AMES GRANT

IN-37
34

P33
P_34

NINTH SEMI-ANNUAL REPORT

...SEARCH ON

# CONTROL OF FREE-FLYING SPACE ROBOT MANIPULATOR SYSTEMS

Submitted to

Dr. Henry Lum, Jr., Chief, Information Sciences Division
Ames Research Center, Moffett Field, CA 94035

by

The Stanford University Aerospace Robotics Laboratory
Department of Aeronautics and Astronautics
Stanford University, Stanford, CA 94305

Research Performed Under NASA Contract NCC 2-333
During the period March 1990 through August 1990

Professor Robert H. Cannon Jr.
Principal Investigator

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

This document reports on the work done under NASA Cooperative Agreement NCC 2-333 during the period March 1990 through August 1990. The research was carried out by a team of five Ph.D. candidate students from the Stanford University Aerospace Robotics Laboratory under the direction of Professor Robert H. Cannon, Jr. The goal of this research is to develop and test experimentally new control techniques for self-contained, autonomous free-flying space robots. Free-flying space robots are envisioned as a key element of any successful long term presence in space. These robots must be capable of performing the assembly, maintenance, and inspection, and repair tasks that currently require astronaut extra-vehicular activity (EVA). Use of robots will provide economic savings as well as improved astronaut safety by reducing and in many cases eliminating the need for human EVA.

The focus of our work is to develop and carry out a set of research projects using laboratory models of satellite robots and a flexible manipulator. The second-generation space-robot models use air-cushion-vehicle (ACV) technology to simulate in two dimensions the drag-free, zero-g conditions of space. Using two large granite surface plates (6' by 12' and 9' by 12') which serve as the platforms for these experiments, we are able to reduce gravity-induced accelerations to under $10^{-5}g$, with a corresponding drag-to-weight ratio of about $10^{-4}$—a very good approximation to the actual conditions in space. The flexible manipulator, also using air-cushion technology, is mounted on a third (4' by 8') granite surface plate.

Our current work is divided into five major research projects: Global Navigation and Control of a Free-Floating Robot, Cooperative Manipulation from a Free-Flying Robot, Multiple-Robot Cooperation, Thrusterless Robotic Locomotion, and Dynamic Payload Manipulation. Each of these projects represents an ongoing or completed experimental PhD thesis.

The Global Navigation and Control project demonstrates simultaneous control of the robot manipulators and the robot base position on the free-flying robot model. This will allow manipulation tasks to be accomplished while the robot body is controlled along a trajectory.

The Free-Flying Cooperative Manipulation project has demonstrated the ability to manipulate an object using cooperative arms from a free-flying robot. Mastery of this technology plays a key role in establishing the valuable presence of robots in space.

The Thrusterless Locomotion project demonstrates locomotion of a free-flying robot via the use of its manipulators. This will provide a viable alternative to expendable gas thrusters for vehicle propulsion. This work is carried out on a slightly revised version of the second generation space robot model.

The Multiple-Robot Cooperation project will demonstrate multiple free-floating robots working in teams to carry out tasks too difficult or complex for a single robot to perform. A third space robot model, identical to the robot fabricated for the Thrusterless Locomotion project, recently has become operational– providing the minimal two robots needed for the multple-robot research.

The Dynamic Payload Manipulation project seeks to demonstrate control of non-rigid payloads and explore the payload's effects on the dynamics of a manipulator system. This research addresses the fundamental issues involved with manipulating space-born objects that possess sloshing fuel tanks or flexible appendages such as solar arrays.

The chapters that follow give detailed progress and status reports on a project-by-project basis.

# Chapter 2

# Navigation and Control of Free-Flying Space Robots

Marc Ullman

## 2.1 Introduction

This chapter summarizes the progress to date in our research on global navigation and control of free-flying space robots. This work represents one of the key aspects of our comprehensive approach to developing new technology for space automation. Ultimately, we envision groups of fully-self contained mobile robots making up the core work force in space.

