chemical substance) and then a still (a device used to separate a chemical from a mixture). From the plan library and from local constraints, CHECS can predict that a heat exchanger (a device used for heating or cooling, shown as dimmed in the figure) is needed on an input to the still. That is, stills require that their inputs are heated and, if no outputs of parts already on the screen are heated, the system could *predict* that the user will have to heat the input to the still.

[7]These figures show snapshots of the current system. CHECS is implemented in Common Lisp; the graphical interface runs on a Macintosh, while a textual version runs on other machines.

The heat exchanger is shown dimmed as a "suggestion" which the user can accept by clicking on it or reject by proceeding in a different manner.

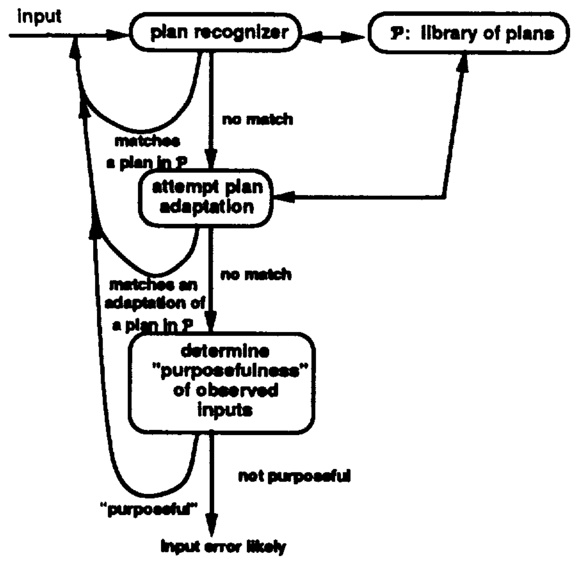


Figure 3: Constructive plan recognition

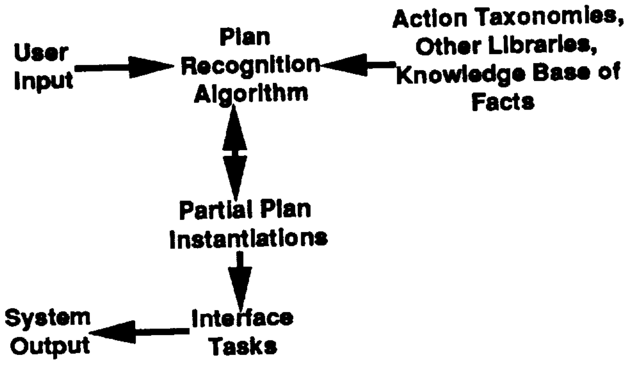

User Input
Plan Recognition Algorithm
Action Taxonomies, Other Libraries, Knowledge Base of Facts
Partial Plan Instantiations
Interface Tasks
System Output

Figure 4: A Generic Plan-Based Interface System

Because CHECS recognizes the plans that underlie designs, CHECS can infer and suggest designs that do not directly match the user's flow graph, but that instead have the same underlying functional (plan) structure. So, for example, even if the user's design of Figure 5 had a heat exchanger connected to the output of the reactor, the plan recognizer could still match the design conceptually against a design with a "heated reactor" (a reactor with a heating coil built in), because a chemical subplan such as "react-and-heat" can be achieved by either flow configuration.

CHECS also provides for error handling by checking design validity. Design validity entails that the user's design achieve a plan in the library. When it does not, the design is invalid with respect to the language of the design library. In its strongest instantiation, the system prevents the user from making mistakes by only allowing the user to perform actions that can lead towards one of the inferred designs.

Finally, CHECS provides advice and active bookkeeping. For example, CHECS can use conceptual matches to suggest functionally equivalent but more optimal designs (such as the use of a heated reactor for a reactor, pipe, and heat exchanger configuration). CHECS also does a better job at bookkeeping then previous CAD systems since it has a higher perspective of the design then available in object-oriented design systems. By inferring the design goal, CHECS can propagate information through the design and utilize it immediately. For example, even though we did not implement this feature in CHECS, inferring the possible matching designs in the design library can allow the system to do a better job (than the user) at the layout of the user's design.

3.2.3 Implications of CHECS

We built our first version of CHECS to study the weaknesses and strengths of an interface integrated with a plan recognition component. Our goal was to isolate shortcomings in a class of approaches and to propose ways around them. We discovered, as outlined in the previous section, that plan recognition can add numerous benefits to an interface in the areas of plan completion, error detection and recovery, advice generation, and context dependent responses. In turn, we also determined that current plan recognition systems are fundamentally limited in ways that limit the power of the associated interface. They have little to say about the recognition of novel plans; they typically search for known goals explained by known action sequences. However, one's planning knowledge is incomplete, can contain errors, and differs among different users. Hence, the recent implementations of plan recognizers make unreasonable assumptions that get in the way of a robust interface. They made some of the user enhancements awkward in practice. The plan recognizers caused the interfaces to exert too much control over the direction of the user in accomplishing his or her task. The more flexible we allow an interface to be (e.g., allowing the user to ReStart, BackUp, or Edit their commands), the more likely communication problems will occur. Current interfaces restrict such freewill at a cost of making the interface awkward to use. We contend that our proposed constructive plan recognizer can achieve a balance between the user and the system. Constructive plan recognition can also broaden the communication medium. As applications become increasingly complex, the interfaces between them and their users take on an ever increasing burden in providing effective communication. Such interfaces can expect mixed modalities in the communication channel. For example, a user can point with a mouse (or other pointing device), select commands from a menu, provide typed natural language commands, or give spoken commands.

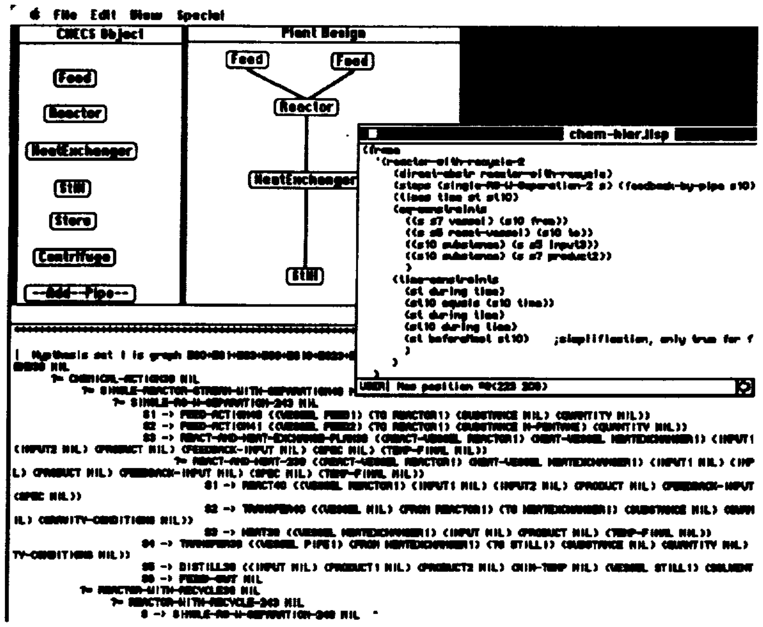
3.3 Intelligent Interfaces for Telerobotics

Our constructive plan recognition algorithm could serve as a backbone for an intelligent interface to a telerobot. Its strengths over standard plan recognizers have already been elucidated. In practice it would be capable of building on any prestored plans in the incomplete task plan library. CPR better fits the patched together nature of telerobotic actions. Since matching complete prestored plans to a particular situation won't always work in the unknowns of space, operators must adapt their plans or learn as they go. An interface that can effectively work for an operator in that environment, hence, must be capable of tracking varying user actions. It must detect and ignore side actions performed by the operator that aren't related to the overall goal while still responding to contingencies that arise.

A telerobot interface could be extended using CPR to add a more cooperative layer of sophistication and to bring the interface and telerobotics task closer together. CPR would monitor the operator's actions in an attempt to infer both high-level and low-level goals based on the latest action in the action sequence. When a higher-level goal is inferred, the interface can request the response planning component to complete the task. Otherwise, based on local actions, the interface can augment the operator's actions with more precise actions.

4 Conclusion

Plan recognition is an important part of any telerobot since it can help the robot determine an astronaut's intentions from his actions as well as working in conjunction with a planner to guide the robot in responding to contingencies. Previous plan recognition paradigms required the system builder to build large libraries of all possible plans and situations. Constructive plan recognition can provide the robust form of plan recognition required for telerobotics. CPR is more realistic than previous plan recognition by recognizing that tasks are often approached in a novel way. CPR is meant to survive without a complete and predefined plan library relieving some of the burden from the user.



```
(frame
  '(reactor-with-recycle-2
    (direct-abstr reactor-with-recycle)
    (steps (single-RO-W-Separation-2 s) (feedback-by-pipe s10))
    ((time time st st10)
    (eq-constraints
      ((s s7 vessel) (s10 from))
      ((s s5 react-vessel) (s10 to))
      ((s10 substance) (s s5 input2))
      ((s10 substance) (s s7 product2))
      )
    (time-constraints
      (st during time)
      (st10 equals (s10 time))
      (st during time)
      (st10 during time)
      (st beforeMeet st10)     ;simplification, only true for f
      )
    )
  )
USER| New position #<(223 308)
```

```
| Hypothesis set 1 is graph [illegible]
[illegible] NIL
    ?= CHEMICAL-ACTION[illegible] NIL
        ?= SINGLE-REACTOR-STREAM-WITH-SEPARATION[illegible]
            ?= SINGLE-RO-W-SEPARATION-243 NIL
                S1 -> FEED-ACTION40 ((VESSEL FEED1) (TO REACTOR1) (SUBSTANCE NIL) (QUANTITY NIL))
                S2 -> FEED-ACTION41 ((VESSEL FEED2) (TO REACTOR1) (SUBSTANCE N-PENTANE) (QUANTITY NIL))
                S3 -> REACT-AND-HEAT-EXCHANGE-PLAN[illegible] (REACT-VESSEL REACTOR1) (HEAT-VESSEL HEATEXCHANGER1) (INPUT1
(INPUT2 NIL) (PRODUCT NIL) (FEEDBACK-INPUT NIL) (SPEC NIL) (TEMP-FINAL NIL))
                    ?= REACT-AND-HEAT-230 (REACT-VESSEL REACTOR1) (HEAT-VESSEL HEATEXCHANGER1) (INPUT1 NIL) (INP
L) (PRODUCT NIL) (FEEDBACK-INPUT NIL) (SPEC NIL) (TEMP-FINAL NIL))
                        S1 -> REACT40 ((VESSEL REACTOR1) (INPUT1 NIL) (INPUT2 NIL) (PRODUCT NIL) (FEEDBACK-INPUT
(SPEC NIL))
                        S2 -> TRANSFER40 ((VESSEL NIL) (FROM REACTOR1) (TO HEATEXCHANGER1) (SUBSTANCE NIL) (QUAN
IL) (QUANTITY-CONDITIONS NIL))
                        S3 -> HEAT30 ((VESSEL HEATEXCHANGER1) (INPUT NIL) (PRODUCT NIL) (TEMP-FINAL NIL))
                S4 -> TRANSFER[illegible] ((VESSEL PIPE1) (FROM HEATEXCHANGER1) (TO STILL1) (SUBSTANCE NIL) (QUANTITY NIL)
TY-CONDITIONS NIL))
                S5 -> DISTILL30 ((INPUT NIL) (PRODUCT1 NIL) (PRODUCT2 NIL) (MIN-TEMP NIL) (VESSEL STILL1) (COOLANT
                S6 -> FEED-OUT NIL
        ?= REACTOR-WITH-RECYCLE[illegible] NIL
            ?= REACTOR-WITH-RECYCLE-243 NIL
                S -> SINGLE-RO-W-SEPARATION-[illegible] NIL
```

Figure 5: Designing a Butane Isomerization Process

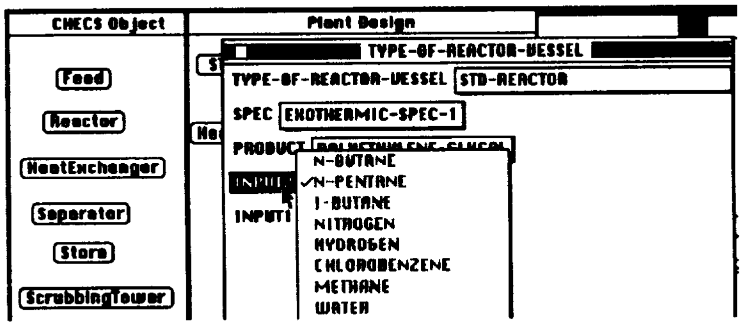


Figure 6: CHECS Dynamic Menu

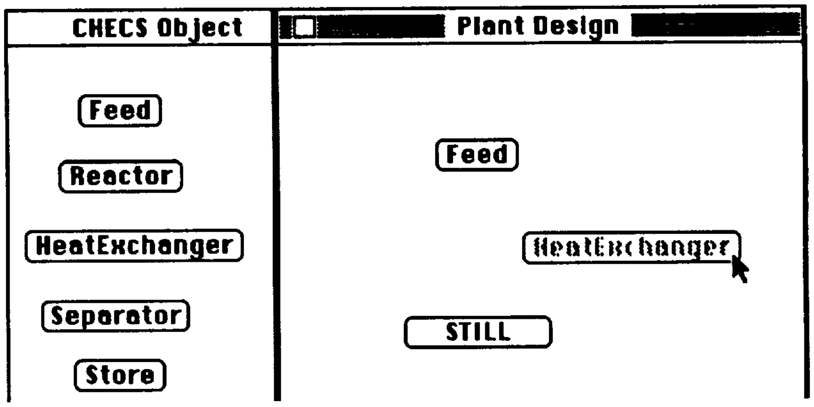


Figure 7: Local Prediction

5 References

Allen, James F. Recognizing Intentions from Natural Language Utterances. In *Computational Models of Discourse*, M. Brady and B. Berwick (eds.), MIT Press, 83.

Allen, James F. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, Inc., Menlo Park, Ca., 87.

Alterman, Richard. An Adaptive Planner. In *Proceedings of AAAI-86*, pages 65-69. Philadelphia, Pa., August, 86.

Bejczy, A. K., Man-Machine Interface Issues in Space Telerobotics: A JPL Research and Development Program. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Broverman, Carol A. and W. Bruce Croft. Reasoning About Exceptions During Plan Execution Monitoring. In *Proceedings of AAAI-87*, pages 190-5. Seattle, Wa., July, 87.

Brown, David C. and B. Chandrasekaran. Knowledge and Control for a Mechanical Design Expert System. *Computer* (7):92-100, July, 86.

Carberry, Mary Sandra. *Pragmatic Modeling in Information System Interfaces*. PhD thesis, University of Delaware, 85.

Carver, N.F., Lesser, V.R., and McCue, D.L. Focusing in Plan Recognition. In *Proceedings of the National Conference on Artificial Intelligence*, The American Association for Artificial Intelligence, Austin, TX, 84.

Chin, David, Exploiting User Expertise in Answer Expression. In *Proceedings of AAAI88*, pages 756-760, St. Paul, Mn., 88.

Cohen, Philip R. and Hector J. Levesque, Speech Acts and Rationality, in *Proceedings of the 23rd ACL Conference*, pages 49-59, Chicago, July, 85.

Dean, T. and M. Brody, Prediction and Causal Reasoning in Planning. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Drummond, M., K. Currie and A. Tate, Contingent Plan Structures for Spacecraft. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Durfee, E. H. and V. R. Lesser, Incremental Planning to Control a Blackboard-Based Problem Solver. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Fischer, Gerhard and Christian Rathke, Knowledge-Based Spreadsheets. In *Proceedings of AAAI-88*, pages 802-807, St. Paul, Mn. 88.

Genesereth, Michael R. The Role of Plans in Automated Consultation. In *Proceedings of IJCAI-79*, pages 311-3, Tokyo, Japan, August, 79.

Georgeff, M. P., A. L. Lansky and M. J. Schoppers, Reasoning and Planning in Dynamic Domains: An Experiment With a Mobile Robot. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Goodman, Bradley A. Reference Identification and Reference Identification Failures. *Computational Linguistics* 12(4):273-305, October-December, 86.

Goodman, Bradley A. and Diane J. Litman, Plan Recognition for Intelligent Interfaces. Submitted for review to IJCAI-89, 89.

Hammond, Kristian J. Explaining and Repairing Plans that Fail. In *Proceedings of IJCAI-87*, pages 109-114, Milan, Italy, August, 87.

Huff, Karen and Victor Lesser. *Knowledge-Based Command Understanding: An Example for the Software Development Environment.* Report TR 82-6, Computer and Information Sciences, University of Massachusetts at Amherst, Amherst, Ma., 82.

Jenkins, L., Space Telerobotic Systems: Applications and Concepts. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Kautz, Henry A. *A Formal Theory of Plan Recognition.* PhD thesis, University of Rochester, 87. Also, TR215, University of Rochester, Dept. of Computer Science, Rochester, N.Y.

Kolodner, Janet L., Robert L. Simpson, Jr., and Katia Sycara-Cyranski. A Process Model of Case-Based Reasoning in Problem Solving. In *Proceedings of IJCAI-85*, pages 284-290, Los Angeles, August, 85.

Litman, Diane J. and James F. Allen. A Plan Recognition Model for Subdialogues in Conversations. *Cognitive Science* 11:163-200, 87.

Pollack, Martha E. Inferring Domain Plans in Question-Answering. PhD thesis, University of Pennsylvania, 86. Also, Report MS-CS-86-40 of the Department of Computer and Information Science, University of Pennsylvania

Rodriguez, G. (ed.), *Proceedings of the Workshop on Space Telerobots*, NASA and JPL, July 87.

Ross, Peter and John Lewis, *Plan Recognition and Chart Parsing*, DAI Research Paper No. 309, University of Edinburgh, Edinburgh, Scotland, 87.

Sacerdoti, Earl D. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence* 5:115-135, 74.

Schank, R. C. and C. J. Reiger, Inferences and the computer understanding of natural language, *Artificial Intelligence* 5, pages 333-412, 74.

Schmidt, C. F., Sridharan, N. S., and Goodson, J. L. The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence. *Artificial Intelligence* 11:45-83, 78.

Sidner, Candace L. Plan parsing for intended response recognition in discourse. *Computational Intelligence* 1(1):1-10, 85.

Wilensky, Robert, Yigal Arens, and David Chin. Talking to UNIX in English: An Overview of UC. *Communications of the ACM*: 27(6): 574-593, 84.

Vilain, Marc, Chart Parsing for Plan Recognition Proposal, Personal communication, BBN, 88.

Wilkens, D. E., Recovering from Execution Errors in SIPE. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Woods, William A. Cascaded ATN Grammars. Amer. J. Computational Linguistics 6(1):1-15, Jan.-Mar., 80.

N90-29037

CAUSAL SIMULATION AND SENSOR PLANNING IN PREDICTIVE MONITORING

Richard J. Doyle

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Abstract

We address two issues which arise in the task of detecting anomalous behavior in complex systems with numerous sensor channels: how to adjust alarm thresholds dynamically, within the changing operating context of the system, and how to utilize sensors selectively, so that nominal operation can be verified reliably without processing a prohibitive amount of sensor data. Our approach involves simulation of a causal model of the system, which provides information on expected sensor values, and on dependencies between predicted events, useful in assessing the relative importance of events so that sensor resources can be allocated effectively. We discuss briefly the potential applicability of this work to the execution monitoring of robot task plans.

The Monitoring Problem

Timely detection of anomalous behavior is essential for the continuous safe operation and longevity of aerospace systems. The pilot of a jet aircraft must be aware of any conditions which may affect thrust during the critical moments of takeoff. The thermal environment onboard Space Station Freedom must be carefully controlled to provide uninterrupted life support for the crew. The Mars Rover must react quickly to an unpredictable environment or the mission may come to an abrupt conclusion.

The monitoring problem becomes more difficult when the behavior of a physical system involves interactions among components or interaction with an environment. Under these conditions, correct operation becomes context-dependent; it is not possible to determine *a priori* a set of sensor values which always imply nominal operation. Moreover, when the number of sensors in a physical system becomes very large, the ability to combine sensor data into a picture of the global state of a system becomes compromised. Studies of plant catastrophes have revealed that information which might have been useful in preventing disaster was typically available but was not prominent enough within the overwhelming morass of data presented to operators.

In this paper, we concentrate on the initial step in the monitoring process---detecting anomalous behavior quickly and reliably. We do not address here the equally important steps of tracking faulted behavior and determining control actions to continue operation in the presence of faults. Within this focus, we address two important issues: (1) how to adjust nominal sensor value expectations dynamically, taking into account the changing operating context of the system, and (2) how to utilize sensors selectively, determining which subset of the available sensors to use at any given time to verify nominal operation efficiently, without processing a prohibitive amount of data.

Two Issues

The traditional approach to verifying the correct operation of a system being monitored involves associating alarm thresholds with sensors. Fixed threshold values for each sensor are determined ahead of time by analyzing the designed nominal behavior for the system. Whenever a sensor value crosses a threshold during operation, an alarm is raised.

The problem with this approach is that the nominal behavior of even moderately complex systems often depends on context. For example, an earth-orbiting spacecraft periodically enters and emerges from the Earth's shadow. Impingent solar radiation changes the thermal profile of the spacecraft, as does the configuration of currently active and consequently, heat-generating subsystems on board. Thresholds on temperature sensors should be adjusted accordingly. A particular temperature value may be indicative of a problem when the spacecraft is in shadow or mostly inactive, but may be within acceptable limits when the spacecraft is in sunlight or many on-board systems are operating.

Fixed alarm thresholds are useful for defining the operating limits of a physical system, such as the point of overbalance of a rover, or the temperature at which, say, the onboard computer of a spacecraft is at risk. Nonetheless, they are woefully inadequate for verifying the nominal operation of a system with many operating modes, or one which interacts with an environment. The problem is that fixed alarm thresholds are derived from an over-summarized model of the behavior of a system. If the thresholds are chosen conservatively, then false alarms occur. If they are chosen boldly, then undetected anomalies occur. What is needed is a capability for adjusting alarm thresholds dynamically. Alarm thresholds should be chosen according to expectations about the nominal behavior of a system as it changes in different operating contexts. Later on in this paper, we present our approach to dynamic alarm threshold adjustment based on causal simulation of the device.

Another issue which arises in monitoring concerns how to best utilize available sensors to efficiently and reliably, but not necessarily comprehensively, verify the nominal operation of a physical system. Just as the nominal values in a system being monitored depend on context, so do the subset of sensors which can most directly verify those values depend on context. The familiar activity of driving an automobile helps to illustrate this idea. A variety of sensors are provided to the operator of an automobile: fuel gauge, temperature gauge, speedometer, several mirrors, etc. However, the driver does not use all of these diverse sensors all of the time. The speedometer may be checked periodically, or when a speed limit sign is passed; the right-side mirror is probably only used during lane changes. There are two points to be made: one concerns relevance, the other concerns resources.

Individual sensors are appropriate for verifying only some small, localized subset of the possible behavior of a system. The choice of which sensors to sample and interpret at any particular time should be based on expectations of what is to happen in the system and, perhaps, how it is to interact with an environment. However, even after a suitable subset of the available sensors is identified, there may not be the resources available, whether human or machine, to sample all the selected sensors and interpret the data within a required response frame. What is needed is a capability for assessing the importance of predicted events, so that while it may not be possible to comprehensively verify the expected behavior of a system, still the most reliable verification within available resources can be performed.

An illustration of the need to focus attention in monitoring comes from the jet aircraft domain. Some of the recent commercial aircraft catastrophes have been attributed to insufficient thrust during the critical moments of takeoff. There are many possible indicators of low thrust available to a flight crew. For example, a low exhaust gas temperature in an engine may produce reduced thrust. Also, a low turbine fan rotation speed in an engine may imply reduced thrust, because fuel input is based partly on this parameter. The problem is how to direct the attention of the flight crew without overwhelming them towards information useful for planning compensating actions in real time.

A monitoring strategy must take into account the reality that not all sensors should or can be checked all of the time. As the operating context of the physical system being monitored changes, the collection of sensors which provide the most immediate information on the state of the system also changes. Further on in this paper, we present our approach to sensor planning in monitoring. We describe a method for assessing the importance of predicted events in a system, based on reasoning about causal dependencies among events, and about how events relate to intended goals of the designers or operators of a system.

Other Work

Within NASA, there are other projects underway in which the goal is to develop a monitoring and a diagnosis capability for aerospace systems. Among these is the KATE project at the Kennedy Space Center, whose domain is the Shuttle Liquid Oxygen Loading system [1]. In this project, causal models are used to support sensor validation, fault diagnosis, and the planning of control actions.

The goal of the FAULTFINDER project at Langley Research Center [2] is to develop an inflight monitoring and diagnosis capability for jet aircraft. These investigators have explored the use of multiple representations and multiple levels of abstraction to be able to reason about diverse faults, to focus attention during reasoning, and to provide accessible information to a flight crew.

Numerous other examples exist of efforts to develop monitoring and control systems. The reader is referred to Dvorak's excellent survey of the area [3].

The causal reasoning paradigm, which is at the core of our approach to the monitoring problem, is now a well-established area of investigation within Artificial Intelligence. The advantages of the causal approach, which involves modeling a system at the level of components and mechanisms, include the ability to reason about unforeseen interactions, the ability to reason about dependencies among events, and the ability to generate accessible explanations. The seminal efforts in this area include Forbus' process-centered approach [4], de Kleer and Brown's device-centered approach [5], and Kuipers' qualitative mathematics approach [6].

In the specific area of monitoring, Dvorak's MIMIC project stands out as the most comprehensive current research effort [7]. Dvorak creates a component-connection model of a system and employs the QSIM qualitative simulator [6] to generate expectations about the system's nominal behavior. An inductive learning method is used to create a set of symptom-fault rules for known faults, and these rules support the formation of fault hypotheses whenever sensor data does not match predictions from the causal model. When anomalous behavior exists, several fault models can be tracked in parallel until one emerges as the hypothesis with the most explanatory power. The ability to continue tracking a faulted system is important because large, complex systems almost always contain faults and the challenge is to maintain safe operation in the presence of faults.

The Approach

At the center of our approach to addressing the two issues of dynamic alarm thresholds and sensor selection is a causal model of the system being monitored and possibly, its environment. Simulation of this model directly solves the problem of alarm threshold adjustment. Predicted values and their time tags indicate how and when to alter the alarm thresholds associated with sensors so that they reflect expectations about the nominal operation of the system in changing contexts.

Another result of simulation is information about causal dependencies among predicted events of a system. This information is used to assess the importance of individual events. Briefly, the most important events are taken to be those which either cause or are caused by the greatest number of other events. An ordering on predicted events reflecting this causal notion of importance serves as the basis for allocating sensor resources to selectively verify the expected behavior of a system [8].

In the remainder of this section, we describe (1) the architecture of our predictive monitoring system, called PREMON, (2) what our causal models of physical systems look like, and how they are simulated, and finally, (3) our approach to sensor planning, based on analyzing causal dependencies.

Architecture

There are three modules in the PREMON system: a causal simulator, a sensor planner, and a sensor interpreter. See Figure 1.

The causal simulator takes as input a causal model of the system to be monitored, and a set of events describing the initial state of the system and perhaps some future scheduled events. The causal simulator produces as output a set of predicted events, and a graph of causal dependencies among those events.

The sensor planner takes as input the causal dependency graph generated by the causal simulator and determines which subset of the predicted events should be verified. These events are passed on to the sensor interpreter.

The sensor interpreter compares expected values as predicted by the causal simulator with actual values from sensors. Alarms are raised here when there are discrepancies. Finally, the most recent sensed data is passed back to the causal simulator to contribute to another predict-plan-sense cycle of monitoring.

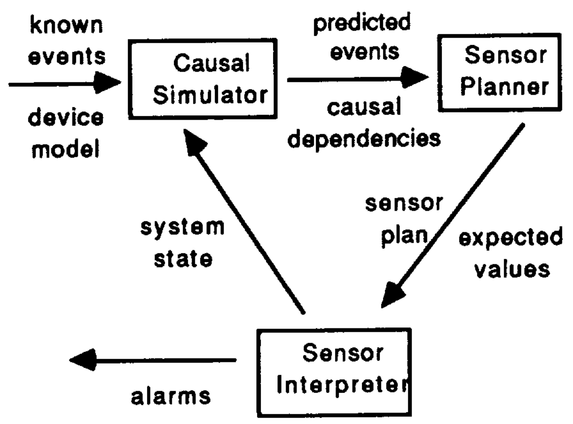


Figure 1. Architecture of PREMON.

Causal Models and Causal Simulation

We represent physical systems as a collection of *quantities* and *mechanisms*. Quantities are continuous parameters such as temperature, position, and amount-of-stuff. Quantities are specified by a *physical object*, a *type*, and an *order*. Examples of quantities are {HEATER TEMPERATURE RATE} and {SWITCH POSITION AMOUNT}.

Events are discontinuous changes in the value of a quantity. Events are specified by a quantity, a *value*, and a *moment*. Examples of events are {HEATER TEMPERATURE RATE POSITIVE 61} and {VALVE-17 POSITION AMOUNT OPEN 0}.

Mechanisms capture causal relations between quantities. More specifically, they describe how a change in one quantity results in a change in another quantity. Examples of mechanisms are HEAT FLOW, THERMAL EXPANSION, LATCH, and GRAVITY. A mechanism is specified by a *time constant*, a *distance*, a *sign*, an *efficiency*, a *bias*, an *alignment*, and a *medium*. Figure 3 shows the representation of a HEAT FLOW mechanism.

A *causal model* then, consists of a set of quantities and a set of mechanisms between those quantities. A causal model can be represented by a graph where the nodes are quantities and the arcs are mechanisms. Simulation of a causal model involves predicting new events, via mechanisms, from known or previously predicted events. The simulation method outlined in the next few paragraphs is described more fully in [9].

When the quantity named in an event appears as the cause quantity in a mechanism, a new event is predicted as follows: (1) the quantity of the new event is the effect quantity of the mechanism, (2) the value of the new event is computed from the value of the given event and the sign and efficiency of the mechanism, (3) the moment of the new event is computed from the moment of the given event and the time constant and distance of the mechanism, and (4) the new event occurs only when constraints specified in the bias, alignment, and medium of the mechanism are satisfied. The bias of a mechanism specifies constraints on directions of change. For example, current through a wire can cause it to heat up, but not to cool down. The alignment of a mechanism specifies constraints expressed as inequalities. For example, heat flow is from the warmer to the cooler site. The medium of a mechanism is a physical connection such as a wire, a pipe, a linkage, etc. The predicted effect occurs only when the specified physical connection is in place.

A typical event, this one describing a temperature change, is shown in Figure 2. The HEAT FLOW mechanism in Figure 3 is used to predict another temperature change event, shown in Figure 4.

QUANTITY	Chiller Temperature Rate
VALUE	Negative
MOMENT	60

Figure 2. A cause event.

TIME CONSTANT	1.0
DISTANCE	10.0
SIGN	+
EFFICIENCY	0.95
BIAS	--
ALIGNMENT	<
MEDIUM	{Chiller Pipe-4 Mirror}

Figure 3. A mechanism.

QUANTITY	Mirror Temperature Rate
VALUE	Negative
MOMENT	70

Figure 4. An effect event.

Simulation would be straightforward if physical systems could be modeled exclusively as simple mechanism chains between input and output quantities. However, some mechanisms serve to enable or disable other mechanisms, such as a valve controlling a fluid flow, or a latch inhibiting the transmission of motion through a mechanical coupling. In these cases, the contributions of the separate mechanisms combine multiplicatively. The value contributed by the enabling or disabling mechanism can be discrete, as in the case of a switch, or continuous, as in the case of a valve. The contributions of separate mechanisms also can combine additively, as when two fluid lines empty into the same container, or two opposed forces produce an equilibrium state.

Sensor Planning

The output of the causal simulator is a trace of predicted events and the dependencies among them. The dependencies are derived from the mechanism structure of the system. A dependency between two events is a record that there is a mechanism in the system which causally relates the events.

Analysis of the causal dependencies in a simulation trace supports decisions about which events to monitor. In our approach, the importance of events is assessed by determining how many other events are effects or causes of a given event. In other words, the importance of an event is related to the amount of subsequent activity it supports and the amount of activity which supports its occurrence. Critical events which lie on several causal paths between inputs and outputs should be verified with care, perhaps with a battery of sensors. On the other hand, events which are side effects and do not support further activity in the system may be ignored completely. See Figure 5.

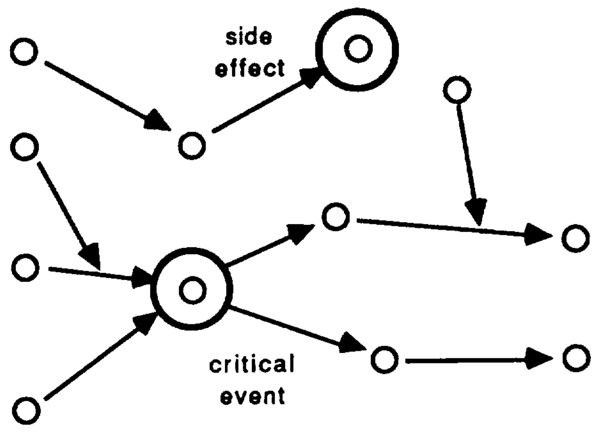


Figure 5. Assessing the importance of events.

This analysis method weights all dependencies in a causal graph equally. Several criteria might form the basis of a non-uniform weighting scheme. For example, *a priori* or empirical knowledge about probabilities of failure might bias the allocation of sensor resources towards those components in a system known to be unreliable. Similarly, parts of a system where redundancy has been built in might be given less careful attention than other parts.

Our causal analysis method for determining what subset of predicted events to monitor is similar to the minimum entropy method of de Kleer and Williams [10] for determining the site of the most useful next measurement in troubleshooting. Their technique involves propagating observed values and failure probabilities along a causal dependency graph for a circuit.

An Example: The JPL Space Simulator

The JPL Space Simulator is an environmental chamber in which spacecraft and instruments can be subjected to some of the aspects of the space environment: intense cold, near vacuum, and solar radiation.

A mirror is used to direct simulated solar radiation onto the spacecraft or instrument inside the chamber. This mirror must be cooled separately from the shroud which surrounds the chamber to compensate for the additional radiation falling on it. Cold gaseous nitrogen is used as the cooling medium and is circulated by a fan. Chilling is achieved by injecting liquid nitrogen into the gaseous nitrogen. Warming is achieved through an electrical heater. A causal simulation of this cooling circuit is shown in Figure 6.

Using the causal analysis technique outlined above, the flow of gaseous nitrogen at the fan is identified as the single most critical event in the predicted nominal behavior of the circuit. This event affects gas flow around the entire circuit and indirectly, heat flow around the entire circuit. The only events unaffected by this event are the source temperature changes at the chiller and heater. This result of causal analysis captures the intuitive notion that nothing at all happens in the cooling circuit if the fan stops operating. Other important events in the predicted operation of the circuit are the temperature changes at the chiller and heater. Measurements made at these sites also provide informative feedback about the nominal operation of the circuit.

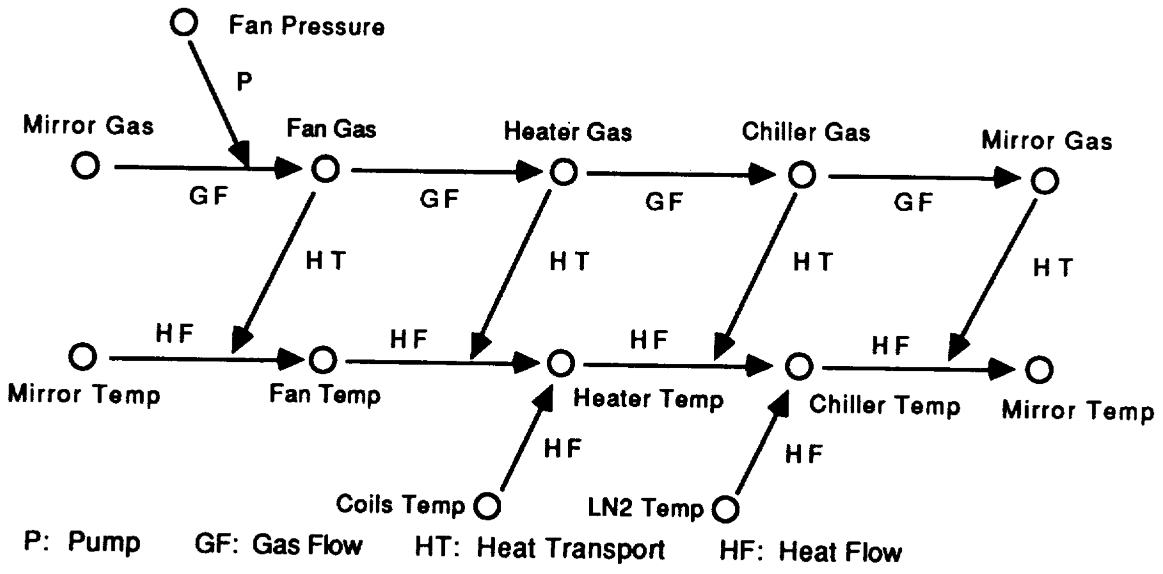


Figure 6. The mirror cooling circuit.

This example has been implemented and illustrates our current causal simulation and sensor planning capabilities. We are beginning to apply our developing predictive monitoring capability to other aerospace systems existing or being designed within NASA. One of these is the Mars Rover. We are looking at the monitoring problems associated with terrain traversal, power distribution, and thermal distribution. In addition, we are looking at telecommunications systems used in sending commands to and receiving telemetry from spacecraft, such as antenna control systems in the Deep Space Network. Finally, we have looked also at some of the smaller, earth-orbiting spacecraft.

Application to Robot Task Plan Execution Monitoring

The task of monitoring the behavior of a physical system bears similarities to the task of monitoring the execution of a robot task plan [11,12,13]. In fact, the roots of the work described in this paper are in an earlier effort which addressed the generation of expectations and perception requests to verify the execution of robot plans [14]. The two issues which form the focus of this paper apply also to the monitoring of robot plans: (1) How to predict the sensor values which imply correct execution of a plan, and (2) How to generate predictions and interpret sensor data selectively to meet real-time constraints with limited computational resources.

The sensor values which indicate nominal execution of a plan vary. For example, the force readings which verify that a gripping action has been performed successfully depend on the grip configuration and the properties of the gripped object. The appearance of an object for recognition by a vision system depends on its reflectance properties and on lighting. Physical models can be used to derive these context-dependent nominal sensor values. Some of these models might be causal models describing physical processes in the environment, including the interactions of the robot with the environment.

Moreover, the importance of the individual actions in a plan may differ, and sensor interpretation resources should be allocated accordingly. For example, the gripping of a tool prior to a sequence of manipulations should be verified carefully, perhaps with a battery of sensors. Conversely, a gross motion prior to a gripping action may be verified more cursorily. The approach to sensor planning based on causal dependency analysis developed for the physical device domain maps directly to the robot task planning domain. The dependency graph in Figure 5 might describe the logical dependencies among the preconditions and consequences of actions in a task plan as easily as the causal dependencies among events in the operation of a physical device. As described above, the analysis method distinguishes critical actions from actions whose consequences are only side effects. Other criteria may also be used to assess the need to verify individual actions in a plan. For example, a motion which makes use of compliance may be assumed to be more robust and require less exact verification.

Other issues which arise in the execution monitoring of robot task plans include: How to deal with uncertainty in the world model and in the operation of the robot? How to infer nominal execution when direct sensing is not possible? At what point(s) should a condition be verified when there is a delay between its establishment by one action and its enabling of another action? Which sensors to read when several are relevant and how to fuse data from disparate sensors? What are the interactions of task planning with sensor planning?

Conclusions

Detecting anomalies in the operation of a system is a difficult problem when the behavior of the system is complex or involves interaction with an environment, and when the number of sensor channels is large. Under these conditions, nominal values and the most informative sensor data change according to context. We have addressed two specific issues in monitoring: how to adjust alarm thresholds dynamically, and how to verify behavior selectively but reliably. At the center of our approach to solving both problems is the use of a causal model of the system being monitored. Simulation of a causal model serves both to generate expectations about nominal sensor values, and to provide dependency information useful in assessing the importance of predicted events and in allocating sensor resources accordingly. Some aspects of this approach appear to be applicable to the task of monitoring the execution of robot plans.

The key idea in this paper is letting go of the notion of comprehensive monitoring. More likely than not, there will be insufficient resources for predicting behavior and interpreting sensor data. In the face of this limitation, our emphasis is on verifying the operation of a system efficiently and reliably, by carefully focusing computational resources to gather the most informative, if incomplete, feedback on nominal operation within changing contexts.

Acknowledgements

The content of this paper has been enhanced through discussions with Kathy Abbott, Raj Doshi, Dan Dvorak, Lorraine Fesq, David Franke, Ben Kuipers, and Ethan Scarl. The expertise of Bob Lane was crucial in developing the model of the mirror cooling circuit.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] Ethan A. Scarl, John R. Jamieson, and Edwin New, "Deriving Fault Location and Control from a Functional Model," *3rd IEEE Symposium on Intelligent Control,* Arlington, Virginia, 1988.

[2] Kathy H. Abbott, "Robust Operative Diagnosis as Problem Solving in a Hypothesis Space," *7th National Conference on Artificial Intelligence,* St. Paul, Minnesota,1988.

[3] Daniel L. Dvorak, *Expert Systems for Monitoring and Control,* Report AI 87-55, Artificial Intelligence Laboratory, The University of Texas at Austin,1987.

[4] Kenneth D. Forbus, "Qualitative Process Theory," in *Qualitative Reasoning about Physical Systems,* D. Bobrow, ed., MIT Press, 1985.

[5] Johan de Kleer and John Seely Brown, "A Qualitative Physics Based on Confluences," in *Qualitative Reasoning about Physical Systems,* D. Bobrow, ed., MIT Press, 1985.

[6] Benjamin J. Kuipers, "Qualitative Simulation," *Artificial Intelligence,* **29**, 1986.

[7] Daniel L. Dvorak and Benjamin J. Kuipers, "Model-Based Monitoring of Dynamic Systems," submitted to *11th International Joint Conference on Artificial Intelligence,* Detroit, 1989.

[8] Richard J. Doyle, Suzanne M. Sellers, and David J. Atkinson, "Predictive Monitoring Based on Causal Simulation," *2nd NASA AI Forum,* 1987.

[9] Richard J. Doyle, "Hypothesizing Device Mechanisms: Opening Up the Black Box," Report TR-1047, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1988.

[10] Johan de Kleer and Brian C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence,* **32**, 1987.

[11] John H. Munson, "Robot Planning, Execution, and Monitoring in an Uncertain Environment," *2nd International Joint Conference on Artificial Intelligence,* 1972.

[12] Austin Tate, "Planning and Condition Monitoring in a FMS," Artificial Intelligence Applications Institute, University of Edinburgh, AIAI-TR-2, 1984.

[13] Carol A. Broverman and W. Bruce Croft, "Reasoning About Exceptions During Plan Execution Monitoring," *6th National Conference on Artificial Intelligence,* Seattle, 1987.

[14] Richard J. Doyle, Rajkumar S. Doshi, and David J. Atkinson, "Generating Perception Requests and Expectations to Verify the Execution of Plans,"*5th National Conference on Artificial Intelligence,* Philadelphia, 1986.

N90-29038

State-Based Scheduling: An Architecture for Telescope Observation Scheduling

Nicola Muscettola and Stephen F. Smith[1]

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper, we extend the applicability of constraint-based scheduling, a methodology previously developed and validated in the domain of factory scheduling, to problem domains that require attendance to a wider range of state-dependent constraints. We focus specifically on the problem of constructing and maintaining a short-term observation schedule for the Hubble Space Telescope (HST), which typifies this type of domain. We examine the nature of the constraints encountered in the HST domain, discuss system requirements with respect to utilization of a constraint-based scheduling methodology in such domains, and present a general framework for state-based scheduling.

1. Introduction

Many planning problems of practical importance involve the allocation of resources over time subject to a large and complex set of constraints. The construction of short-term observation schedules for the Hubble Space Telescope (HST), which is the domain of focus in the research reported in this paper, provides a representative example. HST observation scheduling involves the placement of several thousand exposures on a time line so as to satisfy a wide range of constraints relating to orbit characteristics, power and thermal balance requirements, instrument capabilities, viewing conditions, guidance requirements, and proposer specific restrictions and preferences. Scheduling under constraints, which is characteristic of most real-world scheduling problems, is extremely complex. Complexity derives not only from the diversity and number of constraints that must be attended to, but also from the fact that it is generally not possible to satisfy all constraints. In situations of conflicting objectives, appropriate compromises must be determined.

One approach to scheduling that has demonstrated an ability to effectively cope with a large and conflicting set of constraints is *constraint-based scheduling* [7, 17, 15]. This approach has been investigated and validated in the context of complex factory scheduling problems [15]. Constraint-based scheduling is an incremental problem solving methodology based on repeated analysis of the characteristics of problem constraints induced by the current partial solution (e.g. intervals of likely resource contention, relative flexibility of different activity time constraints, conflicts in the current partial

[1]This research has been sponsored in part by the National Aeronautics and Space Administration under contract # NCC 2-531 and The Robotics Institute.

schedule, etc.) as a means for structuring and exploring the underlying search space. It presumes an ability on the part of the problem solver to selectively reason from local perspectives, and analysis is concerned with subproblem formulation (i.e. which decisions to consider next and which scheduling criteria to emphasize). Commitments are thus made in an *opportunistic* manner (i.e. there is no a priori constraint on the order in which decisions are made). As specific commitments are made, current solution constraints are updated to reflect their consequences. This is a radical departure from traditional dispatch-based approaches to reasoning about efficient resource utilization over time (e.g. [16, 11]), wherein commitments are generated in a strict forward time order. It enables the scheduler to focus immediately on those decisions most critical to overall optimization of scheduling objectives as opposed to encountering them only after other restricting decisions have necessarily been committed to.

In this paper, we introduce a framework for *state-based scheduling*, which extends the applicability of this opportunistic scheduling methodology to domains, like the HST domain, where scheduling decisions must satisfy a wide variety of state-dependent constraints in addition to resource availability. Section 2 considers the nature of the short-term HST observation scheduling problem and the constraints that must be attended to. In Section 3, we consider HST problem constraints in light of the modeling assumptions that were made within the factory scheduling domain, and discuss implications with respect to use of an opportunistic scheduling methodology. In Section 4, we describe a representation and system architecture for state-based scheduling.

2. The HST Scheduling Problem

The HST is a sophisticated observatory due to be placed into low earth orbit in late 1989 and expected to have a lifetime of around 15 years. HST will offer new opportunities to the astronomical community, and contention for viewing time can be expected to be high. Generally speaking, the HST scheduling problem involves determination of execution times for observations specified in a set of previously accepted observation programs subject to a complex and conflicting set of constraints. As in [13], we presume a hierarchical decomposition of the overall problem over different time horizons, and we focus specifically on the short term scheduling problem (one week to one month) where all orbital constraints are known with certainty. Our intent in this section is to provide an indication of the diversity and nature of the problem constraints. The reader is referred to [12] for a more complete description.

An observation program accepted for execution consists of a set of observations designed to meet specific scientific objectives. The number of programs accepted over a given time horizon is expected to exceed the actual capabilities of the telescope. Some programs are thus designated as supplemental, and their inclusion in the schedule is not guaranteed. Within a given observation program, a variety of relationships among specific observations may be specified, including partial orderings on observations, separation constraints, temporal grouping constraints, coordinated parallel observations, same telescope orientation constraints, and conditional execution The observations in a given program may themselves be prioritized, and, in some cases preferences as to completion levels are specified (e.g. 25% minimally, 50% would be adequate, no more than 75%). Some programs specify observations intended to be executed in parallel with those of other programs if their specific constraints can be mutually satisfied (e.g. same telescope pointing position, sufficient power to operate required instruments).

The execution of a specific observation implies the simultaneous satisfaction of many constraints. Various proposer specified requirements (e.g. target, dark time requirements, time critical exposure), orbital characteristics (e.g. passage through the South Atlantic Anomaly, wherein observing is severely restricted), resource availability constraints (e.g. power) and operational constraints (e.g. pointing restrictions relative to the sun, bright and earth; spacecraft roll constraints) combine to delineate possible execution intervals for a given exposure activity. Some of these constraints can be selectively relaxed (e.g. roll constraints can be sometimes compromised - implying less energy from the solar arrays - provided sufficient time is subsequently spent on-roll to recharge the batteries). Execution of an exposure also requires an ability to establish the "state" specified by observing requirements. This typically requires the execution of sequences of auxiliary "setup" activities. The telescope may require repositioning to point at the target (called slewing), guide stars must be acquired and locked onto by the telescope's fine guidance sensors, the designated instrument and detectors must be brought to the appropriate configuration state, etc. Furthermore, specified communication requirements dictate the execution of additional communication activities. The execution of each of these supporting activities is subject to constraints that typically differ from those of the actual exposing activity (e.g. visibility of communication satellites, tape recorder capacity). Observations may be designated as interruptible (e.g. if the target is occulted for some portion of the telescope's orbit), necessitating the execution of additional setup activities (e.g. guide star reacquisition). Setup activities can often be performed in parallel (e.g. slewing while warming up the required instrument), and it is advantageous to do so as long as relevant constraints (e.g. required power) can be mutually satisfied.

Thus, the HST scheduling problem is one of maximizing the amount of science viewing time while attending as much as possible to the diverse preferences of specific observation programs and insuring feasibility with respect to the complex set of constraints surrounding operation of the telescope and execution of observations.

3. Implications for Constraint-Based Problem Solving

Characteristics of the HST scheduling problem, call into question some of the modeling assumptions that were possible in the factory scheduling domain. This, in turn, has implications with respect to providing an ability to generate and revise scheduling decisions in an opportunistic manner. This issue is considered below.

One broad distinction that can be drawn relative to the characteristics of factory scheduling and HST scheduling, is that factory scheduling problems are typically much less dominated by absolute temporal constraints than is the HST scheduling problem. There are of course deadlines in factory scheduling (and meeting them is very important), but these do not place rigid constraints on the execution of particular production activities. The point is that there is a certain degree of robustness in any factory schedule that is generated. Minor deviations from the schedule during its execution do not have a drastic effect on the overall performance of the factory (e.g. whether a given activity is performed 5 minutes ahead or behind schedule typically has little global impact).

A second distinction, owing more to the specific manufacturing domains we have addressed, concerns the level of interaction between the setup activities that must necessarily be performed to satisfy state-dependent constraints on production activities and HST observations respectively, and the presence or absence of constraints on the execution of these setup activities. In the manufacturing scheduling problems we have considered, such interactions have been minimal and setup activities themselves have been relatively unconstrained, allowing the prespecification and use of setup duration constraints[2].

These constraint characteristics have been exploited within our factory scheduling work to facilitate use of an opportunistic, constraint-based scheduling methodology. The OPIS factory scheduling system [17], which exhibits this capability, operates with respect to simplified assumptions regarding the state of resources over time, explicitly modeling only their available capacity, and assuming all other aspects of their state to be a function of the last activity performed. Resource setup activities are implicitly modeled as adjustments to the durations of activities that require them. These assumptions enable advance instantiation of the possible sequences of activities required to produce the set of production units that must be manufactured, and thus enable the scheduler to maintain an accurate characterization of current solution constraints [10].

In the HST domain, in contrast, it is simply not possible to operate under such modeling assumptions. The dominating presence of state-dependent constraints requires reasoning relative to an explicit model of the actual world state and the on-line expansion of sequences of activities to satisfy observation setup constraints. At the same time, given the overall size of the problem, such detailed reasoning can only be feasibly approached once some commitment has been made relative to where on the time line specific observations are to be placed. Thus, it is evident that analysis and opportunistic commitment with regard to specific observations must take place relative to approximate models of current solution constraints, and as such, these commitments can, at best, provide constraints on the actual decisions that must ultimately be taken. Such commitments must be subsequently refined so as to both insure their feasibility (i.e. that requisite activities can be accomplished in a manner consistent with the decision and the current partial schedule) and attend, as much as is possible, to their optimality (i.e. that the final placement of all constituent activities on the time line reflects relevant scheduling objectives). In the following section, we define a general scheduling framework that supports such decision-making.

4. A Generalized Scheduling Framework

In this section we introduce the general purpose framework that we are developing to solve the HST scheduling problem. We begin by introducing the main assumptions and conceptual primitives on which the architecture rests. First we discuss how the physical system over which the scheduler has to reason is represented. Then we discuss how we specify to the scheduler what it should accomplish on the physical system, both in terms of what to do and under what conditions. Finally, we describe the architecture, outlining three modules that constitute it.

4.1. Modeling the Physical System

Every scheduling problem is defined with respect to a physical system. Classical formulations of scheduling problems [8, 2] describe the physical system only in terms of two entities: **actions** and **resources**. For each resource the amount of *processing capacity* available over time is defined. The fundamental assumption is that when an action is executed, it consumes a fixed amount of capacity from a single resource for the course of its duration. The evolution of such a physical system is consistent if there is never an instant of time $\underline{t}$ at which the sum of the requests of processing capacity of each action in-process at time $\underline{t}$ exceeds the capacity available on the resource at that time.

[2]Note, however, that this is certainly not true of all manufacturing environments (e.g. an automated manufacturing cell)

The system description implied by this classical formulation is insufficient for scheduling problems, like the HST scheduling problem, where the execution of actions depends not only on resource availability. If we want to observe a target, for example, we need to insure that the target is visible throughout the observing action. It is conceptually incorrect to interpret a target as a resource and visibility as a processing capacity since it is never the case that the target loses any fraction of its visibility during an observation. The assumption of renewability (i.e. that capacity is required only for the duration of the action) is also problematic in many cases. For example, capacity on the on-board tape recorder is consumed by "write" actions and is not again available until the data is read out.

Our approach to the representation of the physical system is philosophically in line with that of [4, 9, 6]. In the following we will highlight the main characteristics of the corresponding description language; its complete description can be found in [14].

At any point in time we can describe the state of the system with a finite number of predicates. Each predicate represents one of the following:

- **actions**: these are transformations of the state of the system that have a known duration and are explicitly initiated by the executor of the schedule;
- **events**: these are transformations of the state of the system with fixed duration that are outside of the direct control of the executor of the schedule;
- **stable states**: these are reached after an action or an event has terminated. Their duration can depend on the occurrence of other actions or events occurring after their start time.

Let's consider some examples drawn from the HST domain:

The predicate:
LOCKED (*Target_X*)

represents a stable state. It will appear in the description of the state of the system whenever the telescope is pointing toward *Target_X* and is locked on it.

An example of an action is the predicate:
INSTRUMENT–STATE–TRANSITION (*Wf/Pc*, *StandBy*, *Operational*)

which represents the warmup transition on the *Wf/Pc* instrument that brings it from state *StandBy* to state *Operational*.

An example of event would be:
UNLOCKING (*Target_X*)

that express the fact that the telescope is losing its lock on *Target_X*. This event will start when the state of the system contains a predicate indicating that HST is locked on *Target_X* and a predicate indicating that *Target_X* has become not visible.

The basic temporal representation used to describe predicates associated with a temporal duration is the *time map* (TM), described in [3]. To each action, event and stable state is associated a **time token**, consisting of a triple $<P, t_{start}, t_{end}>$, where P is the corresponding predicate and t_{start} and t_{end} are nodes in the time map. All the nodes in a time map (except one) designate either the start time or the end time of a token; the exception is the node **ref** that represents the origin of the time axis. For each node in a TM we maintain two numbers $<d_{MAX}, D_{min}>$ representing respectively the maximum of the minimum distances from **ref** and the minimum of the maximum distances of the node from **ref**. The two couples associated with the nodes of a time token represent a generalization of a time bound as described in [10].