### 2.1.1 Motivation

Although space presents us with an exciting new frontier for science and manufacturing, it has proven to be a costly and dangerous place for people. Space is therefore an ideal environment for sophisticated robots capable of performing tasks that currently require the active participation of astronauts.

While earth based robots have not always proved to be cost effective solutions to manufacturing inefficiencies (due to the abundance of cheap labor), the tremendous cost associated with putting humans in space, especially when EVA is required, makes the economics of robots in space particularly attractive.

As our presence in space expands, we will need robots that are capable of handling a variety of tasks including routine inspection and maintenance as well as unforeseen servicing and repair work. These tasks could be carried out by a fleet of free-flying space robots equipped with a set of dextrous manipulators. Such robots must be able to navigate to a job site, rendezvous with the object in need of service, perform the necessary repair operations, and return to base. Recognizing this need for mobility in space, NASA built the Manned Maneuvering Unit (MMU) to enable astronauts to perform these tasks today.

### 2.1.2 Research Goals

The immediate goals of this project are to:

- demonstrate the ability to simultaneously control robot base position and arm orientation so that a free-flying robot can navigate to a specified location in space while manipulating its arms.
- demonstrate the ability to capture a (possibly moving) free-floating target "on-the-fly" using the manipulator arms while the base is in transit.
- provide a suitable platform for the eventual addition of A.I. based path planning and obstacle avoidance algorithms which will enhance the robustness of task execution.

### 2.1.3 Background

Our laboratory work involves the use of a model satellite robot which operates in two-dimensions using air-cushion technology. We have developed a series of satellite robots which, in two dimensions, experience the drag-free and zero-g characteristics of space. These robots are fully self-contained vehicles with onboard gas supplies, propulsion, electrical power, computers, and vision systems. The latest generation of robots is also equipped with a pair of two-link arms for acquiring and manipulating target objects.

Our work emphasizes the modeling of robot dynamics and the development of new control strategies for dealing with problems of:

- a non-inertially fixed base (i.e. free-floating base)
- redundancy with dissimilar actuators
- combined linear and non-linear actuators
- highly non-linear dynamics
- unstructured environments

It also presents a number of challenging system's design problems resulting from the need to carefully integrate many complex subsystems. These include design and construction of the robot itself which incorporates, electrical, gas, sensor, actuator, computer, communication, and vision systems into an autonomous package measuring under 0.5m in diameter by .75m high[1]. Built on top of this hardware platform is a complex computer system architecture consisting of both onboard and off-board processors that communicate via a fiber optic-based Ethernet link. These computers all run a real-time multitasking operating system and perform a variety of sensor and control tasks including realtime vision processing, dynamics computations, closed-loop digital control, as well as high-level strategic control functions including path planning, sensor fusion, and user interface functions.

[1]Not including the camera boom or the manipulators

## 2.2 Summary of Progress

The following advances have been achieved during the past report period:

- We have demonstrated closed-loop control of a combined spacecraft-manipulator system using full dynamic modelling.
- We have demonstrated the successful tracking and capture of a free-floating, rotating object from a free-floating base.
- We have added an onboard vision system to our robot and integrated it into our control system architecture. A second CPU has been installed onboard the robot to handle local vision processing.
- We have completed our multi-camera-based off-board vision system. It now provides global positioning information for the robot and objects (when they are out of view of the local camera).
- We have developed a modular software architecture that now serves as the framework for our real-time control software.
- Our Point Grabber Vision board has finally been completed and put into production. We have installed one unit onboard the robot to service new onboard camera. We have also installed two units in our off-board card cage to handle the off board cameras.
- A vastly simplified gripper design for the robot end-effectors has been completed and implemented thereby eliminating problems with manufacturing tolerances, bearing alignment, friction, and inherent complexity that plagued the earlier design.

## 2.3 Experimental Hardware

This section reviews the latest refinements we have made to our experimental hardware setup.