Time tokens are also organized along another dimension. Specific sets of predicates are associated with specific **state variables**. For example, the following formulas:

LOCKED (*?target*)
UNLOCKED (*?target*)
LOCKING (*?target*)
SLEWING (*?target_1*, *?target_2*)
UNLOCKING(*?target*)

constitute the descriptors of all possible values of the state variable **HST-pointing-status**. The basic constraint implied by a given state variable is that only one of its possible predicates can hold at any point in time. A state variable has a function similar to the *clipping constraints* described in [3].

The last thing we have to express in order to completely describe the physical system is its dynamics. This includes explicit representations of which predicates across state variables can hold simultaneously at any point in time, the preconditions of a predicate, etc. For example we have to express the fact that the telescope can be locking on a target only while the target is visible. This is expressed by saying that while the predicate *LOCKED* (*?target*) holds, the predicate *VISIBLE* (*?target*) must hold too, where the variable *?target* unifies with the same individual in both predicates. Another way to say this is that in any description of the behavior of the system over time, the presence of a time token $\mathbf{tk}_2$ of type *LOCKED* (*Target_X*) implies the presence of a time token $\mathbf{tk}_2$ of type *VISIBLE* (*Target_X*) that temporally contains it; contains has the same semantics as in [1]. A complete presentation of the description language of the system's dynamics can be found in [14].

4.2. Specification of a Scheduling Problem

The framework for describing the physics of the system presented in Section 4.1 gives us the possibility to express more general scheduling problems than those classically considered. In general, we can say that in order to specify a scheduling problem one has to formulate a set of **scheduling goals** and a set of **scheduling constraints**. In the following we specify what we intend with this terminology.

A **scheduling goal** is a specification of *what* we want the system to do. It takes the form of a predicate that has to hold in any solution. Generally speaking, scheduling goals include both actions to be executed and states to be achieved. In classical formulations of the scheduling problem [2], the latter type of goal has not been accorded full status (e.g. allowing expression of only required resource capacity). By contrast, a solution of to the HST scheduling problem (as well as many others) requires full treatment of both types of scheduling goals. Specification of an observation implies both the definition of actions to be executed (e.g. taking an exposure and communicating to Earth) and the definition of sets of stable states to be achieved (e.g. the viewing instrument and detector in operate mode, the telescope in the Earth's shadow, etc.). The description language presented in Section 4.1 provides a general framework for expression of both types of goals.

The second component of a scheduling problem is a specification of a set of **scheduling constraints**. An important class concerns *when* we want the system to reach the scheduling goals. This generally includes specification of both relative and absolute temporal restrictions on scheduling goals. With respect to this issue as well, classical formulations have typically imposed limiting assumptions (e.g. total orderings over the set of operations to be performed). In the HST domain, a variety of temporal restrictions on observations is possible, including parallel observations, partial orderings of sets of observations, and specific observation time windows. Representation of such temporal constraints is straightforward in the time map formalism. Figure 4-1 represents an observation of duration d that is constrained to start after time t_1 and to end before time t_2. Figure 4-2 represents an expose and communicate action sequence with no intervening temporal gap.

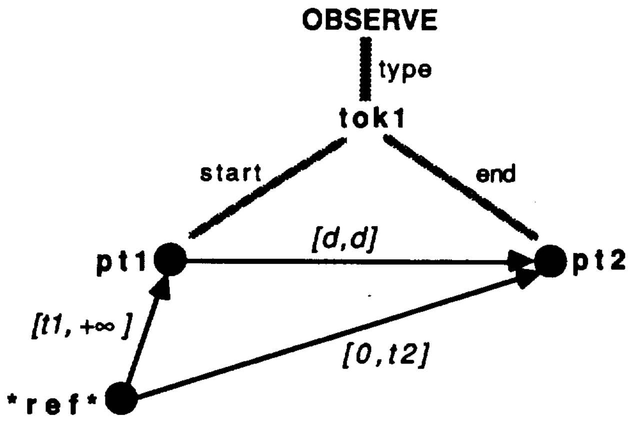


Figure 4-1: Representation of absolute temporal constraints

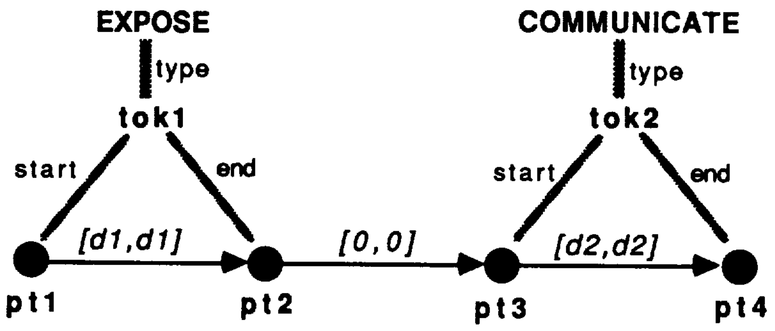


Figure 4-2: Representation of relative temporal constraints

Another class of scheduling constraints relates to objectives and preferences that we would like the system to satisfy to the extent possible. In some cases, such constraints may be defined relative to specific temporal restrictions, for example *"Execute action x as soon as possible after action y"*. In other cases, they define priority relationships among sets of scheduling goals. In the HST domain, for example, an observation program may designate preferences with respect to the number of observations that must be executed. The representation of such constraints has not been discussed in this paper, but we assume use of a utility-based formulation as in [7].

4.3. The Architecture

As in OPIS [17], the scheduler builds its schedule incrementally according to evolving characteristics of solution constraints. This is accomplished through the iterative application of three modules: a **Sub-Problem-Selector**, a **Planner**, and a **Reserver**. This process is outlined below.

The scheduler starts with a description of the expected evolution of the state of the physical system over time and an initial scheduling problem. This information is represented by two separate TMs. Namely:

- **Scheduling Problem TM (SPTM)**: This is a representation of the **scheduling goals** and **temporal scheduling constraints** that constitute the current scheduling problem.
- **System's Simulation TM (SSTM)**: This represents a complete deterministic simulation of the state of the system over time.

The first step of the iterative scheduling process is accomplished by the **Sub-Problem Selector** module. The role of this module is to opportunistically focus the attention of the system. Minimally, this involves selection of a sub-problem TM from the SPTM for solution relative to the full model of the underlying physical system. The introduction of additional scheduling constraints into the selected sub-problem TM is also possible (e.g. restricting attention to the interval between two previously achieved goals), if further constraining of the detailed problem solving effort is deemed appropriate. Decision-making at this level is based on analysis of the scheduling constraints associated with as yet unachieved scheduling goals in the SPTM. To this end, we assume that the consequences of scheduling commitments recorded in the SSTM (see below) are reflected back into the SPTM (similar to the manner in which operation time bounds are modified by resource unavailabilities in OPIS [10]). Since the focus of this module encompasses the entire scheduling problem, any consideration of tradeoffs relative to achievement of scheduling objectives necessary to support subproblem formulation must necessarily make use of approximate models of actual setup constraints (e.g. proximity of targets as a means for approximating slewing time in the HST domain).

The sub-problem TM determined by the sub-problem selector module forms the nucleus of a third type of TM:

- **Plan TM (PTM)**: It represents all the possible evolutions of the system deriving from the execution of a given set of actions that reaches the scheduling goals of a sub-problem TM under the specified scheduling constraints

The process of augmenting the sub-problem TM to form a complete PTM is performed by a **Planner** module: the complete description of the planning algorithm can be found in [14]. In synthesis, the planner keeps a set of planning goals (PGs); a planning goal is a specification of a token that has to be present in the PTM. Initially the set contains all the tokens that form the sub-problem TM. After selecting a PG from the set, the planner will expand it both backwards and forward. The backward expansion is analogous to the one performed in classical linear planning systems [5]. The forward expansion is equivalent to a forward simulation and it is performed in order to detect possible inconsistencies with the **reservations** (defined below) in the current SSTM. While processing the current PG, a new PG is generated and introduced into the set of current sub-goals if:

1. the new PG is pre-condition of the current PG;
2. the new PG is an effect of the current PG;
3. the new PG is has to hold in parallel with the current PG;

4. the new PG corresponds to a reservation that clashes with the current PG.

The new PGs in 1, 2 and 3 are directly obtainable from the System's dynamic description mentioned in Section 4.1. A plan is found when the set of pending PGs is empty.

The last step in the scheduling cycle is performed by the **Reserver** module. This selects a single start time for each of the actions in the PTM. The corresponding evolution of the state of the system is merged with the current SSTM, forming a new SSTM that solves the current scheduling sub-problem. The reserver has also to mark some tokens in the new SSTM as **reservations**, indicating that they need to be preserved in any further extension of the SSTM.

The scheduling cycle is repeated until either all scheduling goals in the SPTM have been achieved or it has been determined that it is not possible to achieve those that remain.

5. Summary

In this paper, we have described a scheduling framework that extends the applicability of an opportunistic scheduling methodology previously developed and validated in factory scheduling domains to problem domains where solutions must satisfy complex state-dependent constraints. We examined the characteristics of the HST observation scheduling problem, which is representative of this type of problem domain, pointing out the dominating presence of state-dependent constraints, the inadequacy of modeling assumptions that were possible in previous work, and the implications with respect to opportunistic scheduling. This led to the presentation of a representation and architecture for state-based scheduling, which we are currently developing to solve the HST observation scheduling problem. This framework enables complete treatment of state-dependent constraints while retaining the flexibility to incrementally construct schedules in an opportunistic manner.

Acknowledgements

The authors wish to thank Katia Sycara for helpful comments on an earlier draft of this paper.

References

[1] Allen, J.
Maintaining Knowledge about Temporal Intervals.
Communications ACM 26:832-843, 1983.

[2] Baker, K.R.
Introduction to Sequencing and Scheduling.
John Wiley and Sons, New York, 1974.

[3] Dean, T.L. and D.V. McDermott.
Temporal Data Base Management.
Artificial Intelligence 32:1-55, 1987.

[4] Dean, T.L., R.J. Firby, and D. Miller.
The FORBIN Paper.
Technical Report YALE/CSD/RR#550, Yale University, Dept. of Computer Science, July, 1987.

[5] Fikes, R.E.,P.E. Hart, and N.J. Nilsson.
Learning and Executing Generalized Robot Plans.
Artificial Intelligence 3:251-288, 1972.

[6] Forbus, K.D.
Introducing Actions into Qualitative Simulation.
Technical Report UIUCDCS-R-88-1452, Univeristy of Illinois at Urbana-Champaign, Dept. of Computer Science, August, 1988.

[7] Fox, M.S., and S.F. Smith.
ISIS: A Knowledge-Based System for Factory Scheduling.
Expert Systems 1(1):25-49, July, 1984.

[8] Garey, M.R. and D.S. Johnson.
Computers and Intractability.
W.H. Freeman and Company, San Francisco, 1979.

[9] Hogge, J.C.
TPLAN: A Temporal Interval-Based Planner with Novel Extensions.
Technical Report UIUCDCS-R-87-1367, Univeristy of Illinois at Urbana-Champaign, Dept. of Computer Science, September, 1987.

[10] LePape, C. and S.F. Smith.
Management of Temporal Constraints for Factory Scheduling.
In C. Rolland, M. Leonard, and F. Bodart (editors), *Proceedings IFIP TC 8/WG 8.1 Working Conference on Temporal Aspects in Information Systems (TAIS 87).* Elsevier Science Publishers, held in Sophia Antipolis, France, May, 1987.

[11] Miller, D.P.
Planning by Search Through Simulations.
Technical Report YALEU/CSD/RR#423, Yale University, Computer Science Dept., October, 1985.

[12] Miller, G., D. Rosenthal, W. Cohen, and M. Johnston.
Expert Systems Tools for Hubble Space Telescope Observation Scheduling.
In *Proceedings of the 1987 Conference on Artificial Intelligence Applications.* NASA Goddard Space Flight Center, 1987.

[13] Miller,G., M. Johnston, S. Vick, J. Sponsler, and K. Lindenmayer.
Knowledge-Based Tools for Hubble Space Telescope Planning and Scheduling: Constraints and Strategies.
In *Proceedings Goddard Conference on Space Applications of Artificial Intelligence.* NASA, May, 1988.

[14] Muscettola, N.
Incremental Temporal Planning.
Technical Report, Carnegie Mellon University, The Robotics Institute, forthcoming.

[15] Ow, P.S. and S.F. Smith.
Viewing Scheduling as an Opportunistic Problem Solving Process.
In R.G. Jeroslow (editor), *Annals of Operations Research 12.* Baltzer Scientific Publishing Co., 1988.

[16] Panwalker S.S. & Iskander W.
A Survey of Scheduling Rules.
Operations Research 25:45-61, 1977.

[17] Smith, S.F.
A Constraint-Based Framework for Reactive Management of Factory Schedules.
In M.D. Oliff (editor), *Intelligent Manufacturing.* Benjamin Cummings Publishers, 1987.

N90-29039

FOCUS OF ATTENTION IN AN ACTIVITY-BASED SCHEDULER

Norman Sadeh and Mark S. Fox

Computer Science Department and Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Earlier research in job shop scheduling has demonstrated the advantages of opportunistically combining order-based and resource-based scheduling techniques. In this paper we investigate an even more flexible approach where each activity is considered a decision point by itself. Heuristics to opportunistically select the next decision point on which to focus attention (i.e., *variable ordering* heuristics) and the next decision to be tried at this point (i.e., *value ordering* heuristics) are described that probabilistically account for both activity precedence and resource requirement interactions. Preliminary experimental results indicate that the variable ordering heuristic greatly increases search efficiency. While least constraining value ordering heuristics have been advocated in the literature [7], our experimental results suggest that other value ordering heuristics combined with our variable-ordering heuristic can produce much better schedules without significantly increasing search.

1. Introduction

We are concerned with the issue of how to opportunistically focus an incremental job shop scheduler's attention on the most critical decision points (*variable ordering* heuristics) and the most promising decisions at these points (*value ordering* heuristics) in order to reduce search and improve the quality of the resulting schedule.

So called order-based and resource-based scheduling techniques have been at the origin of several incremental scheduling algorithms. In an order-based approach, each order is considered a single decision point, i.e. orders are prioritized and scheduled one by one. In a resource-based approach, resources are rated according to their projected levels of demand. Resources are then scheduled one by one, starting with the ones that have the highest demand. Order-based scheduling has proven to be a viable paradigm in problems where slack (i.e. temporal precedence interactions) is the dominating factor. On the other hand, resource-based scheduling is likely to perform better in situations where resource contention (i.e. resource requirement interactions) is critical. Neither approach is perfect. Indeed a lot of real life scheduling problems contain a mix of critical orders and critical resources. In the past few years it has become clear that in order to perform well in a wider class of problems, schedulers need the ability to opportunistically switch from one approach to the other. The OPIS [15, 12, 16] scheduler was the first scheduler to combine both approaches. OPIS uses demand thresholds to identify bottleneck resources. Typically, when a bottleneck resource is detected, all activities requiring that resource and that have not yet been scheduled will be scheduled. When all bottleneck resources have been scheduled, OPIS switches to order-based scheduling. While in order scheduling mode, OPIS may detect the appearance of new bottleneck resources and switch back to its resource scheduling mode. Even such an approach has its shortcomings as the criticalities of the activities requiring a bottleneck resource or belonging to a critical order are not homogeneous, i.e. some of these activities may not be that critical. This consideration led us to the investigation of a new scheduling framework where the decision points are no longer entire resources or entire orders but instead where each activity is a decision point in its own right. Within this framework activity criticality is no longer determined via a sole bottleneck resource or via the sole order to which it belongs. Instead measures of activity criticality account for both temporal precedence interactions (i.e., so-called *intra-order* interactions [15]) and resource requirement interactions (i.e., so-called *inter-order* interactions). By simultaneously accounting for both types of interactions the approach is expected to opportunistically combine advantages of both order-based and resource-based scheduling techniques.

In this paper, we study one variable-ordering heuristic and three value-ordering heuristics for activity-based scheduling. A preliminary set of 38 scheduling problems was used to compare the performances of these heuristics. The experiments clearly indicate that the variable-ordering heuristic significantly reduces search. The comparison of the value-ordering heuristics when combined with the variable ordering heuristic suggests that a least constraining value ordering heuristic is not the only viable approach to maintain the amount of search at an acceptable level. Indeed some other heuristics produced much better schedules without significantly increasing search.

In the next section we describe our model of the job-shop scheduling problem. The following section gives an overview of the activity-based approach to scheduling that we are investigating, and introduces a probabilistic model to account for both intra-order and inter-order interactions. Section 4 presents a variable-ordering heuristic based on this probabilistic model. In section 5 we present three value-ordering heuristics: a least constraining heuristic, a hill-climbing heuristic, and an intermediate heuristic where values are rated according to the probability that they will remain available and that no better values will remain available. Preliminary experimental results are reported in section 6. Section 7 discusses these results. Section 8 contains some concluding remarks.

2. The Model

Formally, we will say that we have a set of N jobs (i.e. orders) to schedule. Each job has a predefined process plan that specifies a partial ordering among the activities (i.e. operations) to be scheduled. Each activity A_k $(1 \le k \le n)$ may require one or several resources R_{ki} $(1 \le i \le p_k)$, for each of which there may be several alternatives R_{kij} $(1 \le j \le q_{ki})$[1]. We will use st_k, et_k, and du_k to respectively denote A_k's start time, end time, and duration.

We view the scheduling problem as a constraint satisfaction problem (CSP).

The variables of the problem are the activity start times, the resources allocated to each activity, and possibly the durations of some activities. An activity's end time is defined as the sum of its start time and duration. Each variable has a bounded (finite or infinite) set of admissible values. For instance, the start time of an activity is always restricted at one end by the order release date and at the other end by the order latest acceptable completion time[2] according to the durations of the activities that precede/follow the activity in the process plan.

We differentiate between two classes of constraints: activity precedence constraints and resource capacity constraints. The activity precedence constraints are the ones defined by the process plans. Our model [14] accounts for all 13 of Allen's temporal constraints [1]. Capacity constraints restrict the number of reservations of a resource over any time interval to the capacity of that resource. For the sake of simplicity, we will assume in this paper that all resources are of unary capacity.

Additionally our model allows for preferences on activity start times and durations as well as on the resources that activities can use. Preferences are modeled with utility functions. These functions map each variable's possible values onto utilities ranging between 0 and 1. Preferences on activity start times and durations allow for the

[1]It is important to keep in mind that several activities may require the same resource. For instance if two activities A_1 and A_2 both require a unique resource which has to be R_1, we have $R_{111} = R_{211} = R_1$.

[2]This is not necessarily the order's due date.

representation of organizational goals such as reducing order tardiness, or reducing work-in-process (WIP) [3, 14]. Resource preferences are very useful to differentiate between functionally equivalent resources with different characteristics (e.g. difference in accuracy). In this paper we will assume that the sum of the utility functions defines a (separable) objective function to be maximized.

3. The Approach

3.1. An Activity-based Scheduler

In an activity-based approach, each activity is treated as an aggregate variable, or decision point, that comprises the activity's start time, its resources, and possibly its duration. The schedule is built incrementally by iteratively selecting an activity to be scheduled and a reservation for that activity (i.e. start time, resources and possibly duration). Every time a new activity is scheduled, new constraints are added to the initial scheduling problem, and propagated. If an inconsistency is detected during propagation, the system backtracks. The process stops either when all activities have been successfully scheduled or when all possible alternatives have been tried without success.

The efficiency of such an incremental approach critically relies on the order in which activities are scheduled and on the order in which possible reservations are tried for each activity. Indeed, because job-shop scheduling is NP-hard, search for a schedule may require exponential time in the worst case. Both empirical and analytical studies of constraint satisfaction problems reported in [6, 5, 13, 9, 10, 11, 17] indicate however that, on the average, search can significantly be reduced if always focused on the most critical decision points and the most promising decisions at these points. Such techniques are often referred to [2] as variable and value ordering heuristics.

In this paper we assume that critical activities are the ones whose good (overall) reservations are most likely to become unavailable if one were to start scheduling other activities first. In general reservations may become unavailable because of operation precedence constraints, because of resource capacity constraints, or because of combinations of both types of constraints. Clearly criticality measures are probabilistic in nature, as their computations require probabilistic assumptions on the values that will be assigned later on to each variable (i.e. the reservations that will later on be assigned to each unscheduled activity). In the next subsection we introduce a probabilistic framework that accounts for the interactions of start time, duration and resource preferences induced by both activity precedence and resource capacity constraints. We will use this model throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based scheduling.

3.2. A Probabilistic Framework to Account for Constraint Interactions

In this subsection we outline[3] a probabilistic model that we will use throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based job-shop scheduling. We justify the model by its ability to account for both intra-order and inter-order interactions and by its relatively low computational requirements. For the sake of simplicity, the formulas presented in this paper assume fixed duration activities. Similar formulas can be deduced when dealing with variable duration activities.

In our model a priori probability distributions are assumed for the possible start times and resources of each unscheduled activity. These probabilities are then refined to account for the interactions induced by the problem constraints (i.e. both intra-order and inter-order interactions). Finally the results of this propagation process are combined to identify critical activities and promising reservations for these activities. In their simplest form the a priori probability distributions are uniform. This amounts to assuming that, a priori, all possible reservations are

[3]A more detailed description can be found in [14].

equally probable. A slightly more sophisticated model consists in *biasing* the a priori distributions towards good values as defined by the utility functions [14]. Such biased distributions are expected to account better for value ordering heuristics that give more attention to higher utility values.

Once the a priori distributions have been built, they can be refined to account for the interactions of the problem constraints. In our model, the propagation is performed in two steps. The probability distributions are first propagated within each order, thereby accounting for intra-order interactions, and then across orders to account for inter-order interactions. Accounting simultaneously for both types of interactions seems indeed very difficult as much from a theoretical point of view as from a purely computational point of view. As a matter of fact the number of ways in which a set of activities can interact is combinatorial in the number of these activities[4]. Instead, by separately accounting for intra-order and inter-order interactions, one greatly reduces the amount of computation to be performed. The propagation results can always be further refined by iterating the propagation an arbitrary number of times.

Concretely, once the a priori distributions have been generated, our propagation process involves the following two steps:

1.
   a. The a priori start time probability distributions are refined to account for activity precedence constraints. The resulting (a posteriori) probability distributions associate to the possible start times of each activity the probability that these start times will be tried by the scheduler and will not result in the violation of an activity precedence constraint. These a posteriori start time distributions can be normalized to express that each activity will occur exactly once, and hence will start exactly once.
   b. For each resource requirement R_{ki} of each activity A_k, and for each resource alternative R_{kij} to fulfill R_{ki}, we compute the probabilistic demand D_{kij} of A_k for R_{kij} as a function of time. This probability is obtained using A_k's normalized a posteriori start time distribution and the a priori probability that A_k uses R_{kij} to fulfill its requirement R_{ki}. Hence $D_{kij}(t)$ represents the probabilistic contribution of A_k to the demand for R_{kij} at time t, if activity reservations were only checked for consistency with respect to the activity precedence constraints. Later on we will refer to $D_{kij}(t)$ as A_k's (probabilistic) individual demand for R_{kij} at time t.
2. Finally the individual demand densities of all activities are aggregated (i.e. summed at each point in time) to reflect the probabilistic demand for each resource in function of time. The resulting aggregate demand densities may get larger than one over some intervals of time, as the individual demand densities from which they originate have not been checked for consistency with respect to the capacity constraints. High demand for a resource over some time interval indicates a critical resource/time interval pair, which requires prompt attention from the scheduler. This is the basis of the variable-ordering heuristic presented in this paper.

More precise probabilities may be obtained by iterating the propagation process. One way to do so consists in computing for each possible activity reservation the probability that this reservation will be available and that no better reservation will be available. The availability probability of a reservation can be approximated by the probability that the reservation does not violate any activity precedence constraint or capacity constraint (see section 5 for details). These probabilities can then be combined into new a priori start time and resource probability distributions, and the propagation process can be carried out all over again. The experimental results that we report in this paper have all been obtained without iterating the propagation process. We are planning to perform similar experiments with probability distributions obtained after iterating the propagation a variable number of times.

[4]In any realistic problem, Monte Carlo simulation would indeed require tremendous amounts of computations if one were to simultaneously account for all the activities and all the constraints. This is because the probability of randomly generating a schedule for all the activities that satisfy all activity precedence and resource capacity constraints is in general extremely small.

Notations

In the remainder of the paper the following notations will be used:

- $p^{PRIOR}(st_k{=}t)$ will denote the a priori probability that A_k will be scheduled to start at time t,
- $p^{POST}(st_k{=}t)$ will be the a posteriori probability that A_k starts at time t, i.e. after accounting for activity precedence constraints,
- $p_N^{POST}(st_k{=}t)$ represents the same probability distribution after it has been normalized to express that A_k will start exactly once,
- $D_{kij}(t)$ represents A_k's individual demand for R_{kij} at time t, and
- $D^{aggr}_{R_{kij}}(t)$ will denote the aggregate demand for R_{kij} at time t.

4. ARR: A Variable Ordering Heuristic Based on Activity Resource Reliance

ARR, the variable ordering heuristic that we study in this paper, consists in looking for the resource/time interval pair that is the most contended for and the activity that relies most on the possession of that resource over that time interval. This activity is selected as the most critical one and hence is the next one to be scheduled.

The intuition behind this heuristic is the following. If activities that critically rely on the possession of highly contended resources were not scheduled first, it is very likely that, by the time the scheduler would turn its attention to them, the reservations that are the most appropriate for these activities would no longer be available.

The aggregate demand densities introduced in subsection 3.2 are used to identify the most demanded resource/time-interval pair. The activity that contributes most to the demand for the resource over the time interval (i.e. the activity with the largest individual demand for the resource over the time interval) is interpreted as the one that relies most on the possession of that resource. Indeed the total demand of an activity A_k for one of its resource requirements R_{ki} is equal to A_k's duration and is distributed over the different alternatives, R_{kij}, for that resource, and over the different possible times when A_k can be carried out. Consequently activities with a lot of slack or several resource alternatives tend to have fairly small individual demand densities at any moment in time. They rely less on the possession of a resource at any moment in time than activities with less slack and/or fewer resource alternatives. This allows ARR to account not only for inter-order interactions but also for intra-order interactions.

The advantage of this approach lies in its relative simplicity: look for the most critical resource/time-interval pair and select the activity that relies most on it. One may however contend that this heuristic does not consider slack as an independent component to activity criticality. Instead slack is only accounted for via resource contention. Another possible problem with this heuristic is that it only considers resource reliance with respect to the most contended resource. In general an activity A_k may require several resources R_{ki}. Rigorously, A_k's criticality should therefore account for each of these resources. It should account for the contention for each of the possible alternatives R_{kij} for these resources R_{ki}, and the reliance of A_k on the possession of each of these alternatives R_{kij}. Computation of such a criticality measure is likely however to be more expensive.

5. Three Value Ordering Heuristics

In the experiments that we ran, we considered the following three value-ordering heuristics:

5.1. LCV: A Least Constraining Value Ordering Heuristic

Least constraining value ordering heuristics are known for being very good at reducing search [6, 2]. Similar heuristics have also been proposed for scheduling, even when viewed as an optimization problem. [7], for instance, suggests the use of a least constraining value ordering heuristic as a primary criterion for selecting a reservation for an activity. The quality of the reservations is only used as a secondary criterion when there are several least constraining reservations to choose from. A similar heuristic is also outlined in [8]. The extremely small number of feasible solutions to a scheduling problem compared to the total number of schedules that one can possibly generate is what has made least constraining value ordering heuristics so attractive.