### 2.3.1 On-board Camera

We have finally installed an on-board camera for local end-point sensing as was envisioned in our original robot design. The camera is mounted on a truss-like boom that holds it out over the center of the manipulator workspace. Looking downward, the camera is equipped with a 6mm wide angle lens so that it sees an area of approximately .5m $\times$ .7m at table height. The camera is driven by one of our new Point Grabber vision boards which is mounted in the I/O section of our onboard VME bus card cage.

The Point Grabber Vision board provides a list of pixel coordinates and intensity values corresponding to bright spots in the image field. These bright spots are produced by IR (infrared) LEDs (light emitting diodes) that are mounted on the robot base, above the manipulator endpoints, and on the objects we are trying to track and capture. The pixel

data produced by the vision system is processed by a dedicated CPU which we have added for this task. This second onboard processor is a 68020-based single board computer. It relays vision information to the primary control computer using the VME bus backplane operating as a virtual network using the TCP/IP facilities provided by VxWorks, our real-time operating system.

In order to accommodate the additional power requirements imposed by adding the second onboard CPU, we replaced the two 50W 24VDC to 5VDC power converters with functionally equivalent 100W units thereby doubling to 40A our onboard current capacity at 5V. The new power converters (which were not available when we did our original design) are directly plug-compatible with the original units making this a very easy upgrade.

### 2.3.2 Global Vision System

With the addition of the on board camera described above, our off board cameras (two cameras mounted over our table facing downward) are now being used as a global positioning system to track the position and orientation of the robot(s) and floating object(s). We are no longer using this system to track the manipulator endpoints since the onboard camera now performs this function. This revised architecture provides a much more realistic scenario in that local endpoint control will surely be done with a local (i.e. onboard endpoint sensor) while global navigation will most likely be done using a combination of onboard sensors and off board navigation aids.

In order to obtain good vision information over our entire table (3m × 4m) we found it necessary to compensate for the geometric (barrel) lens distortion in software. We are using a third order polynomial in $x$ and $y$ (including all the cross terms) [2] to correct for this effect as well as for any residual errors in camera alignment. A calibration jig consisting of rows of LEDs was constructed along with an automated data reduction procedure that provides the necessary set of calibration coefficients. With this calibration, we are able to obtain absolute accuracy of 2mm to 3mm over our entire table with a resolution of 0.25mm to 0.5mm. Since the onboard vision system uses exactly the same procedure but sees an area only 1/3 as large (on a side) its performance is roughly three times better yet.

### 2.3.3 Revised Gripper Design

The robot end-effectors or grippers have been redesigned and remanufactured. They now use commercial linear ball slides rather than the custom bearings used in the original version. The new design, which is much simpler to manufacture, has resulting in a smaller, lighter, and more reliable mechanism that provides faster and smoother operation.

### 2.3.4 Angular Rate Sensor

We have now mounted our Watson Industries "tuning fork" angular rate sensor onboard the robot. This sensor provides a cleaner (less noisy) angular velocity signal than that

[2] The distortion is actually proportionally to $r^2 = x^2 + y^2$ so the dominant terms for $x_c$ are $r^2 x = x^3 + y^2 x$ and for $y_c$ are $r^2 y = x^2 y + y^3$.

obtained from the global vision system[3].

## 2.4 Capturing a Moving Object

This section covers some of the high level control issues that must be addressed in order to achieve the aforementioned goal of capturing free-floating objects.

In order to successfully capture a free-floating target, a free-flying space robot must simultaneously control both its manipulators and its base body position and orientation. Since the corresponding states are coupled with each other, it is necessary to view the system as a whole rather than as two decoupled problems. Simply generating a realizable intercept trajectory—given the limited actuator authority available, the ever present dynamic constraints imposed by a free-floating robot, and any temporal constraints (e.g., the object might float beyond reach if not caught soon enough)—presents a formidable problem.

### 2.4.1 Intercept Trajectories

In order to rendezvous and capture a moving target, a realizable intercept trajectory must be formulated. The intercept trajectory must always assure that the base position and orientation allow the manipulators sufficient freedom to grapple the target successfully without running into the limits of their workspace.

For low mass objects, the robot base intercept trajectory can simply be a straight line toward the object. The grasped object will have very little effect on the robot motion, and the arms will be able to position the object so as to keep it from contacting the robot base. For a massive object; however, the robot must carefully pull along side the object before attempting to grasp it to avoid the possibility of a collision. Thus, knowing the mass of the target allows us to optimize the necessary base motion. These ideas are summarized in Table 2.1.

| | |
|---|---|
| Tip positions | Must always match with the object. |
| Tip velocities | Must match with the object if it has any appreciable mass. |
| Base position | Must be such as to insure that the object is within reach at intercept time. |
| Base velocity | Must match that of the object if its mass is significant in comparison to the robot's. |

Table 2.1: Intercept Trajectory Requirements

[3]The vision system provides velocity information by passing the position data through a simple second order estimator but the resulting signal is somewhat noisy.

### 2.4.2 Rules for Determining a Trajectory

By choosing the intercept time $t_f$ we can describe the intercept requirements in terms of a set of rules. Knowing the intercept time allows us to predict the terminal object position and orientation (assuming that the object is not experiencing any unknown external forces and/or torques). We can then determine the peak acceleration requirements to see if they exceed our maximum actuator capabilities. If this is the case, the intercept time can be modified as necessary until an achievable path is found.

The rules can be summarized as follows:

**Rule 1** The required manipulator end point positions at intercept time are defined by the target position and orientation.

**Rule 2** The manipulators should be in the center of their work space—this configuration allows the maximum range of motion for final correction of alignment errors.

The first two requirements constrain the desired base position at $t_f$ to lie on a sphere (or circle in a 2-D world) whose radius $R_d$ is defined as the mean distance from the center of the base to the manipulator tips. They also tell us the base orientation once its position is known, thus we have:

**Rule 3a** If the object to be caught has insignificant mass, then we select the desired base position to be the point on the sphere closest to our initial base position (corresponding to the minimum distance path).

**Rule 3b** If the object to be caught is extremely massive, we consider the great circle defined by the intersection of the plane normal to the object's velocity vector at intercept time and the aforementioned sphere. The desired base position is selected to be the point on this great circle that is closest to our initial base position.

A non-linear weighting can be used to span the two extremes described by rules 3a and 3b. Similarly,

**Rule 4** If the object to be caught has no significant mass, we need not place any restrictions on the base velocity at intercept time. However, as the mass of the object increases, so too does the need to match the base velocity with that of the object.

### 2.4.3 Base Trajectories

The typical mode of transportation for the robot base will be to use gas jet thrusters.[4] Since the robot base essentially behaves as a set of $1/s^2$ (double integrator) plants, the most fuel

[4] In certain circumstances motion can be achieved by using the robot manipulator arms to leap from one structure to another.[4]

efficient trajectory is one which is bang-off-bang for each of its degrees of freedom.[5] Clearly, the longer the robot can coast at peak velocity, the less fuel it needs to traverse a given distance in a fixed time. Thus we have selected a generic trajectory consisting of a linear function with parabolic blends described by the following family of equations:

$$a(t) = \begin{cases} a_1, & 0 \le t_1 \\ 0, & t_1 \le t \le t_2 \\ a_2, & t_2 \le t \le t_f \end{cases}$$

$$v(t) = \begin{cases} v_o + a_1 t, & 0 \le t_1 \\ v_o + a_1 t_1, & t_1 \le t \le t_2 \\ v_o + a_1 t_1 + a_2(t - t_2), & t_2 \le t \le t_f \end{cases}$$

$$s(t) = \begin{cases} s_o + v_o t + a_1 t^2/2, & 0 \le t_1 \\ s_o + v_o t - a_1 t_1^2/2 + a_1 t_1 t, & t_1 \le t \le t_2 \\ s_o + v_o t - a_1 t_1^2/2 + a_1 t_1 t & \\ \quad + a_2(t - t_2)^2/2, & t_2 \le t \le t_f \end{cases}$$

The on-off times $t_1$ and $t_2$, along with the signs of $a_1$ and $a_2$, can be found to accommodate a specified set of initial and final conditions $[s_o, v_o]$ and $[s_f, v_f]$ respectively. By assuming that the magnitudes of $a_1$ and $a_2$ are known a priori,[6] the solution for the on-off times is given by a quadratic equation, and the appropriate signs on the $a's$ are found via feasibility tests.