LCV is a least constraining value ordering heuristic where every reservation $\{\{st_k=t, R_{k1j_1}, R_{k2j_2},..., R_{kp_kj_{kp_k}}\}\}$, for an activity A_k, is rated according to the probability $RESERV\text{-}AVAIL(st_k=t, R_{k1j_1},...,R_{kp_kj_{p_k}})$ that it would not conflict with another activity's reservation, if one were to first schedule all the other remaining activities. The reservation with the largest such probability is interpreted as the least constraining one.

In our model, we express $RESERV\text{-}AVAIL(st_k=t, R_{k1j_1},...,R_{kp_kj_{p_k}})$ as the product of the probability that $st_k=t$ will not result in the violation of an activity precedence constraint and the conditional probability that each resource $R_{k1j_1}, R_{k2j_2},..., R_{kp_kj_{p_k}}$ will be available between t and $t+du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint:

$$RESERV\text{-}AVAIL(st_k=t, R_{k1j_1},...,R_{kp_kj_{p_k}})$$
$$= \frac{p^{POST}(st_k=t)}{p^{PRIOR}(st_k=t)} \times \prod_{R_{kij} \in \{R_{k1j_1},...,R_{kp_kj_{p_k}}\}} RESOURCE\text{-}AVAIL(R_{kij}, t, t+du_k)$$

where $RESOURCE\text{-}AVAIL(R_{kij}, t, t+du_k)$ is the conditional probability that R_{kij} will be available between t and $t+du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint.

We will approximate the (conditional) probability that R_{kij} will be available at some time τ for activity A_k with $\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)}$. When approximating $RESOURCE\text{-}AVAIL(R_{kij}, t, t+du_k)$, one has to be careful not to come up with too pessimistic an estimate. Indeed it is tempting to assume that the (conditional) probability that R_{kij} will be available for A_k between t and $t+du_k$ is given by the product of $\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)}$ over all possible start times τ between t and $t+du_k$. Depending on whether time is discrete or not, this product would be finite or infinite. In either case the approximation would be too pessimistic. Indeed this would be tantamount to supposing that the activities that contribute to $D^{aggr}_{R_{kij}}(\tau)$ have infinitely small durations, i.e. that these activities can possibly require R_{kij} at time τ without requiring it at $\tau-\delta\tau$ or $\tau+\delta\tau$. Instead, in order to account for the duration of these activities, we will assume that each resource R_{kij} is subdivided into a sequence of buckets of duration $AVG(du)$, where $AVG(du)$ is the average duration of the activities competing for R_{kij}. Consequently $RESOURCE\text{-}AVAIL(R_{kij}, t, t+du_k)$ is given by the probability that A_k can secure a number of buckets equal to its duration, i.e.:

$$RESOURCE\text{-}AVAIL(R_{kij}, t, t+du_k) = \{AVG(\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)})\}^{\frac{du_k}{AVG(du)}}$$

where $AVG(\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)})$ is simply the average of $\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)}$ taken between t and $t+du_k$.

5.2. HC: A Hill-Climbing Value Ordering Heuristic

The second value ordering heuristic that we tested simply consists in ranking an activity's possible reservations according to their utilities, i.e. preferences. Reservations with the highest preferences are the first ones to be tried.

5.3. INT: An Intermediate Value Ordering Heuristic

Our third value ordering heuristic combines features from the previous two. Each reservation is rated according to the probability that it would be available and that no better reservation would be available, if one were to first schedule all the other remaining activities.

6. Preliminary Experimental Results

A testbed was implemented that allows for experimentation with a variety of variable and value ordering heuristics based on the probabilistic framework described in subsection 3.2. The system is implemented in Knowledge Craft running on top of Common Lisp, and can be run either on a MICROVAX 3200 or on a VAX 8800 under VMS.

We performed some preliminary experiments to evaluate the four heuristics presented in this paper. These experiments were run on a set of 38 scheduling problems, involving between 3 and 5 orders and a total number of activities ranging between 10 and 20. The problems involved 3 or 4 resources. They involved only activities with a unique resource requirement (R_{k1}), for which there were one or several alternatives (R_{k1j}). Problems with both equally preferred and non equally preferred resource alternatives were included. The scheduling problems were built to reflect a variety of demand profiles: localized bottlenecks at the beginning, middle, and end of the problem span, global bottlenecks spanning the whole duration of the scheduling problems, and auxiliary bottlenecks were all included. Three different types of start time utility functions were allowed: all start times (between the earliest and latest start times) are equally preferred, late start times are preferred, and triangular start utility functions with a peak corresponding to the due date (minus the duration of the activity). Triangular utility functions were only assigned to the last activities of some orders. Time was discretized and a granularity equal to the third of the smallest activity duration was used. A discrete version of the formulas presented in this paper was used to compute the necessary probability distributions. The probabilities were computed using biased a priori probability distributions obtained by normalizing the utility functions over the domain of possible values of each variable. The granularity of the time intervals used for the ARR variable ordering heuristic varied from one resource to the other and was selected to be equal to the duration of the shortest activity requiring the resource.

Preliminary Experimental Results					
	RAND &HC	RAND &LCV	ARR &HC	ARR &LCV	ARR &INT
Search Efficiency	< 0.47 (> 0.27)	1.00 (0.00)	0.96 (0.05)	1.00 (0.00)	1.00 (0.00)
Schedule Value	not available	0.52 (0.08)	0.68 (0.06)	0.54 (0.06)	0.64 (0.05)

Figure 6-1: Average search efficiencies and schedule values for 5 combinations of variable and value ordering heuristics run on a preliminary set of 38 scheduling problems. The standard deviations appear between parentheses.

The experiments were measured along two dimensions: *search efficiency* (i.e. number of operations to schedule over number of search states generated) and *global utility* of the solution as defined by a normalized objective function. The normalized objective functions were built so that the best possible schedules that could be built without checking for constraint violation would have a global value of 1. There was no guarantee however that a *feasible* schedule with global value of 1 could be built. In the ideal case the search would be performed without backtracking, and the number of search states generated would be equal to the number of activities to schedule (i.e. efficiency of 1). The quality of the schedules is more difficult to assert as the value of the objective function for the optimal schedule varied from one problem to the other and was in most cases smaller than 1. For this reason the values of the schedules should not be viewed as absolute measures. Instead they should only be used to compare the relative performances of the combinations of heuristics that we tried.

The table in Figure 6-1 reports the average search efficiencies and schedule values obtained with five combinations of variable and value ordering heuristics (for a total of 190 experiments). Standard deviations are provided between parentheses. RAND denotes a random variable ordering heuristic, where the next activity to be scheduled is selected at random from the remaining unscheduled activities. Search was stopped when it would require more than 50 search states. For RAND&HC, this cutoff rule had to be used in 12 of the 38 experiments. It did not have to be used for any of the other heuristics. The average search efficiency of RAND&HC is therefore even worse than 0.47. Because search did not terminate in almost a third of the runs with RAND&HC, no good estimate of the value of the schedules produced by this heuristic could be obtained.

7. Discussion

The results reported in Figure 6-1 clearly indicate the importance of a good variable ordering heuristic to increase search efficiency (e.g. ARR&HC vs. RAND&HC). They also indicate that a least constraining value ordering heuristic can make up for a poor variable ordering heuristic (e.g. RAND&HC vs. RAND&LCV and RAND&LCV vs. ARR&LCV). In the examples that we ran the ARR variable ordering heuristic and the LCV value ordering heuristics both contributed to limit search. The quality of the schedules produced by LCV is however very poor when compared to the other two value ordering heuristics (ARR&HC or ARR&INT vs. ARR&LCV). In particular ARR&INT performed as well as ARR&LCV as far as the efficiency of the search is concerned (neither of them had to backtrack in any of the 38 examples) but produced much better schedules. Even a simple value ordering heuristic like HC resulted in very little amount of backtracking when coupled with our variable ordering heuristic (see ARR&HC). Overall HC produced the best schedules although it required more search than INT and LCV, when coupled with ARR. There seems therefore to be a tradeoff between the amount of search performed and the quality of the resulting solution. If little time is available to come up with a solution the most promising values may be the least constraining ones as they are the least likely to result in backtracking. On the other hand, when there is more time available, one may consider looking at riskier values if they are likely to produce better solutions. A value ordering heuristic could accordingly be designed that accounts for the time available to find a schedule.

8. Concluding Remarks

In this paper, we have investigated an activity-based approach to scheduling. Because of its greater flexibility, such an approach is expected to allow for the construction of better schedules than approaches using order-based or resource-based scheduling or even combinations of the two. The price to pay for this flexibility is the potential overhead involved in the selection of the next decision point on which to focus attention. While order-based and resource-based scheduling typically require only the computation of criticality measures for each order or resource in the system, an activity-based scheduling approach may potentially require the computation of similar measures for each of the activities to be scheduled. In the simplest scenario, these measures need moreover to be recomputed every time a new activity has been scheduled. It is therefore important that the computation of these measures can be performed at a relatively cheap computational cost. One may also consider scenarios where several activities are scheduled before new criticality measures are computed.

We have presented a probabilistic framework that successively accounts for both intra-order and inter-order interactions. Although such a two-step propagation involves a slight loss of precision in the way it accounts for interactions, it has the advantage of having a relatively low computational cost [14]. More accurate probability distributions may always be obtained by iterating the propagation process. Our probabilistic framework allows for the definition of a variety of variable and value ordering heuristics. In this paper, we have studied a simple variable-ordering heuristic, ARR, that looks for the most contended resource/time interval pair and the activity that relies the most on the possession of that resource/time interval pair. Preliminary experiments with the heuristic indicate that it greatly contributes to increasing search efficiency. Additionally our experiments with three value ordering heuristics seem to indicate that least constraining value odering heuristics such as the one advocated in [7] may not be the only viable way to maintain search at an acceptable level. Instead, in our experiments, other value ordering heuristics, when coupled with our variable ordering heuristic, produced much better schedules without significantly increasing search.

Our variable ordering heuristic is not perfect. For instance it measures activity criticality only with respect to one resource (the most contended resource/time interval pair). While the heuristic performed well in problems where each activity requires only one resource (for which there may or may not be alternatives), it may not be as effective for activities requiring several resources. We are currently looking at other variable ordering and value ordering heuristics. We are also pursuing our experiments with the heuristics presented in this paper. In particular we still have to study the behavior of these heuristics on larger scheduling problems (i.e. more than 100 activities).

Our long term interest is in the identification of a set of (texture) measures characterizing the search space, that can be used to both structure and guide search in that space. Measures of variable criticality (variable ordering heuristics) and estimates of value goodness (value ordering heuristics) are examples of such measures [4].

Acknowledgement

This research has been supported, in part, by the Defense Advance Research Projects Agency under contract #F30602-88-C-0001, and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

References

[1] J.F.Allen.
Towards a General Theory of Action and Time.
Artificial Intelligence 23(2):123-154, 1984.

[2] Rina Dechter and Judea Pearl.
Network-Based Heuristics for Constraint Satisfaction Problems.
Artificial Intelligence 34(1):1-38, 1988.

[3] M. Fox.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1983.

[4] Mark S. Fox, Norman Sadeh, and Can Baykan.
Constrained Heuristic Search.
Technical Report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1989.
Working Paper.

[5] E.C. Freuder.
A Sufficient Condition for Backtrack-free Search.
Journal of the ACM 29(1):24-32, 1982.

[6] Robert M. Haralick and Gordon L. Elliott.
Increasing Tree Search Efficiency for Constraint Satisfaction Problems.
Artificial Intelligence 14(3):263-313, 1980.

[7] N.P. Keng, D.Y.Y. Yun and M. Rossi.
Interaction-sensitive planning system for job-shop scheduling.
Expert Systems and Intelligent Manufacturing :57-69, 1988.

[8] Nicola Muscettola and Stephen Smith.
A Probabilistic Framework for Resource-Constrained Muti-Agent Planning.
In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 1063-1066. 1987.

[9] B.A. Nadel.
The General Consistent Labeling (or Constraint Satisfaction) Problem.
Technical Report DCS-TR-170, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[10] B.A. Nadel.
Three Constraint Satisfaction Algorithms and Their Complexities: Search-Order Dependent and Effectively Instance-specific Results.
Technical Report DCS-TR-171, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[11] B.A. Nadel.
Theory-based Search-order Selection for Constraint Satisfaction Problems.
Technical Report DCS-TR-183, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[12] Peng Si Ow.
Experiments in Knowledge-based Scheduling.
Technical Report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1986.

[13] Paul W. Purdom, Jr.
Search Rearrangement Backtracking and Polynomial Average Time.
Artificial Intelligence 21:117-133, 1983.

[14] N. Sadeh and M.S. Fox.
Preference Propagation in Temporal/Capacity Constraint Graphs.
Technical Report CMU-CS-88-193, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 1988.
Also appears as Robotics Institute technical report CMU-RI-TR-89-2.

[15] Stephen F. Smith and Peng Si Ow.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1013-1015. 1985.

[16] Stephen F. Smith, Peng Si Ow, Claude Lepape, Bruce McLaren, Nicola Muscettola.
Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans.
In *Proceedings 1986 SME Conference on AI in Manufacturing*, pages 123-137. 1986.

[17] Harold S. Stone and Paolo Sipala.
The average complexity of depth-first search with backtracking and cutoff.
IBM Journal of Research and Development 30(3):242-258, 1986.

REASONING UNDER UNCERTAINTY

N90-29040

A BOLTZMANN MACHINE FOR THE ORGANIZATION OF INTELLIGENT MACHINES

Michael C. Moed and George N. Saridis

Center for Intelligent Robotic Systems for Space Exploration (CIRSSE)
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

1. INTRODUCTION

In our present technological society, there is a major need to build machines that would execute intelligent tasks operating in uncertain environments with minimum interaction with a human operator. Although some designers have built smart robots, utilizing heuristic ideas, there is no systematic approach to design such machines in an engineering manner.

Recently, cross-disciplinary research from the fields of computers, systems, AI and information theory has served to set the foundations of the emerging area of the Design of Intelligent Machines (Saridis, Stephanou 1977).

Since 1977 Saridis has been developing a novel approach, defined as Hierarchical Intelligent Control, designed to organize, coordinate and execute anthropomorphic tasks by a machine with minimum interaction with a human operator. This approach utilizes analytical (probabilistic) models to describe and control the various functions of the Intelligent Machine structured by the intuitively defined principle of Increasing Precision with Decreasing Intelligence (IPDI) (Saridis 1979).

This principle, even though resembles the managerial structure of organizational systems (Levis 1988), has been derived on an analytic basis by Saridis (1988). The impact of this work is in the engineering design of intelligent robots, since it provides analytic techniques for universal production (blueprints) of such machines.

The purpose of the paper is to derive analytically a Boltzmann machine suitable for optimal connection of nodes in a neural net (Fahlman, Hinton, Sejnowski, 1985). Then this machine will serve to search for the optimal design of the Organization level of an Intelligent Machine.

In order to accomplish this, some mathematical theory of the intelligent machines will be first outlined. Then some definitions of the variables associated with the principle, like machine intelligence, machine knowledge, and precision will be made (Saridis, Valavanis 1988). Then a procedure to establish the Boltzmann machine on an analytic basis will be presented and illustrated by an example in designing the organization level of an Intelligent Machine. A new search technique, the Modified Genetic Algorithm, is presented and proved to converge to the minimum of a cost function. Finally, simulations will show the effectiveness of a variety of search techniques for the Intelligent Machine.

2. THE MATHEMATICAL THEORY OF INTELLIGENT CONTROLS

In order to design intelligent machines that require for their operation control system with intelligent functions such as simultaneous utilization of a memory, learning, or multilevel decision making in response to "fuzzy" or qualitative commands, Intelligent Controls have been developed by Saridis (1977, 1983). They utilize the results of cognitive systems research effectively with various mathematical programming control techniques (Birk & Kelley, 1981).

The theory of Intelligent Control systems, proposed by Saridis (1979) combines the powerful high-level decision making of the digital computer with advanced mathematical modeling and synthesis techniques of system theory with linguistic methods of dealing with imprecise or incomplete information. This produces a unified approach suitable for the engineering needs of the future. The theory may be thought of as the result of the intersection of the three major disciplines of Artificial Intelligence, Operations Research, and Control Theory. This research is aimed to establish Intelligent Controls as an engineering discipline, and it plays a central role in the design of Intelligent Autonomous Systems.

Intelligent control can be considered as a fusion between the mathematical and linguistic methods and algorithms applied to systems and processes. In order to solve the modern technological problems that require control systems with intelligent functions such as simultaneous utilisation of a memory, learning, or multilevel decision making in response to "fuzzy" or qualitative commands. Intelligent Control is the process of implementation of an Intelligent Machine and would require a combination of "machine intelligent functions" for task organisation purposes with system theoretic methods for their execution.

The control intelligence is hierarchically distributed according to the Principle of Increasing Precision with Decreasing Intelligence (IPDI), evident in all hierarchical management systems. They are composed of three basic levels of controls even though each level may contain more than one layer of tree-structured functions (Figure 1):

1. The organisation level.
2. The coordination level.
3. The execution level.

The Organisation Level is intended to perform such operations as planning and high level decision making from long term memories. It may require high level information processing such as the knowledge based systems encountered in Artificial Intelligence. These require large quantities of knowledge processing but require little or no precision.

The functions involved in the upper levels of an intelligent machine are imitating functions of human behavior and may be treated as elements of knowledge-based systems. Actually, the activities of planning, decision making, learning, data storage and retrieval, task coordination, etc. may be thought of as knowledge handling and management. Therefore, the flow of knowledge in an intelligent machine may be considered as the key variable of such a system.

Knowledge flow in an intelligent machine's organisation level represents respectively:

1. Data Handling and Management.
2. Planning and Decision performed by the central processing units.
3. Sensing and Data Acquisition obtained through peripheral devices.
4. Formal Languages which define the software.

Subjective probabilistic models or fuzzy sets are assigned to the individual functions. Thus, their entropies may be evaluated for every task executed. This provides an analytical measure of the total activity.

Artificial Intelligence methods also applicable for the processing of knowledge and knowledge rates of the organisation level of an intelligent machine have been developed by Meystel (1985) and his colleagues.

The Coordination Level is an intermediate structure serving as an interface between the organisation and execution level.

It is involved with coordination, decision making and learning on a short term memory, e.g., a buffer. It may utilise linguistic decision schemata with learning capabilities defined in Saridis and Graham (1984), and assign subjective probabilities for each action. The respective entropies may be obtained directly from these subjective probabilities.

The Execution Level executes the appropriate control functions. Its performance measure can also be expressed as an entropy, thus unifying the functions of an "intelligent machine".

Optimal control theory utilises a non-negative functional of the states of a system in the states space, and a specific control from the set of all admissible controls,to define the performance measure for some initial conditions $(x(t), t)$, representing a generalised energy function. Minimisation of the energy functional yields the desired control law for the system.

For an appropriate density function $p(x, u(x,t), t)$ satisfying Jaynes' Maximum entropy principle (1957), it was shown by Saridis (1988) that the entropy $H(u)$ for a particular control action $u(x,t)$ is equivalent to the expected energy or cost functional of the system. Therefore, minimisation of the entropy $H(u)$ yields the optimal control law of the systems.

This statement establishes equivalent measures between information theoretic and optimal control problems and unifies both information and feedback control theories with a common measure of performance. Entropy satisfies the additive property, and any system composed of a combination of such subsystems can be optimised by minimising its total entropy. Information theoretic methods based on entropy may apply (Conant 1976).

Since all levels of a hierarchical intelligent control can be measured by entropies and their rates, then the optimal operation of an "intelligent machine" can be obtained through the solution of mathematical programming problems.

The various aspects of the theory of hierarchically intelligent controls may be summarized as follows:

The theory of intelligent machines may be postulated as the mathematical problem of finding the right sequence of decisions and controls for a system structured according to the principle of increasing precision with decreasing intelligence (constraint) such that it minimizes its total entropy.

The above analytic formulation of the "intelligent machine problem" as a hierarchically intelligent control problem is based on the use of entropy as a measure of performance at all the levels of the hierarchy. It has many advantages because of the tree-like structure of the decision making process, and brings together functions that belong to a variety of disciplines.

3. KNOWLEDGE FLOW AND THE PRINCIPLE OF IPDI

The concept of entropy used in this paper may be generalized if one introduces theory of evidence for the cases that Intelligent Machines are endowed with judgment, a very human property.

The general concepts of Intelligent Control Systems are the fundamental notions of Machine Intelligence, Machine Knowledge, its Rate and Precision. The definitions useful in order to derive the principle of IPDI are presented in (Saridis, Moed 1988).

Analytically, the relations may be summarized as follows:

Knowledge (K) representing a type of information may be represented as

$$K = -\alpha - ln p(K) = (\text{Energy}) \tag{1}$$

where $p(K)$ is the probability density of Knowledge.

From equation (1) the probability density function $p(K)$ satisfies the following expression in agreement with Jaynes' principle of Maximum Entropy (1957):

$$p(K) = e^{-\alpha-K}; \quad \alpha = ln \int_X e^{-K} dx \tag{2}$$

The Rate of Knowledge R which is the main variable of an intelligent machine with discrete states is

$$R = \frac{K}{T} = (\text{Power})$$

It was intuitively thought (Saridis 1983), that the Rate of Knowledge must satisfy the following relation which may be thought of expressing the principle of Increasing Precision with Decreasing Intelligence

$$(MI) : (DB) \longrightarrow (R) \tag{3}$$

A special case with obvious interpretation is, when R is fixed, machine intelligence is largest for a smaller data base e.g. complexity of the process. This is in agreement with Vamos' theory of Metalanguages (1986).

It is interesting to notice the resemblance of this entropy formulation of the Intelligent Control Problem with the ε-entropy formulation of the metric theory of complexity originated by Kolomogorov (1956) and applied to system theory by Zames (1979). Both methods imply that an increase in Knowledge (feedback) reduces the amount of entropy (ε-entropy) which measures the uncertainty involved with the system.

An analytic formulation of the above principle has been derived from simple probabilistic relation among the Rate of Knowledge, Machine Intelligence and the Data Base of Knowledge. The entropies of the various functions come naturally into the picture as a measure of their activities.

4. THE DESIGN OF THE ORGANIZATION LEVEL OF AN INTELLIGENT MACHINE AS A BOLTZMANN MACHINE

In the current literature of parallel architectures for Machine Intelligence, the Boltzmann machine represents a powerful, neural network based architecture that allows efficient searches to optimally obtain the combination of certain hypotheses of input data and constraints (Fahlman, Hinton, Sejnowski 1985).

The Boltsmann architecture may be interpreted as the machine that searches for the optimal interconnection of several nodes (neurons) representing different primitive events in order to produce a string defining an optimal task. Such a device may prove extremely useful for the design of the Organisation Level of an Intelligent Machine (Saridis, Valavanis 1988) (Figure 2).

We associate the state of each node with a binary random variable $x_i = \{0,1\}$, with a priori probabilities $p(x_i = 1) = p_i$, $p(x_i = 0) = 1 - p_i$, where 1 represents the firing of neuron i, and 0 indicates neuron i idle. The state vector of the network, $X = \{x_1, x_2, \dots, x_i, \dots, x_n\}$ is an ordered set of 0's and 1's describing the state of the machine in terms of firing/idle nodes, for an n node machine. The neurons of the machine can be visible, or hidden (Hinton, Sejnowksi 1986). It is possible to extract the string of primitive events representing the optimal task by examining the state vector of the visible nodes in the network in steady state response to a given input.

The standard formulation of the Boltsmann machine uses Energy as a cost function which is minimised to find the optimal state of the machine. However, in (1) we defined knowledge as a form of Energy. This is not the function to be minimised in the Intelligent Machine. Instead, we will be minimising the Energy of Flow of Knowledge (F), which is the amount of knowledge which must flow through the machine in order to accomplish a particular task. This is found by:

$$F = R * T \tag{4}$$

where T is the total time of knowledge flow. By minimising F, the Intelligent Machine reduces the amount of Energy required to make a decision.

5. ENTROPY AS A MEASURE OF UNCERTAINTY

Entropy is used as a measure of uncertainty in the intelligent machine. The entropy manifests itself in the interaction and interconnection of nodes in the network. We can define energy of flow of knowledge into node i by:

$$F_i = \alpha_i - \frac{1}{2}\sum_j w_{ij}x_jx_i \tag{5}$$

and the probability the machine is in a state where Energy $= F_i$ by:

$$P(F_i) = e^{-\alpha_i + \frac{1}{2}\sum_j w_{ij}x_jx_i} \tag{6}$$

where:

w_{ij} is the interconnection weight between nodes i and j

$w_{ii} = 0$

α_i is a probability normalising factor which insures $.5 \leq P(F_i) \leq 1$

Unlike the Boltsmann machine, this formulation does not remove α_i when $x_i = 0$. Instead, the machine operates from a base entropy level which it tries to reduce.

By bounding $P(F_i)$ by 0.5, we find that the entropy of being in a state where Energy $= F_i$ increases as F_i increases. In other words, as the Energy increases, the uncertainty increases as well.

The Entropy of Knowledge Flow in the machine can be formulated as:

$$H(F) = -\sum_i P(F_i) ln\{P(F_i)\}$$

Therefore:

$$H(F) = \sum_i (\alpha_i - \frac{1}{2}\sum_j w_{ij}x_ix_j)e^{-\alpha + \frac{1}{2}\sum_j w_{ij}x_ix_j}$$

6. SEARCH TECHNIQUES FOR THE INTELLIGENT MACHINE

Three random search techniques are compared here which may be used to find the minimum entropy in the Organisation Level of an intelligent machine. By examining the active visible neurons in the minimum entropy state of the network, one can determine the sequence of primitive events which produce a string

[illegible]

defining an optimal task for an intelligent machine. The techniques presented here allow escape from local entropy minima, which lead to incorrect task decisions, by randomly selecting states while searching for the global entropy minimum.

6.1 A Genetic Algorithm Search Technique

A technique which minimises a system cost funciton is the Genetic Algorithm (Holland 1975). In contrast to other random search techniques, the Genetic Algorithm (GA) maintains a population of points in the space while searching for the optimum.

Here we present a modified GA which will converge in probability to the minimum cost. The standard GA has been changed by inserting spacer steps of an algorithm which is known to converge in probability, Expanding Subinterval Random Search.

Spacer steps are defined as follows: Suppose B is an algorithm which together with a descent function Z and solution set T converges in probability. We can define an algorithm C by $C(x) = \{y : Z \leq Z(x)\}$. In other words, C applied to x can give any point so long as it does not increase the value of Z, the current cost. B represents the spacer step, and the complex process between spacer steps is C. Thus, the overall process amounts to repeated applications of the composite algorithm CB. CB will converge in probability if B is repeated infinitely often and C does not increase the value of the current cost (Luenberger 1984).

We introduce the concept of "immigration" to imbed ESRS into GA. Infinitely often, we insert a randomly generated point into the GA search which forms the spacer step. The frequency of insertion is called the "immigration rate." By changing the "immigration rate," the algorithm adjusts its focus from global to local searches. This rate may be fixed dependent on the complexity of the search space, or may vary while the search is in progress. A high "immigration rate" will force random search. A low rate will cause the GA. Parallels can be drawn to Simulated Annealing which starts as a near random search, and eventually becomes gradient descent. For the modified GA, the "immigration rate" is analogous to thermal energy in Simulated Annealing. The modified algorithm described in detail below converges in probability to the minimum cost.

6.1.1 Standard Genetic Algorithm

In general, each point in the space is represented by a binary string and has an associated cost dictated by the system cost function for that point. Since the makeup of the population is changed each iteration to emphasise members (points) which minimise the cost function, a near-uniform population will develop corresponding to a local minima in the cost function.

The following notation is used:

$$\begin{aligned}
P &= \text{population of members (points)}\\
P' &= \text{new population of members}\\
|P| &= \text{number of members in } P\\
P_k &= \text{kth member of the population } P\\
P_k(m) &= \text{mth bit of } P_k\\
J_K &= \text{Cost of } P_k\\
S_k &= \text{probability of member } k \text{ being selected from current population}\\
J_{\max} &= \text{max cost of any possible string in } P\\
n &= \text{length of } P_k \text{ in bits}
\end{aligned}$$

Each iteration of the search algorithm proceeds as follows:

```
Repeat:
  1. Compute $J_k, \forall P_k \epsilon P$
  2. Let $J'_k = J_{\max} - J_k, \forall k$. Compute $S_k = \frac{J'_k}{\sum_i J'_i}, \forall k$.
     Repeat:
     3.1 Randomly select $P_j, P_k$ from $P$ based on $S_j, S_k$.
     3.2 Randomly generate an index $i$ between 1..$n$. Exchange the right string halves of
         $P_j, P_k$ (i.e., $P'_j(i..n) = P_k(i..n)$ and $P'_k(i..n) = P_j(i..n)$). This is called "crossover" or "mating."
```

```
    3.3 Place $P'_j, P'_k$ in $P'$. Return $P_j, P_k$ to $P$
        until $|P'| = |P|$.
4. Set $P = P'$
until $P_k \epsilon P$ and $P_k$ has minimum cost.
```

In an attempt to prevent population convergence on a local minima ("premature convergence"), a "mutation" operator is added to the system. With a new generation of the population, each bit of every member has a small probability of inverting. The inversion adds diversity to the population and promotes search in previously unexplored regions of the space in an attempt to find the global cost minimum.

Particular aspects of this algorithm make it a powerful search tool. The "crossover" mechanism forces search on an n-dimensional hypercube by discovering and promoting particular substrings (called "building blocks") which perform well. These "building blocks" combine to discover the topology of the search space, which may not be known initially. Since the algorithm uses a population of points, many planes of the hypercube can be searched at once, leading to "implicit parallelism." Further, since members within a population are independent, a new population may be formed by "mating" in parallel. Steps 3.1-3.3 can be blocked together and generate two new members in parallel with other "mating" blocks. These features as well as others are described in depth in (Goldberg 1989). Applications of this algorithm are presented in (De Jong 1975, Grefenstette et al 1985, Davis and Coombs, 1987).

Heuristic algorithms within GA have been developed to avoid convergence at local minima (Maldin 1984, Suh and Van Gucht 1987). The "SIGH" system (Ackley 1987) uses active and passive subpopulations to escape local minima. When particular members of the population are performing poorly, they become passive until the active subpopulation converges. If this convergence is premature, the passive members are activated, bringing diversity and new structure to the search.

6.1.2 Modified Genetic Algorithm

Unfortunately, many of the heuristically driven GA searches perform well for a small set of functions, and prematurely converge for functions outside that set. However, it can be shown that under certain conditions, the GA will converge in probability to the global minimum of the cost function.

The conditions are as follows:

1. Instead of (or in addition to) the "mutation" operator, an "immigration" operator is used. Introduce a randomly generated member P_i to P' every M populations for some integer $M > 0$.
2. If $P_k \epsilon P$ and $\forall P_i \epsilon P, P_k \geq P_i$ then $P_k \epsilon P'$.

Step 1 inbeds ESRS into the GA, where ESRS is algorithm B as described by (Luenberger 1984) and stated above. Step 2 insures $C(x) = \{y : Z(y) \leq Z(x)\}$ where C is the GA algorithm. Therefore, CB, the modified GA, converges in probability to the cost minimum.

As one can see, these conditions do not bind the algorithm severely. The "immigration" rate (immigrations/population), 1/M, is related to the "mutation" rate (mutations/bit) as follows:

$$1/M = (\text{mutations/bit}) * (\text{members/population})$$

In fact, the "immigration" of new members may be probabilistic, with probability 1/M.

6.2 Simulated Annealing

One random search technique commonly used to find the global minimum cost in a Boltzmann Machine is Simulated Annealing. This technique simulates the annealing process of metal by probabilistically allowing uphill steps in a state-dependent cost function while finding the global cost minimum, or ground state. The algorithm allows control of the search randomness by a user specified parameter, T. In true metal annealing, this cost function is the Energy of the system, E, and T is the annealing temperature (Kirkpatrick et al. 1983). This method can easily be adapted for finding the minimum entropy of the Organization Level of an intelligent machine.

Given a small random change in the system state $X_i = \{x_1, x_2, \ldots, x_n\}$ to X'_i and the resulting entropy change, ΔH, if $\Delta H \leq 0$, the change is accepted. If $\Delta H > 0$, the probability that the new state is accepted is:

$$p(X_{i+1} = X'_i) = e^{-\Delta H / K_B T} \tag{16}$$

where K_B is the Boltzmann Constant and T is a user set parameter. By reducing T along a schedule, called the annealing schedule, the system should settle into a near-ground state as T approaches 0.

Another method for simulated annealing is discussed in (Hinton, Sejnowski 1986). Using this method, if the entropy change between X_i and X_i' is ΔH, then regardless of the previous state, accept state X_i' with probability:

$$p(X_{i+1} = X_i') = \frac{1}{1 + e^{-\Delta H/T}} \tag{17}$$

Since an intelligent machine consists of a set of binary states, it should be noted that in both of the above methods, X_i' is Hamming distance 1 from X_i (Kam et al. 1985).

The process of simulated annealing escapes local minima through its probabilistic random search, and probabilistically converges to the global cost minimum under certain conditions (Geman, Geman 1984). The next technique, Expanding Subinterval Random Search, probabilistically guarantees convergence within a δ neighborhood to the global minimum of a specified cost function.

6.3 Expanding Subinterval Random Search

A third technique for finding the global minimum value for a cost function for a dynamic system is Expanding Subinterval Random Search as described in (Saridis 1976). Using entropy as the cost function and given a state X_i, one may define the following random search algorithm for an appropriately selected μ:

$$X_{i+1} = \begin{cases} X_i' & \text{if} \quad H(X_i') - H(X_i) \le 2\mu \\ X_i & \text{if} \quad H(X_i') - H(X_i) > 2\mu \end{cases} \tag{18}$$

where $H(Y)$ is the entropy induced by state $Y = (y_1, y_2, \dots, y_n)$ and X_i' is a randomly selected state vector generated from a prespecified independent and identically distributed density function, defined by (5).

It is shown that:

$$\lim_{n \to \infty} \; Prob \; [H(X_n) - H^*_{\min} < \delta] = 1 \tag{19}$$

where $H^*_{\min}$ is the global minimum entropy of the network. The existence of $H^*_{\min}$ is proven in the cited work.

This method can be used on-line to find the global minimum entropy in the Organisation Level of an intelligent machine.

7. EXPERIMENTAL RESULTS

7.1 Simulation of Search Techniques

A net was created which recognised strings of 15 bit binary numbers. The net was formulated using the standard Energy methods found in (Hinton, Sejnowski 1986). Energy was used instead of Entropy in these simulations for two reasons. First, to compare the results of this simulation to the results of simulations by other researchers, a standard measure had to be used. Second, the method for creating regions of attraction in an Entropy based net is still being investigated.

The net had three Energy minima, corresponding to states (001010100100100, 110110110001101, 001111101100010). The respective Energy for these three states were (0.8, 0.6, 1.0). Each simulation technique attempts to find the global Energy minimum of the net, which was 0.6. The cases presented here show best and worst performance of each technique over 10 trials. Other cases which varied the depth and width of the Energy wells are presented in (Saridis and Moed, 1988). For this experiment, the wells were narrow.

The **modified Genetic Algorithm** was performed as presented in Section 6.1. The population was set at 20 members. Each member was 15 bits long, so the number of bits in each population was 300. The "immigration rate" was set to 0.5 which corresponds to a mutation rate of 0.025.

Simulated Annealing was performed using the acceptance criteria in (17). The system was cooled in accordance with:

$$\frac{T_1(t)}{T_0} = \frac{1}{\log(10 + t)}$$

where $T_1(t)$ = temperature at time t

T_0 = initial temperature.

The net state changed in Hamming distance 1 increments.

Expanding Subinterval Random Search (Saridis 1976) was slightly modified to reinforce the probabilistic selection of node states which reduced the Energy in the net. The probability of a node being active was initially 0.5. When the Energy was reduced during search, the probability of the node being reactivated became

$$P(x_i = 1) = P(x_i = 1) + [1.0 - P(x_i = 1)] * 0.1$$

if the node was active, or

$$P(x_i = 1) = P(x_i = 1) - P(x_i = 1) * 0.1$$

if the node was inactive.

Figures 3–8 present the best and worst performance of each algorithm over 10 trials. Modified GA found the minimum Energy string between the 20th and 180th population. Since there were 20 strings per population, this indicates that between 400 and 3600 points had to be generated. The best performance by Simulated Annealing required over 5500 iterations. The worst performance did not converse in 12000 iterations (the most attempted). As a guideline, the best performance of the random search ESRS was slightly over 2000 iterations. The worst performance did not converge in 12000 iterations. The results of these limited experiments force a closer examination of the Modified Genetic Algorithm as a search technique for minimizing the Energy in a Boltzmann machine.

8. CONCLUSIONS

A mathematical theory for intelligent machines was proposed and traced back to its origins. The methodology was developed to formulate the "intelligent machine", of which an intelligent robot system is a typical example, as a mathematical programming problem as using the aggregated entropy of the system as its performance measure. The levels of the machine structured according to the **Principle of Increasing Precision with Decreasing Intelligence** can adapt performance measures easily expressed as entropies. This work establishes an analytic formulation of the Principle, provides entropy measures for the account of the underlying activities, and integrates it with the main theory of "Intelligent Machines". Optimal solutions of the problem of the "intelligent machine" can be obtained by minimizing the overall entropy of the system.

This formulation was proven to be applicable to the derivation and design of parallel architectures for Machine Intelligence. The Boltzmann machine was analytically derived from the definitions of knowledge flow and Jaynes' principle of maximum entropy. The Modified Genetic Algorithm was presented as a search technique which converged in probability to the minimum of a specified cost function. Three techniques, the Modified Genetic Algorithm, Simulated Annealing, and Expanding Subinterval Random Search were described as methods to find the global minimum Energy of a Boltzmann Machine. Simulations using these search techniques were conducted, and results indicate that the modified Genetic Algorithm may be an efficient method to find the minimum Energy.

REFERENCES

Ackley, D. H., (1987), "A Connectionist Machine for Genetic Hillclimbing," Kluwer Academic Publishers.

American Heritage Dictionary of the English Language, (1969).

Birk, J. R. and Kelley, R. B., (1981), "An Overview of the Basic Research Needed to Advance the State of Knowledge in Robotics," *IEEE Trans. on SMC*, SMC–11, No. 8, pp. 575–579.

Conant, R. C., (1976), "Laws of Information Which Govern Systems," IEEE Trans. on SMC, SMC–6, 4, 240–255, April 1976.

Davis, L., Coomb, S., (1987), "Genetic Algorithms and Communication Link Speed Design: Theoretical Considerations," Proceedings of an International Conference on Genetic Algorithms and their Applications, Cambridge, MA, 252-256.

De Jong, K. A., (1975). "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Doctoral Dissertation, University of Michigan.

Fahlman, S. E., Hinton, G. E., Sejnowski, T. J. (1983), "Massively Parallel Architectures for AI: NETL, THISLE and Boltzmann Machines," Proceedings of National Conference on AI, Menlo Park, CA.

Geman, S. and Geman, D. (1984), "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI–6, pp. 721–471, November 1984.

Goldberg, D. E., (1989), "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley.

Grefenstette, J. J., Gopal, R., Rosmaita, B. J., Van Gucht, D., (1985), "Genetic Algorithms for the Traveling Salesman Problem," Proceedings of an International Conference on Genetic Algorithms and their Applications, Pittsburgh, PA, 160-168.
Hayes-Roth, et al., (1983), *Building Expert Systems*, Addison-Wesley, New York.
Hinton, G. E., Sejnowski, T. J. (1986), "Learning and Relearning in Boltzmann Machines," pp. 282-317, in *Parallel Distributed Processing*, ed, D. E. Rumelhart and J. L. McClellan, MIT Press.
Holland, J. H., (1975), "Adaptation in Natural and Artificial Systems," The University of Michigan Press, Ann Arbor, Michigan.
Hopfield, J.J. (1982), "Neural Networks and Physical Systems with Emergent Collective Computation Abilities," Proc. National Acad. Sci., U.S.A, Vol. 79, pp. 2554-2558, April 1982.
Jaynes, E. T., (1957), "Information Theory and Statistical Mechanics," *Physical Review*, 106, 4.
Kam, M., Gues, A. and Cheng, R. (1987), "On Stable Points and Cycles in Binary Neural Networks," Proceedings of the IEEE Intl. Symp. on Intelligent Control, pp. 321-326, Philadelphia, PA, January.
Kirkpatrick, S., Gelatt, C.D. Jr., and Vecchi, M. P. (1983),"Optimization by Simulated Annealing," *Science*, Vol. 220, Number 4598, May 13, 1983.
Kolmogorov, A. N., (1956), "On Some Asymptotic Characteristics of Completely Bounded Metric Systems," Dokl Akad Nank, SSSR, Vol. 108, No. 3, pp. 385-9.
Luenberger, D. L., (1984), "Linear and Nonlinear Programming," Second Addition, Addison-Wesley.
Maudlin, M. L., (1984), "Maintaining Diversity in Genetic Search," Proceedings of the National Conference on Artificial Intelligence, 247-250.
Meystel, A., (1985), "Intelligent Motion Control in Anthropomorphic Machines," Chapter in *Applied Artificial Intelligence,* S. Andriole Ed. Pentrocellis Books, Princeton, NJ.
Saridis, G. N. (1977), "Expanding Subinterval Random Search for System Identification and Control," IEEE Trans. on Auto. Control, pp. 405-412, June.
Saridis, G. N., (1979), "Toward the Realization of Intelligent Controls," *IEEE Proceedings*, Vol. 67, No. 8.
Saridis, G. N., (1985), "Control Performance as an Entropy," *Control Theory and Advanced Technology*, 1, 2.
Saridis, G. N., (1988), "Entropy Formulation of Optimal and Adaptive Control," IEEE Transactions on AC, Vol. 33, No. 8, pp. 713-721.
Saridis, G. N. and Graham, J. H., (1984), "Linguistic Decision Schemata for Intelligent Robots," *Automatica*, Vol. 20, No. 1, 121-126.
Saridis, G. N., Moed, M. C., (1988), "Analytic Formulation of Intelligent Machines as Neural Nets," Proceedings of the IEEE Conference on Intelligent Control, Washington, DC, August 1988.
Saridis, G. N., Stephanou, H. E., (1977), "A Hierarchical Approach to the Control of a Prosthetic Arm," *IEEE Trans. on SMC,* Vol. SMC-7, No. 6, pp. 407-420.
Saridis, G. N. and Valavanis, K. P., (1988), "Analytical Design of Intelligent Machines," *Automatica the IFAC Journal.*
Suh, J. Y., Van Gucht, D., (1987), "Incorporating Heuristic Information into Genetic Search," Proceedings of an International Conference on Genetic Algorithms and their Applications, Cambridge, MA, 100-107.
Vamos, T., (1986), "Metalanguages - Conceptual Models. Bridge Between Machine and Human Intelligence," Working Paper E/37, Hungarian Academy of Science.
Zames, G., (1979), "On the Metric Complexity of Causal Linear Systems, ϵ-entropy and ϵ-dimension for Continuous Time," *IEEE Trans. Automat. Control,* Vol. AC-124, pp. 222-230, April.

ACKNOWLEDGMENT

This work was supported by NASA Grant NAGW-1333.

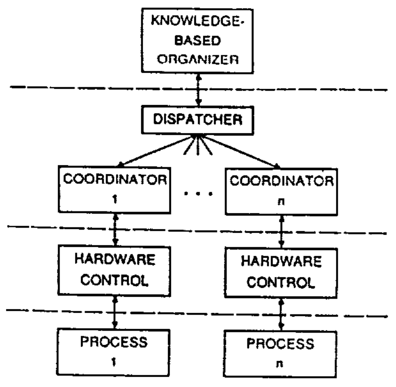


Figure 1. Hierarchical Intelligent Control System

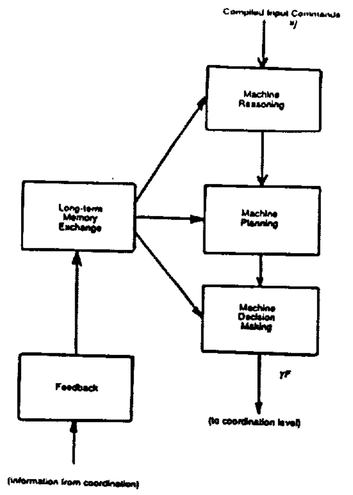


Figure 2. Block Diagram of the Organization Level

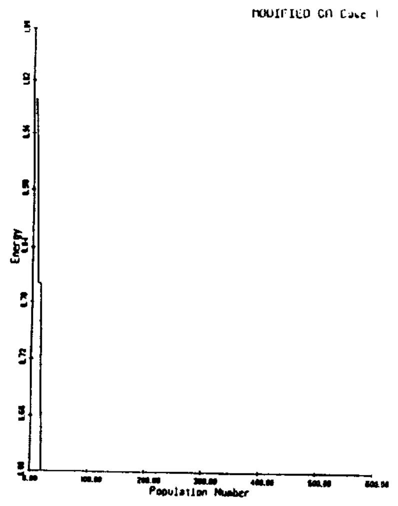


Figure 3. Best Case

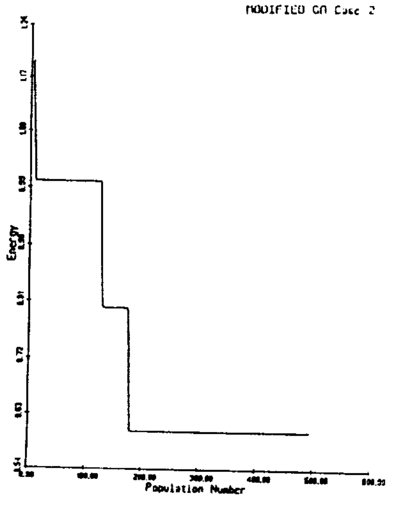


Figure 4. Worst Case

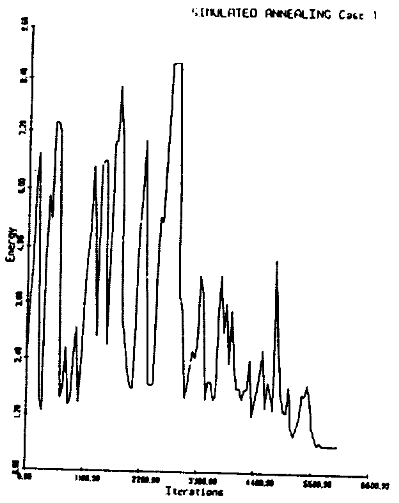


Figure 5. Best Case

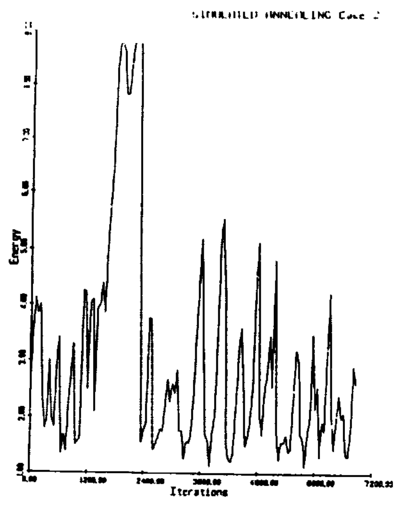


Figure 6. Worst Case

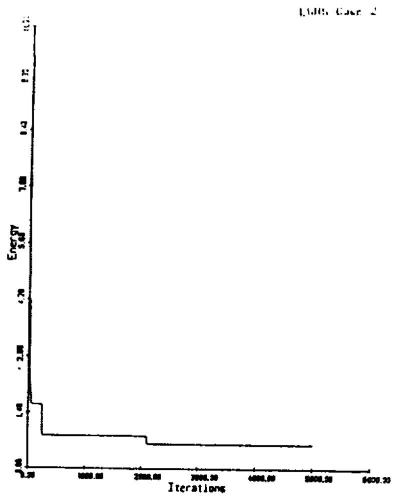


Figure 7. Best Case

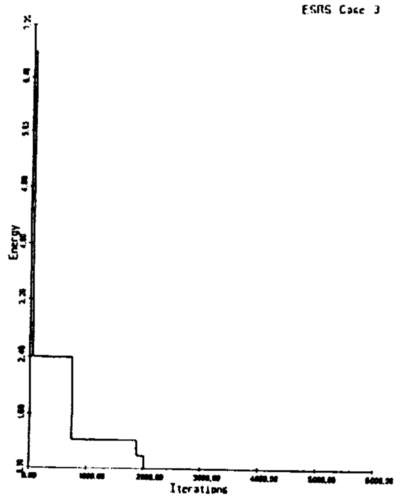


Figure 8. Worst Case

N90-29041

GRASP PLANNING UNDER UNCERTAINTY

A.M.Erkmen and H.E.Stephanou
School of Information Technology and Engineering
George Mason University
Fairfax, Virginia 22030

Abstract

This paper deals with the planning of dexterous grasps for multifingered robot hands operating in uncertain environments. We first describe a sensor-based approach to the planning of a *reach* path prior to grasping. We then develop an on-line, joint space finger path planning algorithm for the *enclose* phase of grasping. The algorithm minimizes the impact *momentum* of the hand. It uses a *Preshape Jacobian* matrix to map task-level hand preshape requirements into kinematic constraints. A master slave scheme avoids inter-finger collisions and reduces the dimensionality of the planning problem.

1 INTRODUCTION

The work described in this paper is motivated by applications that involve dexterous manipulation by autonomous or teleoperated robots in unstructured, uncertain environments. Examples may include equipment maintenance and repair operations in space, under the sea, in a nuclear power plant, or in a chemically contaminated area.

Robot manipulators have traditionally used a gripper (capable of opening and closing motions) attached to their wrist to achieve a rather modest level of mechanical dexterity. This has been adequate for simple manufacturing applications in which the environment may be conveniently structured. The need for a higher level of dexterity, more versatility and more adaptability in end-effectors has become increasingly apparent as the application of automation has grown into areas where the environment is unstructured, and tasks have become more complex. Multifingered hands hold a great deal of promise because of their ability to impart precise localized forces and velocities to objects, and because of their ability to provide stable grasps. Unfortunately, complications arise, as finger coordination, finger trajectory planning, and task planning are not well defined for multifingered hands.

The motion of a robot hand is subdivided into five phases [7]: (i) the reach phase during which the hand moves to the vicinity of some object, (ii) the preshape phase defines an approach volume between the fingers, (iii) the enclose phase until the object reaches the focus of the approach volume, (iv) the grip phase during which fingers apply forces to the object, (iv) the manipulation phase deals with the transfer of the available degrees of freedom to the object.

Our overall objective is to derive intelligent control algorithms for multifingered robot hands in unstructured environments. An outline of our overall approach to grasp planning is outlined in this section. Section two deals with more specific issues related to planning the reach path, while sections three and four focus on the derivation of a minimum momentum approach to finger path planning during the enclose motion of a preshaped hand.

Reach phase
Our main emphasis here is on the development of active sensing strategies. We have developed an *evidential classifier* [12] based on the concept of prototypes for the recognition of graspable objects from incomplete evidence. Prototypical objects and their possible interpretations are stored in a knowledge base during the off-line training stage. The output of the classifier is in the form of belief functions, and must therefore be disambiguated prior to grasping. A *disambiguation* scheme that minimizes the entropy [13] of the interpretation is discussed in section two of this paper.

Sensory data are first gathered off-line, and processed by the evidential classifier to determine a set of candidate reachable objects, or *targets* [15]. Targets are modeled as *attractors* in state space. Similarly, several sets of obstacles are identified, and also represented *repellers*. Each set of targets and repellers is assigned a weight corresponding to their entropy. These sets are used for the local (i.e. around the current position) planning of the reach path. During the execution of the planned motion, additional sensory data are gathered. This is done by using a Newton iteration method (discussed in section 3) that guides the hand closer to targets and obstacles with higher entropy. As additional and/or more refined data are acquired, the classification of targets and obstacles is updated on-line.

Preshape phase
The purpose of this phase is to preshape the hand into a configuration suitable for the anticipated action. Our work here is focused on a new theory of *prehensibility* [10] in which a topological model of prehension is used in conjunction with a knowledge based system to determine hand preshapes from a list of object properties and high-level task specifications. Objects are described geometrically (e.g. cylindrical shape), topologically (e.g. number of vertices, edges, faces), and functionally (e.g. used as a tool). Tasks are described in terms of geometrical, topological, and functional, and behavioral properties.

Enclose phase
The main focus of this paper is on the enclose phase. In sections 4 and 5, we describe a master-slave finger path planning algorithm during the enclose phase. Inputs to the algorithm include the hand preshape, and the desired cartesian position of the master fingertip. The algorithm generates a sequence of knot points (in joint space) that minimized the impact momentum of the master finger, while preserving the hand preshape constraints during the enclose motion. Our approach is based on a Newton iteration applied to the master finger. Using a dyadic expansion of the differential momentum of the master finger, we define a *Preshape Jacobian* that incorporates global (i.e. hand level) preshape constraints into the local (finger level) Newton scheme.
The method is illustrated by computing a Pinch Jacobian and a Hook Jacobian for 2D (planar) motion.

Grip phase
We have studied the performance of a *tentacle-based*, massively redundant manipulator [11] as an alternative to manipulation by an arm/hand combination. The tentacle manipulator is able to grasp by wrapping its links around an object in the same manner used by octopi. This method of grasping is advantageous because the tentacle becomes an all-in-one arm and gripping device capable of a variety of configurations and grasps, while utilizing the mechanics of serial manipulators. We have developed a quantitative method for the evaluation of grasp manipulability and stability accounts for multiple object contacts for each tentacle. Methods for applying both precision and power grasps to three dimensional objects for manipulation using a tentacle manipulator have been derived. These grasps are advantageous because each can be obtained from the other by merely curling or uncurling links from around an object, thereby reducing the number and complexity of grasp configurations.

2 SENSOR BASED REACHING

This section deals with object recognition and path planning during the reach phase of a dexterous grasp.

2.1 Minimum entropy disambiguation

An evidential classifier for object recognition has been described in [12]. We assume that low level sensory data processing has been completed, and that objects have already been detected in the segmented scene. The classifier uses shape primitives (e.g. rectangle, square, triangle) and matches them against an aggregate of prototypical graspable objects that are representative of all the classes (e.g. Pyramid, L-Shape, Handle, Cylinder) of interest. Because the sensory data as well as the aggregate of prototypes are generally incomplete, the classifier output is in the form of a belief function over the object frame of discernment, **FO**.

It is necessary to *disambiguate* the output of the classifier, i.e. to map the belief function in FO into a singleton (single object class), prior to grasping. This is done by using a minimum entropy criterion.

2.1.1 Algorithm

Let the output of the classifier be a belief function with core: Q = $\{q_1, \ldots, q_n\}$, and basic probability assignments: B = $\{b(q_1), \ldots, b(q_n)\}$.
The *class entropy* $h_c(\omega_i)$ of the ith object class is:

$$h_c(\omega_i) = \sum_{p_i \subseteq q_j} \left[-b(q_j)\log(b(q_j)) - \frac{p_i^{\#}}{q_j^{\#}}\log\left(\frac{p_i^{\#}}{q_j^{\#}}\right)\right]$$

The summation is over all focal elements q_j (in FO) containing ω_i. For a given object, the class entropies are computed for each class (singleton of FO). The object is then assigned to the class that yields the lowest entropy.