### 2.4.4 Manipulator Trajectories

When executing maneuvers requiring large base motions, one typically need not be concerned with the actual manipulator end point trajectories for the majority of the transit time.[7] However, upon approaching a target it becomes necessary to specify a set of coordinated trajectories for the both the manipulator end effectors and the robot base.

During such transit maneuvers it is often desirable to place the arms in a set position. This may entail tucking the arms in so as to minimize the chances of damaging them by bumping into other objects or holding them in a set position to steady a payload while the robot is in motion. When described in terms of the manipulator joint angles or in terms

[5] For the 2-D case this concept is directly applicable, since the three degrees of freedom $[x, y, \theta]$ and their derivatives can be specified independently. In the full 3-D case, this is no longer true, since the final orientation is a function not only of the change in orientation angles but also of their respective time histories (i.e. it is a nonholonomic system). One partial solution to this problem is to find the *simple* rotation that takes the base from the given initial orientation to the desired final orientation. The final configuration would then be independent of the rate of rotation about this axis, and we could optimize this trajectory. The drawback is that it makes no provisions for non-zero initial and/or final angular velocities.

[6] We assume that $a_1$ and $a_2$ are equal in magnitude; however, the equations are general and do not require this assumption.

[7] An exception might be if the robot were carrying a pointing device (e.g. a camera) whose orientation needed to be controlled.

of a cartesian coordinate system attached to the robot base, these trajectories are simply a set of constants.

Once the robot gets "close" to the object to be grasped we need a trajectory that will allow us to meet the manipulator tip requirements outlined in Table 2.1. A set of quintic polynomials provides a mechanism by which the position, velocity, and acceleration of each manipulator end effector can be varied smoothly from the conditions at the start of the grasp to those at the time of the actual capture. These polynomials correspond to a set of inertially fixed cartesian coordinates and can be updated during the reaching phase to take advantage of improving estimates of the final target position and velocity.

## 2.5 Experimental Work

This section details some of the issues we have encountered recently in our experimental work.

### 2.5.1 Sensor Fusion

With the introduction of the onboard camera, our sensor fusion problem increased in complexity as well as realism. We now have to deal with multiple reference frames and multiple coordinate systems. Our dynamic controllers which utilize computed-torque or inverse dynamics formulations make the assumption that they are operating in an inertial reference frame—that is, they assume that their inputs (e.g. manipulator end point position) are expressed relative to an inertial reference frame. This means that we need to convert our local vision data which is obtained in the frame of the robot base (clearly a non-inertial reference frame since the robot base my be rotating) to the global or inertial reference frame. This transformation requires combining information from the onboard camera with data from the global position system (and possibly the angular rate sensor).

### 2.5.2 High-Level Control

It is clear from a description of the steps necessary in rendezvousing with and capturing a free-floating object that some form of high-level control is necessary. We have developed and implemented a powerful Finite State Machine (FSM) based system which handles both transient as well as persistent stimuli. Persistent stimuli can have discrete values (e.g. "Base = Tracked" or "Object = InView") and maintain their state until changed. Transient stimuli, by comparison, have only the states *active* and *inactive* and are automatically reset (to *inactive*) upon invocation of each transition routine. Transition routines are installed into the state machine dynamically (at run time) and can be bound to boolean expressions composed of stimuli states (e.g. "Base = Tracked AND Object = InView"). Among the benefits of this design are:

- The ability to transparently maintain the state of the system in the various stimuli states.

- The ability to "flow" through a state if the conditions for executing one of its transition routines are met immediately upon entering the new state (without waiting for any additional stimuli events).
- A very natural and easy-to-use synchronization mechanism.
- A simple mechanism for executing an ordered set of instructions each of which depend on an external event even when these events occur in a random order.