The belief function reflects two types of distribution among possible classes: (i) a topological distribution formed by the creation of focal elements corresponding to sets of classes, (ii) a probabilistic distribution of belief values assigned to the focal elements. Each class contributes to both types of distributions, and therefore to the generation of entropy. A class contributes to entropy in the topological distribution if it contributes to the confusion in choice. This occurs when a class is embedded in a focal element (i.e. set of classes). A class may also contribute to entropy because of the distribution of belief among focal elements.

2.1.2 Example

Let the belief function M denote the output of the evidential classifier [12]:

M = [{PYRD,LSHP,HNDL,(PYRD,LSHP)},{.09,.17,.4,.18}]

PYRD denotes the class of pyramid shaped objects, LSHP stands for L-shape objects, and HNDL represents the class of handle shaped objects.

The class entropy of PYRD is:

$$h_c(PYRD) = -b(q_1)\log(b(q_1)) - b(q_4)\log(b(q_4))$$

$$-\frac{PYRD^*}{PYRD^*}\log\left(\frac{PYRD^*}{PYRD^*}\right) - \frac{PYRD^*}{PYRD^* + LSHP^*}\log\left(\frac{PYRD^*}{PYRD^* + LSHP^*}\right)$$

$$= .23 + .16 = .39$$

The class entropy of LSHP is:

$$h_c(LSHP) = -b(q_2)\log(b(q_2)) - b(q_4)\log(b(q_4))$$

$$-\frac{LSHP^*}{LSHP^*}\log\left(\frac{LSHP^*}{LSHP^*}\right) - \frac{LSHP^*}{PYRD^* + LSHP^*}\log\left(\frac{LSHP^*}{PYRD^* + LSHP^*}\right)$$

$$= .27 + .13 = .40$$

The class entropy of HNDL is:

$$h_c(HNDL) = -b(q_3)\log(b(q_3)) - \frac{HNDL^*}{HNDL^*}\log\left(\frac{HNDL^*}{HNDL^*}\right) = .16 + 0 = .16$$

The object is classified as a handle since the HNDL class has the lowest entropy.

2.2 Reach path planning

In this section, we assume that a set of targets (Eg. HNDL) and several sets of obstacles (e.g. PYRD and LSHP) have been recognized. Our goal is to design an on-line path planning algorithm for the reach phase. This algorithm can adapt to updates in the classification of targets and obstacles as additional sensory data are gathered.

2.2.1 Target and obstacle representation

Our approach to path planning is based on a local rather than global strategy. To accomplish this goal, we generate two sets of vector polynomial functions: the *target function* $H^A(X)$ and the *obstacle functions* $H^{R_j}(X)$ such that

$$H^A(X_i^a) = 0 \ , \ H^{R_j}\left(X_i^{p_j}\right) = 0$$

where X is the state vector, X_i^a is the location of the ith attractor (target), and $X_i^{p_j}$ is the location of the ith repeller (obstacle) in the jth obstacle set.

2.2.2 Algorithm

Our approach to path planning is based on the Newton iteration:

$$X_{k+1} = X_k - \frac{\delta_k^a}{\gamma^a}[\nabla H^a(X_k)]^{-1} H^a(X_k)$$

$$+ \sum_j \frac{\delta_k^{p_j}}{\gamma^{p_j}}\left[\nabla H^{p_j}(X_k)\right]^{-1} H^{p_j}(X_k)$$

where γ^a is the class entropy for the set of attractors, γ^{p_j} is the class entropy for the jth set of repellers, and δ_k^a , $\delta_k^{p_j}$ are weighting coefficients discussed in [15].

2.2.3 Example

Assume 4 HNDL attractors at

$X_1^a = [1\ 2]\ X_2^a = [3\ 1]\ X_3^a = [4\ 2]$

3 PYRD repellers at

$X_1^{p_1} = [1.5\ 3]\ X_2^{p_1} = [2.5\ 2]\ X_3^{p_1} = [3.5\ 3.5]$

and 4 LSHP repellers at

$X_1^{p_2} = [1\ 1]\ X_2^{p_2} = [2\ 4]\ X_3^{p_2} = [4\ 3]\ X_4^{p_2} = [5\ 2]$

Fig. 1a shows the *field* created by the targets and obstacles. Fig. 1b shows the trajectories for two sets of target and obstacle class entropies. Repellers are represented by filled-in squares, and attractors by filled-in circles. Initial states are indicated by empty circles. Trajectories labeled T_1, T_2, etc. are generated with $\gamma^a = 1$, $\gamma^{p_j} = 1.25$, while trajectories labeled S_1, S_2, etc. are generated by assigning lower entropy for the attractors, and higher entropy for the repellers, namely: $\gamma^a = .5$, $\gamma^{p_j} = 2$ In this case, the trajectories pass closer to the obstacles.

3 FINGER PATH PLANNING

In this section, we apply a Newton iteration similar to the one described in section two to path planning for a single finger during the enclose phase. The results are then extended in section 4 to the grasping motion of a two-fingered hand, by using a master-slave scheme. Our algorithm is based on a Newton iteration scheme that generates a sequence of knot points (in joint space) through which the master finger must pass. This scheme minimizes the *impact momentum* of the finger, evaluated at the desired fingertip contact location.

3.1 Finger momentum

When a wrench vector is applied to the ith finger, it causes changes in its momentum vector G_i. Let r_i denote the fingertip position vector, v_i denote the fingertip velocity, and Δr_i denote a finite fingertip displacement, all in cartesian coordinates relative to a base (palmar) frame. Similarly, let θ_i and ω_i denote the vectors of joint angles and velocities for the master finger, and $\Delta\theta_i$ denote a finite finger displacement in joint space.

The *momentum* of the ith finger is given by :

$G_i = m_i v_i \times r_i + m_i v_i$

where $m_i v_i \times r_i$ is the angular momentum of the finger, $m_i v_i$ is its linear momentum, and m_i is its mass, assumed to be concentrated at the fingertip.

3.2 Finger path planning

The differential mapping of Δr_i into momentum changes ΔG_i is given by:

$$\Delta G_i = J_{r_i}\Delta r_i = J_{r_i} J_{f_i}\Delta\theta_i = J_{\theta_i}\Delta\theta_i$$

where J_{f_i} is the finger Jacobian, i.e. $\Delta r_i = J_{f_i}\,\Delta\theta_i$

Our approach to finger path planning is based on the minimization of its impact momentum. Let θ_i^* denote the desired fingertip position. To find the roots of the momentum function $G_i(\theta_i - \theta_i^*)$, we generate a sequence of knots points by the Newton iteration :

$$\theta_i^{k+1} = \theta_i^k - J_{\theta_i}^{-1}\, G_i(\theta_i^k - \theta_i^*)$$

3.3 Dyadic expansion

Since the finger momentum is a vector quantity, the iterative procedure used for its minimization requires the expansion of the differential momentum in its ***dyadic*** form (reference), i.e.:

$$\Delta G_i = (\Delta r_i \cdot \nabla) G_i$$

$$= \frac{1}{2}[\nabla \times (G_i \times \Delta r_i) + \nabla(G_i \cdot \Delta r_i) + \Delta r_i(\nabla \cdot G_i) - G_i \times (\nabla \times \Delta r_i) - \Delta r_i \times (\nabla \times G_i)]$$

For the special case of 2D grasping in a plane, the angular momentum does not lie in the plane of motion. The linear and angular components of the momentum are therefore not additive. Instead, we use the planar discrepancy between angular and linear momentum (for unit mass), i.e.

$$G_i = (v_i \times r_i) \times v_i$$

The expansion of the dyadic G_i yields :

$$(\Delta r_i \cdot \nabla) G_i = (\Delta r_i \cdot \nabla)(v_i \times r_i) \times v_i$$

$$= \frac{1}{2}\{\, [(v_i \cdot \Delta r_i) r_i - (r_i \cdot v_i)](\nabla \cdot v_i)$$

$$+ [(v_i \times r_i) \cdot \Delta r_i](\nabla \times v_i)$$

$$+ (r_i \cdot v_i)[v_i \times (\nabla \times \Delta r_i) + \Delta r_i \times (\nabla \times v_i)]$$

$$- (v_i \cdot v_i)[\Delta r_i \times (\nabla \times r_i) + r_i \times (\nabla \times \Delta r_i)]$$

$$+ \nabla[(v_i \cdot v_i)(r_i \cdot \Delta r_i) - (r_i \cdot v_i)(v_i \cdot \Delta r_i)]$$

$$- (v_i \cdot \Delta r_i)(V_i r_i - R_i v_i + 2 v_i) + 2(v_i \cdot v_i)\Delta r_i \,\}$$

where

V_i and R_i are the Jacobian matrices of the functions v_i and r_i respectively.

The first line in the dyadic expansion consists of divergence terms. The next three lines consist of curl terms, while the last two lines only contain terms that are not differential.

We postulate that hand preshapes can be analyzed in terms of two motion characteristics:

(i) the *hand flux*, and
(ii) the *hand curl*

Figure 2 shows a schematic representation of a three fingered hand. The hand encloses a *volume* bounded by the fingers. The hand flux is the sum of the divergence of its N fingers moving over a shrinking preshape volume:

$$(\nabla \cdot v)^H = \sum_{i=1}^{N} (\nabla \cdot v_i)$$

As the fingers close during grasping, the fingertip moves along a path enclosing a surface that shrinks. The curl vectors define the directional curling of the finger with respect to its own base, and are given by:

$$(\nabla \times r)^H = \sum_{i=1}^{N} \nabla \times r_i$$

$$(\nabla \times v)^H = \sum_{i=1}^{N} \nabla \times v_i$$

4 MINIMUM MOMENTUM GRASPING

Different types of grasping motion are normally associated with the different preshapes of the hand. In this section, we derive a *Preshape Jacobian* matrix for the preshapes of a 2-fingered hand moving in the plane. Our goal is to map high-level task specifications into joint angles and velocities.

4.1 The Preshape Jacobian

In this section, we modify the Newton scheme described in section 3.2 to include the global (hand level) preshape constraints embedded in the Preshape Jacobian. The modified iteration,

$$\theta_m^{k+1} = \theta_m^k - J_\pi^{-1} G_m(\theta_m^k - \theta_m^*)$$

also minimizes the momentum of the master finger (i=m), but uses the matrix J_π which we call the *Preshape Jacobian* instead of the matrix J_{θ_i} The Preshape Jacobian is defined by:

$$J_\pi \Delta\theta_m = \Delta G_\pi$$

ΔG_π is the *preshape momentum differential.* It incorporates global preshape constraint information into the path of the master finger.

ΔG_π is obtained from ΔG_i by replacing the flux and curl terms for a single finger by the flux and curl expressions for the whole hand, i.e. replace

$$(\nabla \cdot v_i) \quad \text{by} \quad (\nabla \cdot v)^H = \sum_{i=1}^{N} \sigma_i^{\phi v} (\nabla \cdot v_i)$$

$$(\nabla \times v_i) \quad \text{by} \quad (\nabla \times v)^H = \sum_{i=1}^{N} \sigma^{\kappa v} (\nabla \times v_i)$$

$$(\nabla \times r_i) \quad \text{by} \quad (\nabla \times r)^H = \sum_{i=1}^{N} \sigma_i^{\kappa r} (\nabla \times r_i)$$

where the coefficients:

$-1 \leq \sigma^{\phi v}, \sigma^{\kappa v}, \sigma^{\kappa r} \leq 1$

depend on the specific hand preshape constraint. Two examples are given below.

4.2 The Pinch Jacobian

In the pinch grasp (fig. 3a), both fingers contribute to grasping. We assume the two fingers are preshaped symmetrically, and that they remain symmetric during the reach motion. The following set of constraints is used to model the pinch preshape:

$$(\nabla \times \mathrm{v})^H = 0 \qquad (\nabla \times \mathrm{r})^H = 0 \qquad (\nabla \cdot \mathrm{v})^H = \nabla \cdot v_1 + \nabla \cdot v_2$$

and leads to the *Pinch Jacobian*:

$$\Delta G_\pi = J(\theta, \omega)_{\pi_{pinch}} \Delta \theta_m$$

4.3 The Hook Jacobian

For the hook grasp (fig. 3b), fingertip 2 is coupled with joint j of finger 1. The constraints are:

$$\mathrm{r}_{2_x} = \mathrm{r}_{1_x} \qquad \mathrm{r}_{2_y} = -\mathrm{r}_{1_y} \qquad \mathrm{v}_{2_x} = \mathrm{v}_{1_x} \qquad \mathrm{v}_{2_y} = -\mathrm{v}_{1_y}$$

and lead to:

$$(\nabla \times \mathrm{r})^H = \nabla \times \mathrm{r}_2 - \nabla \times \mathrm{r}_1 \quad (\nabla \times \mathrm{v})^H = \nabla \times \mathrm{v}_2 - \nabla \times \mathrm{v}_1 \quad (\nabla \cdot \mathrm{v})^H = \nabla \cdot \mathrm{v}_2 + \nabla \cdot \mathrm{v}_1$$

The *Hook Jacobian* is determined from:

$$\Delta G_\pi = J(\theta, \omega)_{\pi_{hook}} \Delta \theta_m$$

5 DISCUSSION

The minimum momentum grasp planning described in this paper was motivated by applications such as NASA's EVA Retriever, which is required to grasp loose objects tumbling freely in space. In our algorithm, one finger is designated as the master, and its path is planned so as to minimize the impact momentum. A Preshape Jacobian was derived to map task-dependent preshape constraints into kinematic constraints, and thus provide the necessary coupling with the slave fingers. Planning the paths of the slave fingers follows directly from these constraints. We are currently conducting computer simulations for various 2D grasps. The concept of Julia sets [2] is used to graphically study the convergence of the grasping process in various regions of the state space. We are also in the process of deriving expressions of four (fingertip, lateral pinch, cylindrical, and hook) 3D Preshape Jacobians for the Stanford/JPL hand.

6 REFERENCES

1. M.R.Cutkosky and P.K.Wright, 1986, "Modeling manufacturing grips and correlations with the design of robotic hands," *Proc. IEEE Conf. Robotics and Automation*, pp. 1533-1539
2. R.L.Devaney, 1987, *Chaotic Dynamical Systems*, Addison Wesley
3. A.M.Erkmen and H.E.Stephanou, 1987, "Evidential control of dextrous grasps," *Proc. IEEE Conf. Systems Man and Cybernetics*, pp.583-587
4. P.Hsu, Z.Li, and S.Sastry, "1988, On grasping and coordinated manipulation by a multifingered robot hand," *Proc. IEEE Conf. Robotics and Automation*, pp. 384-389

5. T.Iberall, 1987, "The nature of human prehension: three dextrous hands in one," *Proc. IEEE Conf. Robotics and Automation*, pp. 396-401
6. Z.Li and S.Sastry, 1987, "Task oriented optimal grasping by multifingered Robot Hands," *Proc. IEEE Conf. Robotics and Automation*, pp.389-394
7. D.M.Lyons, 1985, "A simple set of grasps for a dextrous hand," *Proc. IEEE Conf. Robotics and Automation*, pp.588-593
8. D.M.Lyons, 1986, "Tagged potential fields: an approach to specification of complex manipulator configurations," *Proc. IEEE Conf. Robotics and Automation,* pp.1749-1754
9. J.R.Napier, 1956, "The prehensile movement of the human hand," *J. Bone Joint Surgery*, Vol.38B, pp.902-913
10. T.N.Nguyen and H.E.Stephanou, 1989, "A topological model of multifingered prehension," *IEEE Int. Conf. Robotics and Automation*
11. J.S.Pettinato and H.E.Stephanou, 1989, "Manipulability and stability of a tentacle based robot manipulator," *IEEE Int. Conf. Robotics and Automation*
12. H.E.Stephanou and A.M.Erkmen, 1988, "Evidential classification of dexterous grasps for the integration of perception and action," *Journal of Robotic Systems*, Vol.5, No.4, pp.309-336
13. H.E.Stephanou and S.Y.Lu, 1988, "Measuring consensus effectiveness by a generalized entropy criterion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 10, No.4, pp.544-554
14. R.Tomovic, G.A.Bekey and W.J.Karplus, 1987, "A strategy for grasp synthesis with multifingered robot hands," *Proc. IEEE Conf. Robotics and Automation*, pp.83-89
15. F.Yegenoglu, A.M.Erkmen, and H.E.Stephanou, 1988, "On-line path planning under uncertainty," *Proc. IEEE Conf. Decision and Control*

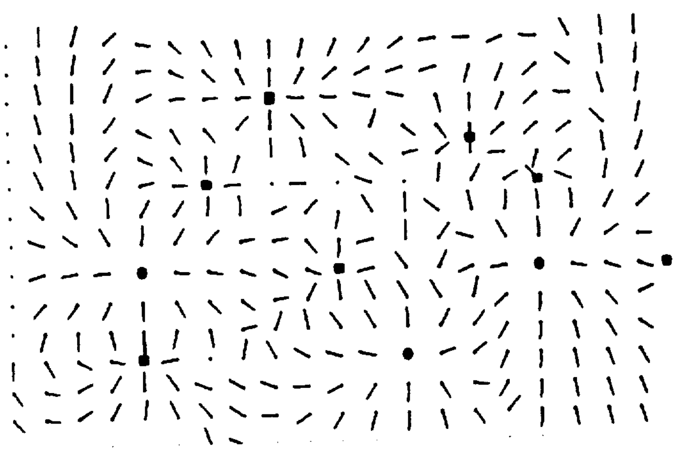


Fig.1a Field

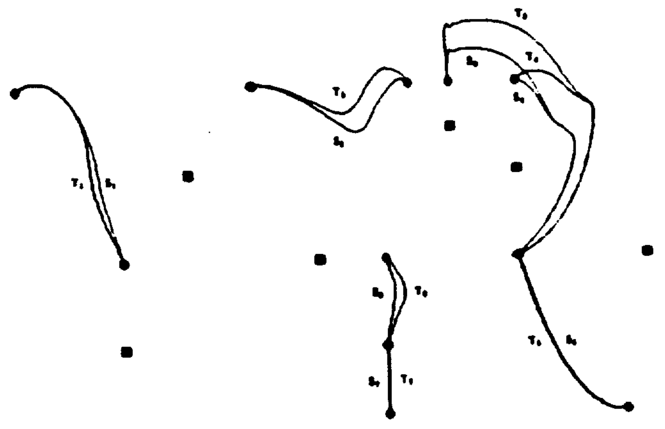


Fig.1b Trajectories

Fig.1 Reach Path Planning

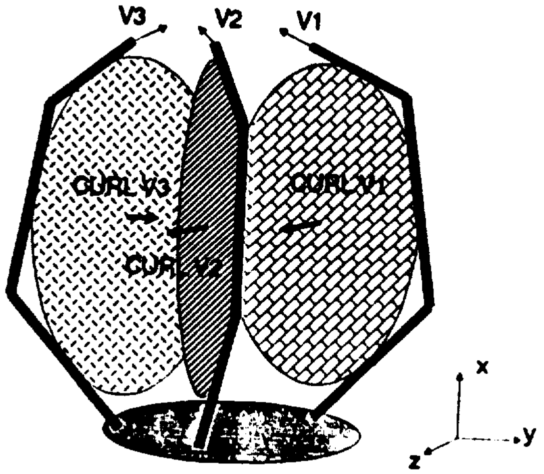


Fig. 2a Link Slices

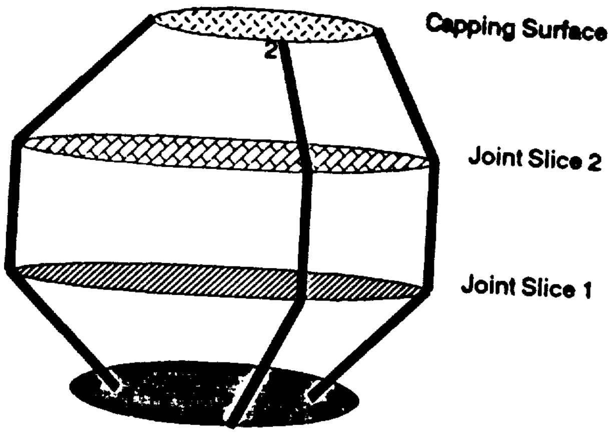


Fig. 2b Joint Slices

Fig. 2 Volumetric Model

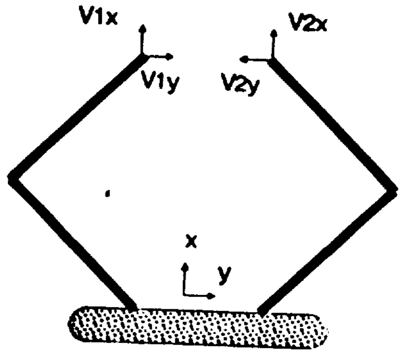


Fig.3a 2D Pinch Grasp

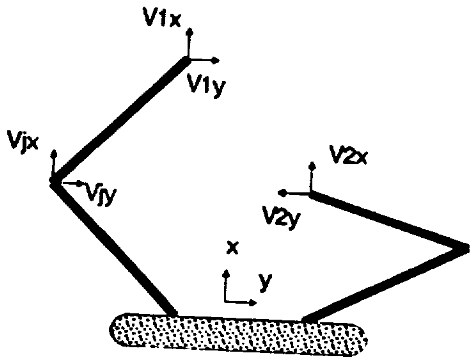


Fig.3b 2D Hook Grasp

Fig. 3 2D Preshapes

N90-29042

Approximation Algorithms for Planning and Control

Mark Boddy and Thomas Dean[1]
Department of Computer Science
Brown University
Box 1910, Providence, RI 02912

Abstract

A control system operating in a complex environment will encounter a variety of different situations, with varying amounts of time available to respond to critical events. Ideally, such a control system will do the best possible with the time available. In other words, its responses should approximate those that would result from having unlimited time for computation, where the degree of the approximation depends on the amount of time it actually has. There exist approximation algorithms for a wide variety of problems. Unfortunately, the solution to any reasonably complex control problem will require solving several computationally intensive problems. Algorithms for successive approximation are a subclass of the class of *anytime algorithms*, algorithms that return answers for any amount of computation time, where the answers improve as more time is allotted. In this paper, we describe an architecture for allocating computation time to a set of anytime algorithms, based on expectations regarding the value of the answers they return. The architecture we describe is quite general, producing optimal schedules for a set of algorithms under widely varying conditions.

1 Introduction

In the best of all worlds, there are infinite computing resources. Unfortunately, this is not the best of all worlds, and, while computing resources are steadily becoming cheaper, there are problems that occur routinely in robotics and process planning that will exhaust any resources that we might plausibly bring to bear. We refer to the class of *NP-hard* problems that, so far, have eluded the best efforts of algorithm designers to provide efficient solutions, and will likely continue eluding them.

Of course, the *NP-hard* problems are not the only obstacle to designing effective control algorithms. There are plenty of problems (*e.g.*, various shortest-path problems) for which there exist polynomial-time solutions that run too slowly on existing machines to support real-time control. In some cases, we can compensate by caching results in tables and computing the answers to problems in real time by table lookup. This approach has its own drawbacks, however, as tables require storage and for many problems the required storage is more than is practical. In addition, as our notion of control expands to encompass more and more complicated sorts of behavior, the number of functions that we would have to tabulate becomes quite large, making the idea impractical.

One conclusion to be drawn from the above is that for some problems we cannot expect the best possible answers; if we want to tackle certain problems, we will have to satisfy ourselves with approximate solutions. Computer science in general and artificial intelligence in particular has been concerned for some time with approximate solutions, and as a consequence many algorithms exist for well-known problems. We can't, however, apply such algorithms directly since these well-known problems are just subproblems of the complex sort of control problems encountered in robotics and process planning. What is needed is a method for integrating solutions to these simpler well-known problems so as to provide reasonable performance for the more complex problems.

In this paper, we present an approach to dealing with problems in real-time planning and control. Our approach involves using a particular sort of algorithm called an *anytime* algorithm. An anytime algorithm can be interrupted at any point during its execution to return an answer whose utility or expected value

[1]This work was supported in part by the National Science Foundation under grant IRI-8612644 and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

is a monotonic increasing function of the time spent computing. The more time available the better the answer returned. A set of such algorithms can be orchestrated to provide solutions to various sorts of control problems that are in some well-defined sense optimal. Our techniques are particularly suited to applications in which the response time for certain critical events is subject to wide variation, and applications that require the solution to several independent subproblems each of which is compute intensive. Such applications are referred to as *time dependent*. We begin our discussion with an introduction to the class of anytime algorithms.

2 Anytime Algorithms

Almost any algorithm can be trivially turned into an anytime algorithm by embedding it in a second algorithm that runs the original algorithm as an inferior process. At any point when the parent process is interrupted and asked for an answer, it checks to see if the inferior process has terminated; if so it returns the answer generated by the inferior process, and otherwise it returns some default answer. The utility of the answers returned by the parent process is a trivial monotonic increasing function of the time spent computing: a step function with a single step. In most cases, however, we can provide a more useful anytime algorithm (*i.e.,* one which produces a succession of increasingly useful results). For instance, many search algorithms employ some sort of a metric for determining if one answer is better than another. At all times, the algorithm keeps track of the best answer computed so far. Such an algorithm could easily be designed to return its current best answer at any point in the computation.

For certain problems in the complexity class *NP*, while there are no known efficient algorithms that compute the exact answers in polynomial time, there exist approximation algorithms that can be shown empirically to provide good answers in a small number of steps. Rather than use complicated methods for choosing the best of some possibly exponential number of alternatives to explore, these algorithms flip coins to determine where to search next. A good example of this sort of algorithm is a probabilistic algorithm for testing primality [Harel, 1987]. This algorithm makes use of the fact that with probability approximately $\frac{1}{2}$, any of the numbers less than the number being tested can serve as a *witness* to its being composite. Finding a witness establishes that the number is not prime. That a number chosen at random is not a witness increases the probability that the number being tested is prime. The time necessary to run this algorithm depends on the probability bound required; the more points tested, the smaller the probability that we will falsely identify a number as a prime. An anytime algorithm for primality testing using this approach would continue choosing numbers at random and testing them as witnesses until it was interrupted (or determined that the number was composite), and then return the probability that the number was in fact a prime.

Another approach to combinatoric problems is to use approximation algorithms which search a smaller space (they are "approximate" because the optimal answer may not be in the reduced solution space). An example of this type of algorithm is the 2-OPT algorithm used for generating approximations to instances of the traveling salesman problem (TSP). 2-OPT begins with a cheaply generated tour that includes each city specified in the TSP instance. It then chooses two arcs in the tour, removes them, and reconnects the disconnected cities to form a new tour of smaller cost. In the standard approach, this cycle is repeated until there is no pair which can be exchanged to improve the tour. It has been shown empirically that running 2-OPT to completion produces tours which average within about 8% of the cost of the optimal tour. There are more complicated edge-exchange algorithms that do better [Lin and Kernighan, 1973]. An anytime algorithm implemented using 2-OPT will exchange pairs of arcs until it is interrupted and asked for an answer, at which point it returns the current tour.

In any interesting control problem, there are lots of different things that must be computed. We may have anytime algorithms for each individual problem, but what we need is some way of coordinating their behavior to produce a composite solution that makes optimal use of the available processor time. In order to engineer such coordination, we need two things: reasonably accurate expectations regarding the utility of the results returned by anytime algorithms as a function of computation time, and some strategy for using these expectations to allocate processor time. The first is relatively easy if we have the luxury of testing our algorithms on real or simulated data; we simply run the anytime algorithms repeatedly and gather statistics on the accuracy of the results obtained as a function of computation time. The second requirement can be more difficult to satisfy, and we devote the following sections to its discussion.

3 Scheduling Anytime Algorithms

The processes that we seek to control generally cannot be halted to wait for the controller to compute a response. However, we often have some idea of how much time is available for computing a re-

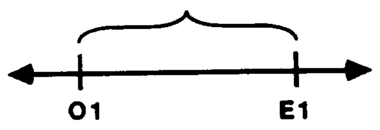


Figure 1: Predicting critical events

sponse. There are a significant number of control problems that can be viewed in terms of reacting to predicted events, employing some model to predict critical events and computing functions to determine how best to respond to those critical events. Figure 1 depicts a time-line showing an observation O1 which can be used to predict the occurrence of a critical event E1. In this simple example, the time between the observation and the predicted occurrence of the event is the time available to compute a response. In tracking a ping-pong ball, for instance, one can predict the time until impact and, hence, the time available to think about how to orient the paddle and take whatever steps are required move it into that orientation. In the traditional approach to control, a discrete control algorithm samples the data at regular intervals, computes a control action, and then executes that action. The control algorithm has a fixed response time. If the sampling interval changes, then the algorithm has to be changed. In many control problems encountered in robotics, sample rates will depend on how quickly a robot can position a sensor, take a reading, and interpret the results. Ideally, the sampling interval will not matter; the controller will do the best it can with the time available.

The robot control problem is complicated by the fact that there may be more than one process to be controlled at the same time. Many problems in control involve coordinating multiple processes. In guiding a mobile robot, the process of avoiding obstacles has to be coordinated with the process of navigating through doorways. Some processes must be monitored and adjusted frequently. In other cases, such as coordinating an assembly process with a parts inventory control process, there is more time between critical events but the parameter adjustments also take more time. Given the problem of coordinating the process of planning a route with the process of driving a car, the two processes have very different utilities; taking a little more time to get there is worth avoiding an accident. Resources such as processor time and access to sensors will need to be allocated to competing controllers. This should happen in a principled way, *i.e.*, so that the resources available are used to produce the best aggregate response

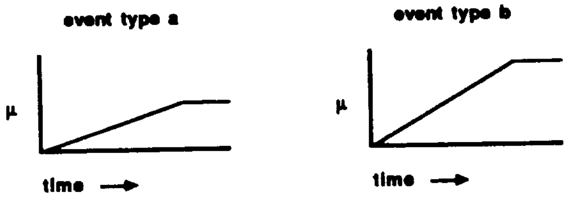


Figure 2: Performance profiles

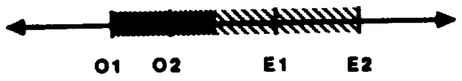


Figure 3: Deliberation scheduling

for all of the processes being controlled.

In [Dean and Boddy, 1988], we define a framework for constructing solutions to time-dependent planning and control problems called ***expectation-driven iterative refinement***. A solution to a time-dependent problem using expectation-driven iterative refinement will consist of a set of anytime algorithms and a ***deliberation-scheduling*** algorithm that allocates computational resources to the set of anytime algorithms based on expectations regarding their performance. An ***optimal*** deliberation schedule for a given situation is a deliberation schedule that maximizes the expected utility of the robot's performance in that situation. An optimal deliberation-scheduling algorithm always generates the optimal schedule for the current situation. An optimal deliberation-scheduling algorithm thus provides the "principled way" of allocating resources that is needed. The basic idea is akin to using a domain-independent planning algorithm coupled with a domain-specific library of plans to generate sequences of actions in novel situations.

The expected utility of the anytime algorithms to be scheduled are represented by ***performance profiles*** that indicate how the expected utility of the answers returned by a given anytime algorithm changes with the amount of time allocated. Figure 2 shows performance profiles for two different algorithms, one for problems of type a, the other for problems of type b. Figure 3 shows two observations and the corresponding predicted events. In this case, all of the time between E1 and E2 can be used in computing a response for E2. If the expected utility of deliberat-

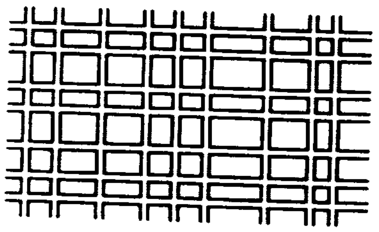


Figure 4: A city map for the robot-courier problem

ing further about E2 is higher than for spending time on E1, then time before E1 may be allocated to E2 as well. If E1 is of type a and E2 is of type b, then deliberation time will be allocated as shown by the shaded areas in Figure 3.

In the next section, we sketch an example of the application of expectation-driven iterative refinement to a robot-planning problem.

4 The Robot Courier

Suppose that we are in charge of designing the control program for a robot courier for a delivery service in a large city. The function of these couriers is to pick up small parcels and deliver them to specified locations. We assume that the city streets are arranged in an irregularly-spaced grid, and that the robot has a map of the city (see Figure 4) to assist in path planning. The robot is also capable of finding its way from one point to another without a planned path by keeping track of the heading of the destination as it performs a form of obstacle avoidance. Path planning helps because a planned path may be more direct. The utility of the robot's performance we define in terms of the time required to complete the entire set of deliveries.

The robot must plan a tour that visits all of the locations on its current list of deliveries. We refer to this as *tour improvement* planning. Once the robot has an ordering for the locations, it may spend time determining how to get from one to another of them. We refer to this as *path* planning. We assume that path planning is accomplished by constructing an ordered set of *target points* between the two locations. Arguably, controlling the robot in navigating between target points will not normally affect the expected utility of tour improvement or path planning. To simplify our discussion, we will concentrate on just these two types of planning and their role in controlling the behavior of the robot. Deliberation scheduling for the robot courier then consists of allocating time to algorithms for tour improvement and path planning based on the expected improvement in the robot's performance.

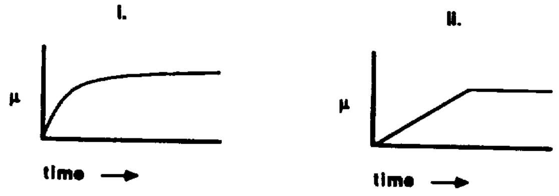


Figure 5: Performance profiles for the robot courier

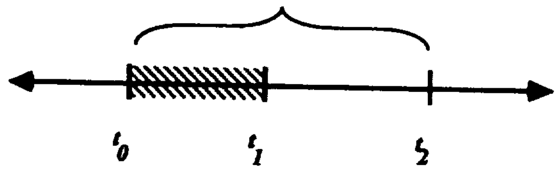


Figure 6: Path planning for a single path

In order to use expectation-driven iterative refinement, it is necessary that we have some expectations regarding the performance of our control algorithms. In the case of the robot-courier problem these expectations can be obtained by performing trial runs to gather the statistics necessary to construct performance profiles for the anytime algorithms for tour improvement and path planning. The tour-improvement algorithm we use is an adaptation of 2-OPT, and has a performance profile of the form shown in Figure 5-i. The path-planning algorithm we employ is a heuristic search algorithm of the sort described by Korf [Korf, 1987], and has a performance profile of the form shown in Figure 5-ii.

Consider the problem of scheduling just the path-planning algorithm for a tour whose order is already fixed. Since the utility of the robot's response is maximized by minimizing the time expended in traversing the tour, the deliberation-scheduling algorithm should minimise the sum of planning and travel time required. Figure 6 shows a tour of two points (*i.e.*, one path to plan for). The robot plans from t_0 to t_1, and then spends from t_1 to t_2 traversing the path. The expected value of the distance from t_1 to t_2 will depend on how long the robot plans (*i.e.*, $t_1 - t_0$). The distance from t_0 to t_2 is the quantity to be minimized in order to produce an optimal deliberation schedule. The problem is slightly more complicated for a tour of n points. Figure 7 depicts the problem of deliberation scheduling for several points. There

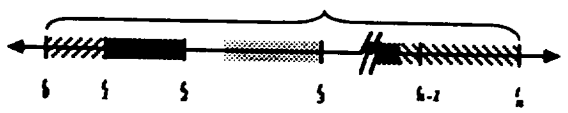


Figure 7: Path planning for several paths

are gaps where no planning is done, because all of the paths left to traverse have already been allocated the maximum useful deliberation time. The quantity to be minimized in this example is $\sum_{i=1}^{n}(t_i - t_{i-1})$. For the robot courier, this problem can be solved analytically; an optimal deliberation-scheduling algorithm appears in [Boddy and Dean, 1989].

Adding tour improvement complicates the problem. Since the path-planning algorithm requires a particular ordering of the points on the tour, the tour-improvement algorithm must be run first. Since the expected savings in time from path planning depends on the distance between locations, the expected utility of scheduling path planning depends on the expected length of the improved tour. In this case, the results of the two algorithms combine by composition: the expected utility of the final result involves the sum of the time spent on tour improvement and the time required to plan for and traverse the improved tour, which is a function of the time spent on tour improvement. It will probably help to go through this in a little more detail.

Figure 8 show a series of five snapshots illustrating the robot in various states of planning and deliberation scheduling. In each of the five snapshots, a "*" indicates the time at which the snapshot is taken, t_0 to t_1 is the time spent path planning before starting to travel to the first location in the current tour, and t_k to t_{k+1} (for $1 \leq k \leq n-1$) is the time spent traveling from the k-th to the $k+1$-st location. Figure 8-i depicts the situation in which the robot has some randomly-generated initial tour and λ_i is the expected time to traverse that tour. At this point the robot has to determine how to allocate time to tour improvement and path planning. The deliberation scheduling required to make this determination can be done very quickly using an algorithm discussed in [Boddy and Dean, 1989]. Here we assume that the time required for this type of deliberation scheduling is ϵ. The current framework for expectation-driven refinement requires that the time required for deliberation scheduling be negligible. In practice, the deliberation-scheduling algorithms we have implemented have been fast enough that this is a reasonable assumption.

Figure 8-ii shows the robot's expectations after the first bit of deliberation scheduling. The interval labeled δ is the amount of time allocated to tour improvement based on expectations concerning both the tour-improvement algorithm and the path-planning algorithm. Expectations regarding the path planner's performance are based on a tour in which the distance between any two adjacent locations is the same. The expected time spent in path planning and path traversal look something like λ_{ii}. Figure 8-iii shows the robot's expectations after actually performing tour improvement. At this point, the robot knows the exact order of the improved tour, and is no longer assuming that the distances are all the same. The interval labeled λ_{iii} is meant to indicate the expected time needed to traverse the tour with no path planning (t_0 is identical to t_1).

Now the robot must determine how to allocate time to planning each individual leg of the improved tour. This is deliberation scheduling of the sort depicted in Figure 7, in which the robot decides how long to apply the path planning algorithm to planning the route between each pair of adjacent locations in the tour. Figure 8-iv shows the resulting deliberation schedule after spending ϵ on this type of deliberation scheduling. The interval labeled λ_{iv} indicates the expected time for carrying out both path planning and path traversal. Finally, Figure 8-v shows the actual schedule and elapsed time λ_v resulting when the robot traverses the tour. Of course, the actual tour may take more or less time than the robot's initial expectations.

The robot-courier example illustrates both kinds of deliberation-scheduling interactions discussed earlier. Solving the problem as a whole requires solving two subproblems that compete for resources: tour improvement and path planning. Path planning for a tour requires dealing with multiple processes: planning the individual routes for each pair of adjacent locations in the tour.

5 Conclusion

The control of complex processes demands that we coordinate our computational and control processes to keep up with the processes that we seek to con-

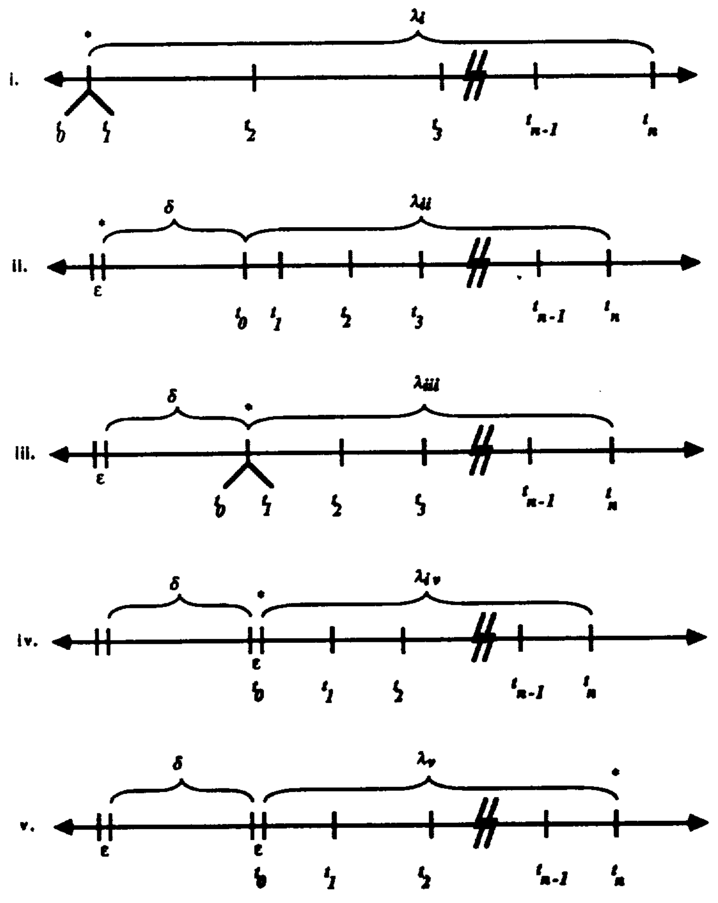


Figure 8: Combining tour improvement and path planning

trol. The traditional approach has been to try to make our computational processes so fast that we can keep pace with any process we are interested in controlling. However, as we tackle more and more complicated control problems, computational complexity limits our ability to reduce computing time. One way to deal with complexity is to use approximation schemes, sacrificing accuracy for speed. In situations in which the control processes provide varying amounts of time to respond, sticking to an approximation scheme with a fixed run time can result in a severe loss in performance. In this paper, we suggest a disciplined approach to using approximation algorithms to cope with processes whose critical or time-dependent events can be predicted with reasonable accuracy. Our approach enables us to allocate processor time to a set of approximation algorithms in order to optimize the performance of a complex control system. The framework of expectation-driven refinement described in this paper provides the basis for solving a wide variety of problems in control and process planning.

References

[Boddy and Dean, 1989] Mark Boddy and Thomas Dean. Solving time-dependent planning problems, 1989.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings AAAI-88*, pages 49–54. AAAI, 1988.

[Harel, 1987] David Harel. *ALGORITHMICS: The Spirit of Computing*. Addison-Wesley, 1987.

[Korf, 1987] Richard E. Korf. Real-time path planning. In *Proceedings DARPA Knowledge-Based Planning Workshop*, 1987.

[Lin and Kernighan, 1973] S. Lin and B. W. Kernighan. An effective heuristic for the travelling salesman problem. *Operations Research*, 21:498–516, 1973.

N90-29043

MULTIRESOLUTIONAL MODELS OF UNCERTAINTY GENERATION AND REDUCTION

A. Meystel
Department of Electrical and Computer Engineering
Drexel University
Philadelphia, PA 19104

Abstract
Kolmogorov's axiomatic principles of the probability theory, are reconsidered in this paper in the scope of their applicability to the processes of knowledge acquisition and interpretation.The model of uncertainty generation is modified in order to reflect the reality of engineering problems, particularly in the area of intelligent control.This model implies algorithms of learning which are organized in three groups which reflect the degree of conceptualization of the knowledge the system is dealing with.It is essential that these algorithms are motivated by and consistent with the multiresolutional model of knowledge representation which is reflected in the structure of models and the algorithms of learning.

Key Words: Abstraction, Generalization, Error, Interpretation, Knowledge, Model, Multiresolutional, Redundancy, Representation, Tier, Uncertainty.

1. Introduction

Multiresolutional system of Knowledge Representation (MKR) and processing (MRKP) is based upon five postulates formulated in [1]: (P1) - descend from the verbal descriptions, (P2) - existence of the external global thesaurus with interpretations, (P3) - dependence on the context, (P4) - metrizability, (P5) - holism[1]. All of these postulates establish representation as a body which must be uncertain. Indeed, the set of verbal descriptions which is the source of representation cannot be complete, and all of these descriptions cannot be adequate (uncertainty of incompleteness and of inadequacy), interpretations from the global thesaurus can be utilized only if they are curtailed (uncertainty of abridgment), context allows for subjective processes encoding and decoding (uncertainty of subjectivity), metrizability is possible only within a certain scope of consideration

[1] A single tesselatum cannot be used for representation, only a complete set of all tesselata can represent the system, the sets of mechanisms of generalization operating among the tessellata is also a part of representation; thus redundancy of representation cannot (and should not) be avoided.

(uncertainty of scoping), finally, the mechanisms of generalization and instantiation carry with themselves all four sources of uncertainty mentioned above, and yet we have to use them (uncertainty of inference [2]).

Uncertainty calls for evaluation which is required for decision making. Indeed, after the alternatives of the future decision are constructed (whether in the problems of design, or in the planning/control problems) these alternatives are to be compared. Consistent comparison can be done only if the judgment is developed about the uncertainty of the evaluation of our alternatives. Each alternative together with the evaluation of its merits and shortcomings has a definite probability of occurrence. The set of alternatives is obtained presumably by combinatorial methods discussed in ATG area [3,4]. The combinations will be compared based on a set {merit, shortcomings, probability]. Thus, the body of probability theory should be evaluated in order to answer the question: can we use its recommendations in the process of uncertainty evaluation?

This makes the 6 famous Kolmogorov's axioms [2] a mechanism that can be used for making our judgment on the utilization of the theory of probability per ce. His axioms are stated for the set E of elementary events which are called elementary events ($E=\{\psi, \eta, \xi,...,\}$, F is the set of subsets of E, and elements of F are called random events. These are the axioms formulated by Kolmogorov for the system consisting of E and F.

Axiom 1. F is a field of sets [3].

Axiom 2. F contains the set E.

Axiom 3. To each set $A \supset \mathsf{F}$ is assigned a non-negative real number P(A). This number P(A) is called the probability of the event A[4].

Axiom 4. P(E) equals 1.

Axiom 5. If A and B have no element in common then $P(A+B)=P(A)+P(B)$.

The set of couples $\{\mathsf{F}, P(\mathsf{F})\}$ is called a *field of probability* where $P(\mathsf{F})$ are the probability values satisfying Axioms 1-5.

Axiom 6. For a decreasing sequence of events $A_1 \supset A_2 \supset ... \supset A_n \supset ...$ of for which the

[2] It is presumed that inference is built upon parallel or sequential mechanisms of generalization and instantiation, (easy to verify, all known rules of inference and logical resolution are based upon determining properties of belonging to a class, or forming a class).

[3] Field is understood as a system which includes all sums, differences, and products of all elements as well as all subsets of it. So F is understood as a mechanism of generating combinations.

[4] One can also use the words: "possibility", "preferability", and so on. The idea of relative frequency is never raised in the set of axioms. This means that the axioms may fit into the structure of fuzzy set theory, Dempster-Shaffer theory, and so on.

product of all sets $\prod A_n=0$ the following equation holds: $\lim P(A_n)=0$, if $n \to \infty$.

We would like to question the validity of the Axioms of 5 and 6 for the case of MKR. Indeed, the phenomenon of having no element in common is not a simple thing especially taking in account the fact that even within a single tessellatum all objects under consideration can be often considered as built of the same primitives (components). On the other hand, condition $\prod A_n=0$ means that the events (sets) under consideration are incompatible. However, the infinite inclusion $A_1 \supset A_2 \supset \ldots \supset A_n \supset \ldots$ does not require necessarily that $\lim P(A_n)=0$, if $n \to \infty$. Everything depends on interpretation of inclusion. This becomes especially important when the process of consecutive generalization is considered.

As it was mentioned in [1], the core of MRKP operations does not differ from the process of the **automated theory generation (ATG)** [3,4]. Since the mechanisms of generalization are involved, then any **process of representation is based upon theory generation**. Like in ATG, the subsystem of representation is supposed to invent and utilize an algorithm of transforming a tessellatum built at a definite resolution into tessellata of lower resolutions. This can be considered a process of synthesizing a consistent system of tessellata constructed at different resolutions and transformable one into another. This synthesis can be performed in a different way depending on initial problem specifications, and entail different results. So, MKR is a source of uncertainty which cannot be considered a fault or a failure: this is an intrinsic property of the system which should be properly understood rather than to uncompromisingly fought with.

2. General Mechanism of Knowledge Processing (GMKP).

A structure of GMKP is demonstrated in Figure 1. It operates as follows.

1. A subset of an object is considered to be of interest. It is presumed that this sub-object (SO) is a part of an object, which in turn, is a part of a particular Domain, which finally, is a part of the World[5]. Information concerned with SO (ISO) is obtained through the set of available sensors which can include all practical variety of them starting with the transducers for delivering actual physical information transformed into a form convenient for the particular system configuration, and ending with the terminals for computer reading necessary documents.

[5] It is important to accept the existence of the World as a part of the problem even is the problem is specified within an extremely narrow domain with a small subset of an insignificant lonely object. The World affects the problem in a powerful way almost in all known cases: via thesaurus, and the process of interpretation no operation of MKR can be performed without taking in account the links with the World.

The Sensor Information Carrier (SIC) delivers ISO to the system for MRKP in a form that contains information about the *code* carried by this particular SIC, and about the *modality* of this particular sensor. The code contains the information of the label and the value, this information should be decoded, and the process of inference is performed, after which all information is structured and stored which (after this) makes it *knowledge*[6].

The left part of the Figure 1 can operate only if the right part exists. As soon as the modality[7] of sensor information is becoming known, o particular Domain of the World Knowledge is being evoked, and the mechanism of interpretation is being prepared taking in account the context, and listing the available rules that can be utilized by the system for dealing with the decoded information.

After the Storage of Knowledge received the interpreted information, the mechanisms of learning are getting involved. The whole body of the stored knowledge is reconsidered in the view of correctness of the classification results after the new information has arrived. As a result of the learning process, new rules for interpretation can be obtained which in fact can affect the process of interpretation and inference and change the prior (recent) results.

SO generates all sources of uncertainty: error of measurement (E), uncertainty of incompleteness (I), and uncertainty of redundancy (R). New EIR-uncertainties are generated within the code as a result of coding and communication; within the interpretation as a result of the EIR-interpretation properties, and within the storage as a result of EIR-properties of classification and other tools of information organization. On the other hand, all subsystems of the right part of Figure 1 contain the same deficiencies.

All these factors should be taken in account when the degrees of belief are being determined. Usually they are generated within the loop of "learning - interpretation - storage". This is why we are especially concerned with the EIR-properties of the external bodies of knowledge which are used for interpretation. One of these properties is the frequency of updating. The case presented in Section 3 should illustrate how these properties are being generated.

[6] Knowledge is defined as internally structured information considered to be a part of some external organization, and allowing for interpretation in some particular context.

[7] Modality of sensor is understood as a subset of the physical phenomena this sensor can sense and submit to the system (like vision, hearing, touch (i.e. surface properties are being sensed), temperature,and many others).

3. A Case Study "Knowledge of a Particular Actuator".

We will discuss knowledge of a particular type of machine actuator: induction motor (the results may be partially used for the similar analysis of synchronous, and DC brushless motors). The model in a form of system of differential equations is very complicated for all these types of actuators, also it is inconvenient in practice of utilization, and can generate many errors because of errors in input information, and because of many factors one neglects in order to use differential equations for modeling the induction motor).

In Figure 2,a a typical "speed-torque" curve is shown which in decades was used to describe the operation of induction motor. Analytically it can be represented in a form

$$T(s) = \frac{2T_{max}}{\frac{s}{s_{max}} + \frac{s_{max}}{s}}$$

where $s=(\omega_0-\omega)/\omega_0$ is a so called "slip" (difference between the speed of the rotating field and the speed of the rotor), ω-speed, T-torque, T_{max}-maximum torque, s_{max}- "slip" corresponding to the maximum torque. This formula (first derived by Kloss) was successfully used in decades. About half a century ago, some reservations were voiced. The Kloss formula was good when its correctness was verified by measurement performed by the Prony method using a very imprecise and often messy method of measurements based upon lever with friction balanced against the torque developed on the shaft. More accurate measurements performed by 2 and 3-machine aggregates led to the experimental data which looked like a curve shown in Figure 2,b.

It was clear that there are many factors creating the phenomenon of these "distortions), and the researchers started working on this "enigmatic"[8] behavior. In forties it became clear that instead of the Kloss formula one should use an expression which looks like

$$T(s) = \sum T_i(s)$$

which contains many Kloss formulas for a variety of the following factors:

1) Fields of "teeth" harmonics (those substantial in magnitude),
2) Imaginary "skin-cylinder" rotor which appears because of the final

[8] Enigmatic-usually means: not coinciding with the model I (he, scientific community) thought of.

surface machining of the rotor,

3) Components of the torque which are generated in the zones of bad insulation among the laminations of the rotor "iron",
4) Harmonics of the stator spatial field due to the nonsymmetrical distribution of the stator winding along the inner surface of the stator window,
5) Saturation of the machine,
6) Nonsymmetry of the stator voltage, and so on

....

Using all of them in all cases would be totally senseless. Using some of them in analytical form would be a matter of choice for a particular designer. Manufacturers started giving instead of a curve T(s) a fuzzy zone around the imaginary average T(s).

In the fifties the topic with T(s) has been exhausted, but one still could not compute a system with induction motor. Now it was clear that something else generates error: probably the dynamic processes. Thus the focus of attention of the researchers in this area shifted toward the time diagrams of speed and torque. Full twelve dynamic equations of the three-phase induction motor (highly nonlinear and coupled) were hard to use, and not always easy to believe. In Figure 3,a two curves demonstrate: 1) T(t) if the Kloss formula is used, 2) T(t) often seen in practice (the phase portrait is shown in Figure 3,b). In the sixties it became clear that due to the exponential components of the current in the winding at the moment of connection to the line, a "swinging" component of the field generated the oscillations with the frequency of the voltage[9]. Oscillations in the end of the process were on the natural frequency of the motor (if modelled as a second order system). It the systems with SCR switches and/or controllers the first component could be controlled and even eliminated. (The whole picture is even more complicated, but the major factors here are presented properly). In Figure 3,c a set of voltage and harmonics is shown dependent on the components of a real speed-torque characteristics.

In Figure 4 the whole multiplicity of factors to be taken in account is collected in a hierarchy. The lower is the level of consideration, the more one can find items (components of the torque) which can be neglected under proper circumstances, which are not as significant as the other components that are retained when the information is generalized for submission to the upper (low resolution) levels. However, it is clear that no judgment of error can be done unless the system is considered as a hierarchy of generalizations and its model can be discussed tessellatum by tessellatum together with their inclusion rules [1].

The following observation can be formulated for each two adjacent resolution levels (a

[9] As a matter of fact, this component was to blame for 70% of all shaft breakages known in industrial practice.

tier) which is being confirmed by the variety of other technological examples. Learning process consists of consecutive refinement of the sets of knowledge containing error [10] (for the low resolution level of all tiers) with transforming it into goal oriented modelled knowledge (for high resolution levels of all tiers). The following questions must be addressed before the qualification of data is done as containing some error and evaluating this error.

1. What should have been considered an error at each stage of our development of the model of T(s) a) a model error, b) an error of measurements ?

2. Which of the models should have been considered a "true model" for dealing with the problem of error evaluation and qualification?

4. Model of Error Generation and Reduction

We will address these questions in a form of recommending a general approach for dealing with processes of error propagation in the system, and for recommending measures of its reduction. Let us first consider the updated Figure 1 which can be corrected based upon the principle of learning formulated above (see Figure 5). We saw that no judgment of error can be made before the system is organized as a hierarchy of generalizations (abstractions, multiresolutional hierarchy, etc). Thus, the hierarchy of resolution conscious information should be sought for from the SO. Then, the Code which arrives should be considered a generalized code which allows for nested hierarchical treatment (recursive interpretation, and/or consecutive refinement). Thus, in Figure 5 a loop is shown from Generalized Interpretation (GI) back to Generalized Code.

Now, the storage is becoming a multiresolutional system, and the whole right side of the structure is being adjusted with methodology of [1]: the source of knowledge is being treated as a multiresolutional structure, rules constitute a hierarchy of classes and a hierarchy of rules within the class, finally, the processes of learning are done consecutively with gradual involvement of each consecutive tessellatum. The system of learning is shown in Figure 6.