We currently have implemented two distinct finite state machines–one for initializing the system and bringing it up to a known state and one for tracking, capturing, and bringing to rest a moving object. As the state machines execute, they reconfigure the control system topology through a variety of modes. These include:

- Joint-based manipulator control with polynomial trajectories
- Cartesian space endpoint control with polynomial trajectories
- Cartesian space endpoint control with object tracking trajectories
- Cartesian space object control with polynomial trajectories

The capture state machine also performs such high-level functions as:

- Deciding when and where to intercept the moving object
- Predicting where the object will be at intercept time and what its velocity and acceleration will be.
- Deciding how to grip the object.
- Deciding if the object has been tracked well enough to grasp it.
- Deciding if the grasp was successful.
- Deciding how to bring the object to rest.

## 2.6 Future Work

We intend to demonstrated the ability to track and capture a free-floating, rotating object from a free-flying base. We have already demonstrated the ability to capture a free-floating object with the base free-floating (i.e. when the object comes to the robot) as well as the ability to capture a stationary object that is initially out of reach (so that the robot must execute a base trajectory in order to capture the object). We are currently working on extending this work by combining these capabilities so that we can capture objects which are moving and initially out of reach. This will require high-level coordinated control of the robot base and the manipulators and will represent the culmination of our current work in combined vehicle/manipulator control.

We also intend to enhance the user interface, making it easier to use our robot system. Ultimately, it is our hope that an untrained user could sit down in front of the system and begin using it in a matter of minutes.

# Chapter 3

# Experiments in Cooperative Manipulation from a Free-Flying Robot

Ross Koningstein

## 3.1 Introduction

Multi-arm cooperative manipulation is one of the basic technologies required for a space-based robot. In this experiment, underlying technologies allowing cooperative manipulation from a free-flying base were developed. Experiments have demonstrated that recursive algorithmic formulations for computed torque are suitable for implementation on real-time systems. This allows a simple, powerful solution of the computed torque [3] control problem for free-flying multi-armed systems, including those possessing closed kinematic chains. Using these algorithms, it is no longer necessary to hand-derive equations of motion for use in control systems - high performance can be achieved using preprogrammed algorithms acting on the robot's state.

### 3.1.1 Research Goals

Specifically, the goals of this project are to:

- Develop a recursive algorithmic solution for computed torque control.
- Study the effects of base accelerations on control.
- Verify the findings experimentally.

## 3.2 Status

This initial research effort into multi-arm cooperative manipulation was completed during this report period. A PhD thesis [5] documenting this work is attached to this report.

## 3.3 Conclusions

In summary, high-performance multi-arm cooperative manipulation from a free-flying robot has been demonstrated. Computed torque control of the robot was facilitated by a newly developed recursive algorithm. Thorough studies provided new insights into the effects of the free-flying base on manipulation. Each new finding has been fully demonstrated experimentally.

# Chapter 4

# Multiple Robot Cooperation

William C. Dickson

## 4.1 Introduction

This chapter summarizes our progress to date in the area of multiple robot cooperation. This work will eventually unite the various lines of research presently being conducted in fixed- and floating-base cooperative manipulation, and in global navigation and control of space robots. Our goal is to demonstrate multiple free-floating robots working in teams to carry out tasks too difficult or complex for a single robot to perform. Achieving this cooperative ability will involve solving specialized problems in dynamics and control, high-level path planning, and communication.

## 4.2 Progress Summary

Activities completed from March 1990 to August 1990 were:

- Third second-generation mobile robot operational and near completion.
- Manipulation object constructed.
- Vision system installed over the 6' by 12' granite table for use in initial multiple-robot experiments.
- Network-based communication module completed.
- Network-based condition module completed.
- Real-time network-based simulation module completed.
- Demonstrated multiple-robot control of manipulation object using vision system and path planning.

## 4.3 Research Goals

Some of the goals of this project are:

- Cooperative manipulation and assembly by multiple robots.
- Fine cooperative manipulation in presence of on-off control.
- Development of control strategies for path following.
- Path generation considering dynamic constraints and obstacle avoidance.

## 4.4 Experimental Hardware

Construction on the third second-generation mobile robot is nearing completion. This robot, identical in design to the second, utilizes a momentum wheel not present on the original second-generation robot. The momentum wheel allows the robots to control their orientation without the use of thrusters. Two newly designed momentum wheels are currently being machined for the identical pair of robots to be used in this research.

The end effectors are pneumatically driven plungers used by the robots to cooperatively manipulate a free-floating object. The object, recently completed, is constructed from two half-square-feet floating pads connected by a three-feet-long metal bar.

## 4.5 Shared Data Module

The Shared Data Module previously implemented in UNIX has been further developed and is now running on our real-time systems. This module facilitates data and command flow over the network between multiple processors. Low-level functions associated with this module allow the calling processor to initiate any of the following data or command block transfers:

- Send a block to a second processor.
- Receive a block from a second processor.
- Order a block transfer between two other processors.

Each data or command block is accessed by name through user-level functions that utilize the low-level functions described above. This extra level of separation removes the user from network and configuration dependencies, resulting in the following user-level data or command block transfers:

- Send a block to a named block (on an unknown processor).
- Receive a block from a named block (on an unknown processor).
- Order a transfer between two named blocks (on any processor(s)).

The Shared Data Module has been successfully tested in two configurations. In the first configuration, composed of two robots and a coordinating off-board processor, both synchronous (each sample loop of the robots) and asynchronous (at robot or coordinator decision points) block transfers were enacted between the robots and the coordinator. In a second and more demanding configuration, a fourth processor was added to facilitate artificial vision in a simulated environment (the simulations, discussed in Section 4.7, were performed by the coordinator processor). In this situation, two extra block transfers occur in each robot's sample loop to exchange simulated sensor and control signals with the coordinator/simulator processor, and additional block transfers from the coordinator/simulator provided information to the artificial vision processor. In both of these configurations, data and command transfers accounted for less than one percent of the sample period– demonstrating that the network communication speed is more than sufficient for our purposes.

## 4.6 Condition Module

One important aspect of cooperation is the synchronization of the cooperating agents. Before the two agents can perform a task together, they must inform each other or a coordinating agent that they are in fact ready to perform. In addition, the operator may desire to have some say in whether or not a task should be performed, even if the cooperating agents are ready. Several conditions determine when the agents should begin their task. In general, a set of conditions that define a particular pattern must be met before a corresponding action is taken.

An example pattern is the case of two robots cooperatively manipulating an object. The pattern, named "OK to Manip Object", is valid when the "Robot1 Status" and "Robot2 Status" conditions have the value "Gripping Object", and that the value of the "Operator Command" condition is "Manipulate Object". Once this pattern is valid, the coordinating agent informs the robots that they are to begin their cooperative manipulation. When one of the conditions is updated and the pattern is invalid, the robots are informed to take the appropriate action.

A network-based Condition Module was developed to facilitate this idea of multi-conditional task execution. The four elements provided by the Condition Module are the Condition Set, the Pattern Set, the Condition Monitor, and the Condition Update facility.

The basic element of the Condition Module is a list of named conditions defined as the Condition Set. Each of the conditions can take on any one of many associated values. An example of a condition is the present activity desired by the operator. A descriptive name for this condition might be "Operator Command", and some of the possible values could be "Grasp Object", "Move Object", and "Refuel". The condition describing the status of a robot could be named "Robot Status", with some possible values being "Inactive", "Tracking Object", and "Reaching for Object". Together, the name and value of a condition define a condition-value pair.