Then the following conceptual structure is required to support the MRKP system in the view of dealing with processes of error generation and its reduction (Figure 7). The whole processing is considered as a multiresolutional system of consecutive encoding/decoding procedures. In a number of cases a hierarchy of sensors can be expected that makes the encoding subsystem working with a multiplicity of inputs to all levels.

The process of consecutive refinement is illustrated in the structure of search shown in

[10] Error is understood here as a deviation from experimental data.

Figure 8 for $\sqrt{2}$. In this case the two conditions are to be satisfied

1) $[(\text{upper level})+(.1)(\text{lower level})]^2<2$;

2)
$$(\text{lower level}) < \frac{2 - (\text{upper level})^2}{(.2)\ (\text{upper level})}$$

Each upper level is obtained by averaging the lower level. This generalization rule: averaging is expected to be domination for the highest levels of resolution. It allows for dealing only with interval type of the error with uniform distribution of error within the interval.

The following conclusions can be made:
1. The characteristics of error will depend completely on the procedure of generalization accepted within the particular paradigm. Averaging seems to be the most appropriate procedure of generalization for the higher levels of resolution.
2. Instead of dealing with the second order statistics one can deal with a resolutional hierarchy of first order statistics, each of them for the interval error with a uniform distribution within the interval.
3. If the nature of the error allows for possible models of errors with the infinite interval, it can be substituted by the interval error with the same entropy of the error.
4. Errors are to be dealt with using algorithms of Multiresolutional Nested Consecutive Refinement.
5. Information improvement (learning) procedure can be arranged which allows to predict the level of uncertainty, and to postpone the decision making until the desirable level of uncertainty is achieved.

References

1.A.Meystel,"Techniques and Potential Capabilities of Multiresolutional Information (Knowledge) Processing" (in these Proceedings)

2. A. Kolmogorov, Foundations of the Theory of Probability, (first published in 1933 in Germany, in 1936 in Russia, in 1950 in USA), Chelsea Publishing Company, NY 1956

3. S. Amarel, In Principles of Self-Organization, Trans. of the University of Illinois Symposium on Self-Organization, June 8-9, 1961, Eds. H.Von Foerster, G. Zopf, Jr., Pergamon Press, Oxford, 1962

4. R. Davis, D. Lenat, Knowledge-Based Systems in Artificial Intelligence, McGraw Hill, NY 1982

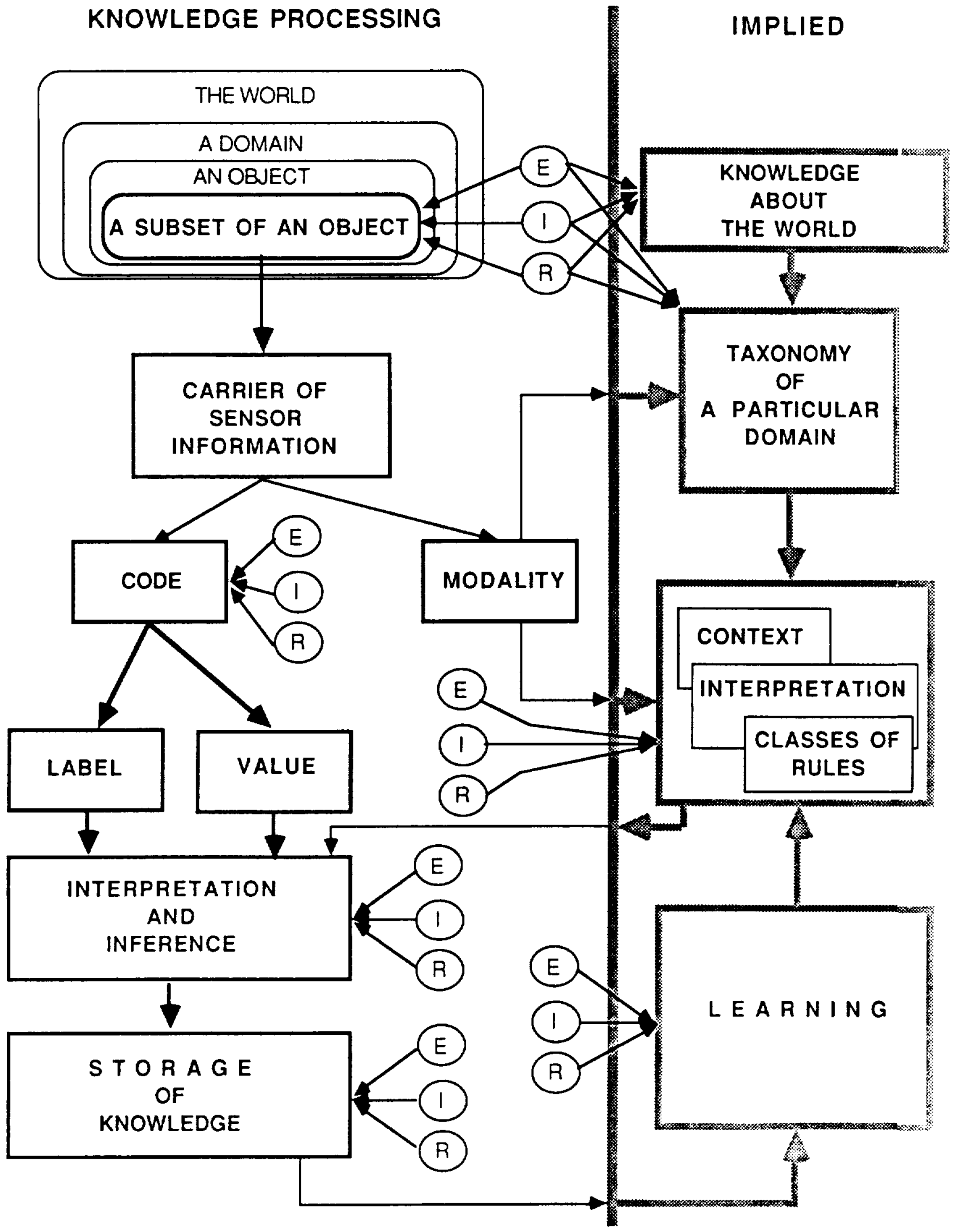
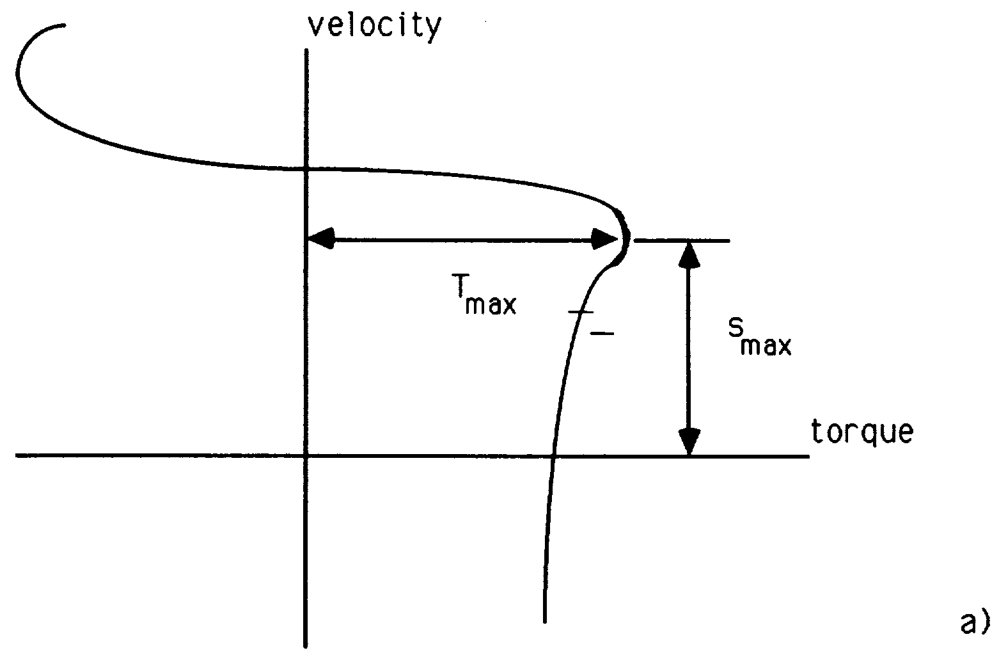


Figure 1.Conceptual Knowledge Processing diagram, and the implied bulk of the required external knowledge support.



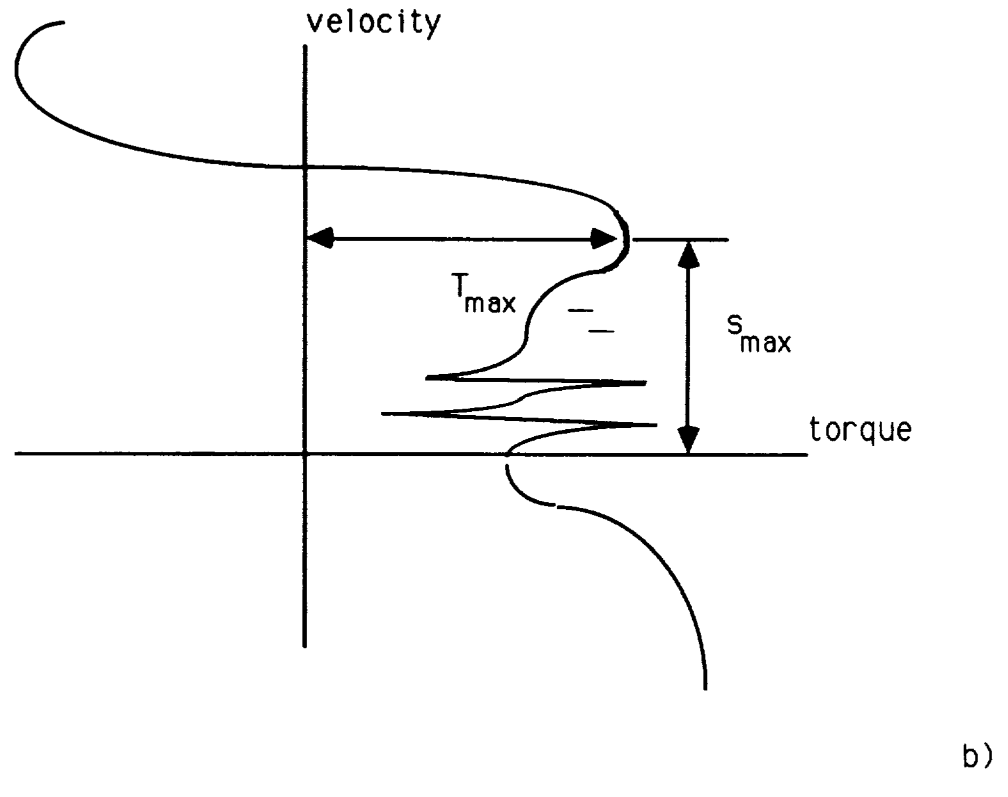


Figure 2. Torque-speed characteristics of an induction motor

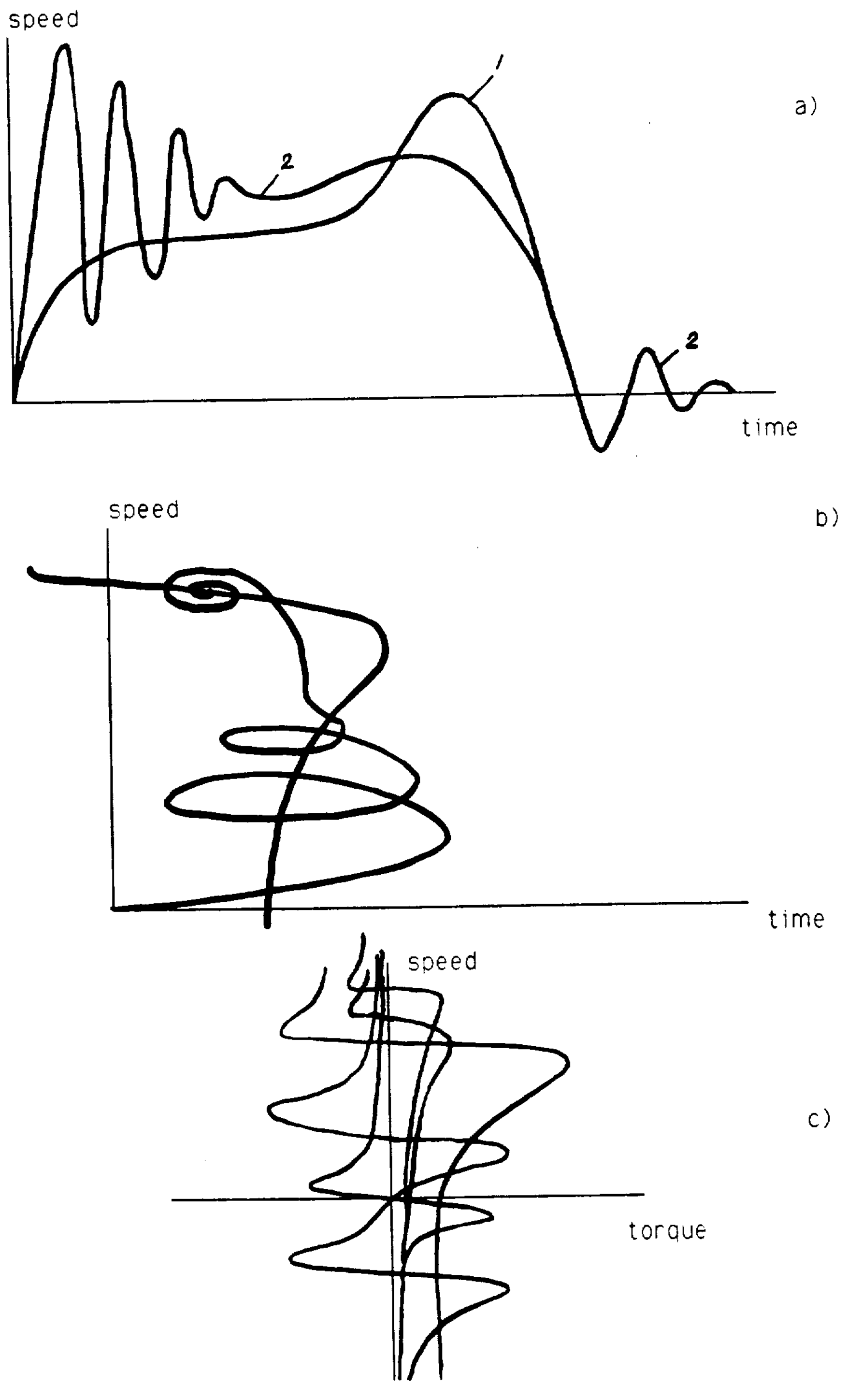


Figure 3. Dynamic Characteristics of the Induction Motor

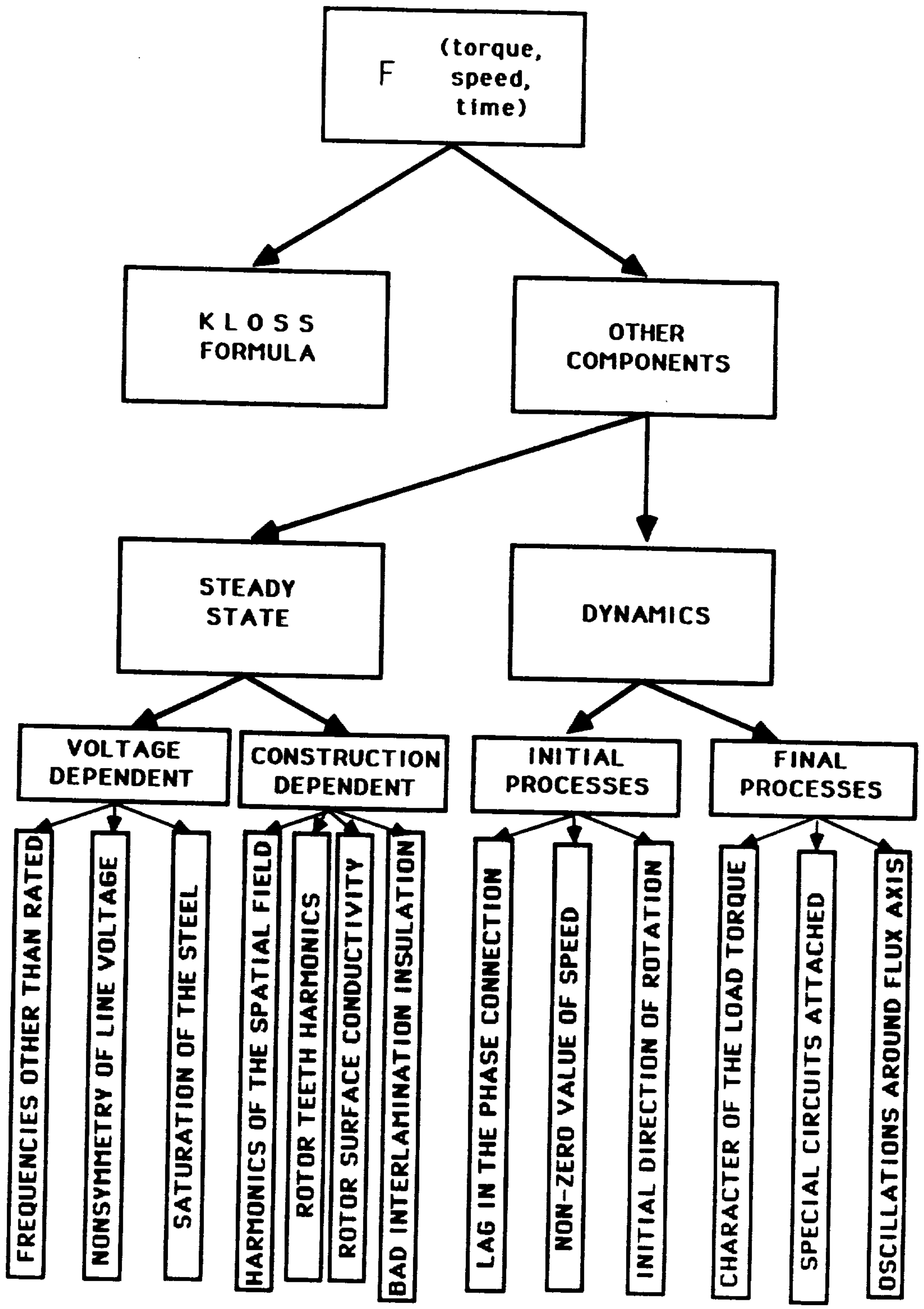

F (torque, speed, time)
KLOSS FORMULA
OTHER COMPONENTS
STEADY STATE
DYNAMICS
VOLTAGE DEPENDENT
CONSTRUCTION DEPENDENT
INITIAL PROCESSES
FINAL PROCESSES
FREQUENCIES OTHER THAN RATED
NONSYMMETRY OF LINE VOLTAGE
SATURATION OF THE STEEL
HARMONICS OF THE SPATIAL FIELD
ROTOR TEETH HARMONICS
ROTOR SURFACE CONDUCTIVITY
BAD INTERLAMINATION INSULATION
LAG IN THE PHASE CONNECTION
NON-ZERO VALUE OF SPEED
INITIAL DIRECTION OF ROTATION
CHARACTER OF THE LOAD TORQUE
SPECIAL CIRCUITS ATTACHED
OSCILLATIONS AROUND FLUX AXIS

Figure 4. Structure of knowledge refinement for the case with induction motor

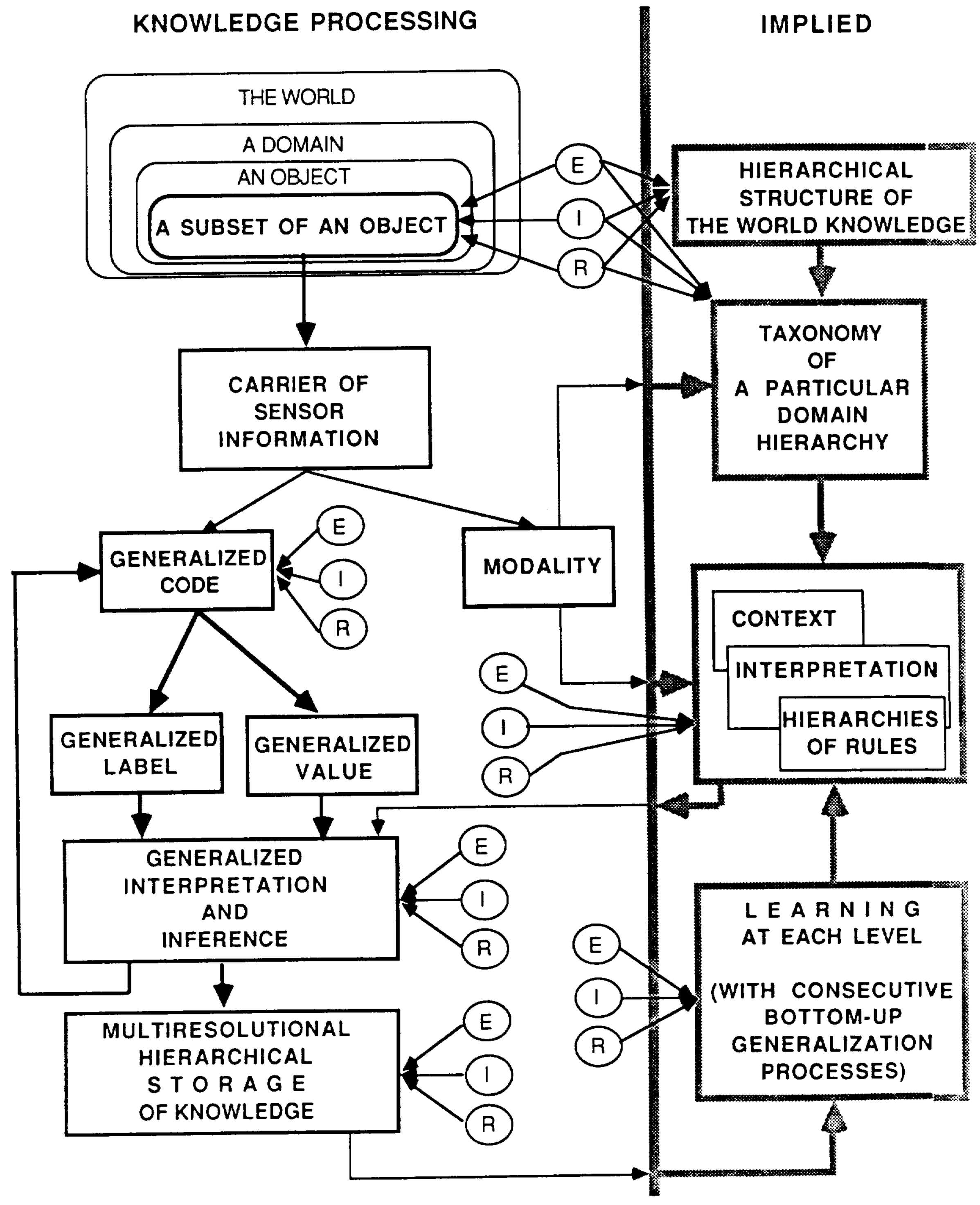

KNOWLEDGE PROCESSING
IMPLIED
THE WORLD
A DOMAIN
AN OBJECT
A SUBSET OF AN OBJECT
E
I
R
HIERARCHICAL STRUCTURE OF THE WORLD KNOWLEDGE
TAXONOMY OF A PARTICULAR DOMAIN HIERARCHY
CARRIER OF SENSOR INFORMATION
GENERALIZED CODE
MODALITY
CONTEXT
INTERPRETATION
HIERARCHIES OF RULES
GENERALIZED LABEL
GENERALIZED VALUE
GENERALIZED INTERPRETATION AND INFERENCE
LEARNING AT EACH LEVEL
(WITH CONSECUTIVE BOTTOM-UP GENERALIZATION PROCESSES)
MULTIRESOLUTIONAL HIERARCHICAL STORAGE OF KNOWLEDGE

Figure 5. Multiresolutional Knowledge Processing (MRKP) diagram, and the implied bulk of the required external knowledge support.

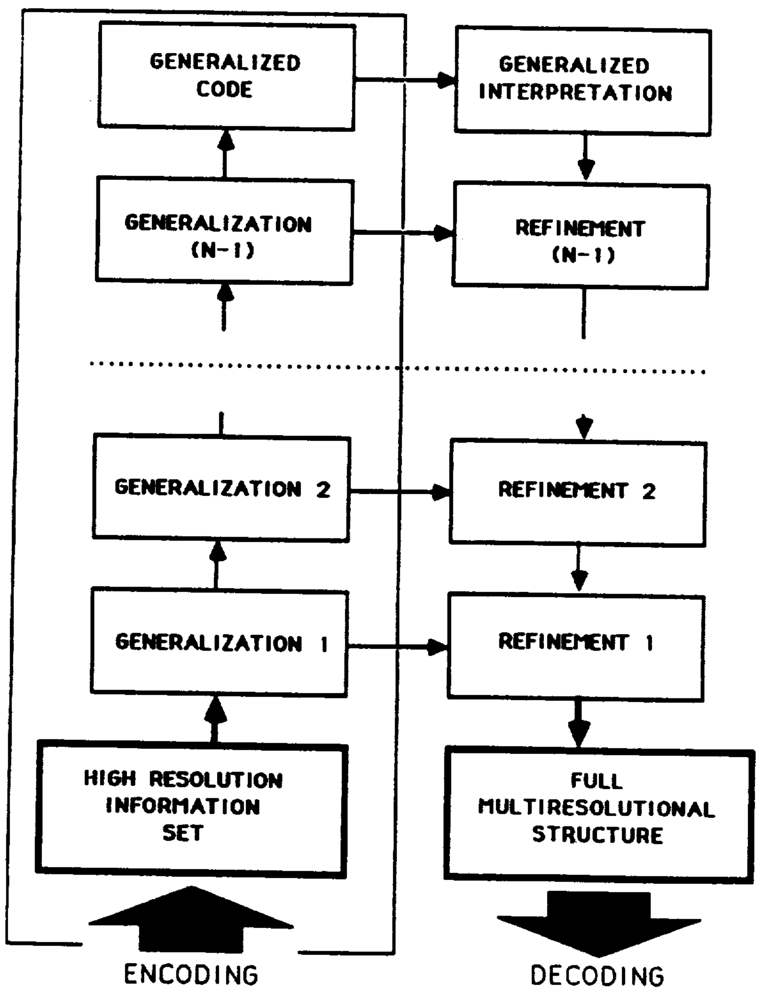


Figure 6. Mutually supportive processes of "encoding-decoding"

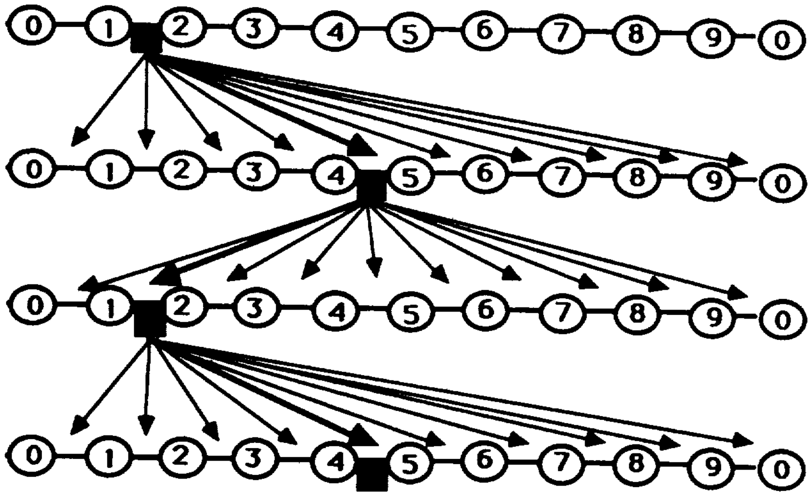


Figure 7. MRKP for solving √2 problem