The Condition Module supplies a user-level function for creating entries in the Pattern Set. An entry consists of a pattern name and a list of condition-value pairs. All conditions

listed in the pattern entry must occupy the declared values before the pattern is considered valid. The pattern remains valid until one of the conditions is updated to a different value, at which point the pattern becomes invalid. Several patterns can be valid concurrently, and patterns can be sub- or super-sets of other patterns. The user can install with each pattern lists of functions, or "hooks", to be executed when the pattern becomes valid or invalid. These valid- and invalid-pattern hooks initiate and halt activities related to the defined pattern. When the example pattern "OK to Manip Object" becomes true, one of the valid-pattern hooks would send a message to the robots informing them to begin their cooperative manipulation. Similarly, an invalid-pattern hook informs the robots to halt their cooperative manipulation when the pattern becomes invalid. The user can also define a stimulus list for the valid and invalid periods of each pattern. An entry in a stimulus list consists of a condition-value pair and a corresponding stimulus function. A valid-pattern stimulus function is executed whenever the pattern is valid and the associated condition-value update is given. The analogous rule applies to the invalid-pattern stimuli. The purpose of the stimulus functionality is to facilitate repetitive activities within the domain defined by the particular pattern. In the case of the pattern "OK to Manip Object", a robot could periodically send a stimulus to the coordinating agent indicating the present manipulation status. One value of the condition-value stimulus may serve as a keep-alive signal, while others may indicate completion of the present task or an error condition. If the stimulus was a keep-alive signal, the stimulus function would, for instance, reset a timer. In any case, the stimulus causes the corresponding stimulus function to be executed.

The functions of the Condition Monitor are to maintain the Condition Set and to execute the valid- and invalid-pattern hook and stimulus functions when appropriate. The Condition Update facility provides functions that allow the cooperating agents or other sources (such as a user interface) to submit condition value updates to the Condition Monitor. This update facility manages the network communication necessary in the multi-processor environment.

All of the above examples represent important issues in any multiple-agent system. The Condition Module provides a formal structure for specifying the complex conditional relationships that would be difficult to program in the standard "if-then-else" fashion, while also providing a mechanism for executing the associated tasks.

## 4.7 Simulation Module

Prior to the completion of the third robot, testing of new controllers, communication architectures, and path planners for use in the multiple-robot manipulation system necessitated the use of simulations. The third robot possessed a functioning computer early in its development, but the addition of the robotic arms and most of the circuitry followed much later. Thus, the robot could communicate with other robots and computers, but it had no sensing or manipulating capabilities. These conditions inspired the development of a software module that performs real-time simulations of multiple physical systems.

The simulation paradigm is as follows. Simulation routines provided by the Simulation

Module update the simulated states and sensors of the various physical systems of interest. These simulations can execute on one or several processors. The controllers associated with the physical systems read the simulated sensors over the network using a routine provided by the Simulation Module. These sensor signals are used to compute control forces and torques. Now, instead of sending these control signals to real on-board actuators, the controllers write the controls back over the network to the simulation processor using another Simulation Module routine – where they are used as inputs to the simulated systems. These inputs are supplied by the controllers asynchronously, allowing for realistic sample-rate differences between processors. All data movement associated with the Simulation Module is facilitated through Shared Data Module routines, as discussed in Section 4.5.

In a typical control loop, sensors are read, the control is computed, and last the control is sent to the actuators. The simulation module has replaced the first and third parts of this loop, while the computation of the control signals can still performed by the actual processor. This separation of functionality has been crucial in our development of real-time software in two ways. First, computation of the controls is oblivious to the origin of the sensor signals and to the destiny of the resulting control signals. For this reason, the same real-time control software used with the simulated system is completely compatible with the true physical system. Second, the control software can run on the real robot processors, allowing real-time networking and timing issues in the multiple-robot system to be addressed and debugged before the robots are mechanically functional. These desired modular properties were proven as the switch was made from simulation to real hardware– no software changes were necessary; after setting a software flag, function calls that deal with real sensor and actuator signals were executed. When repairs or modifications disable one or more of the robots, the simulation mode can be re-enabled simply by resetting the appropriate flag– allowing research to continue.

## 4.8 Experimental Results

Initial experiments have successfully demonstrated multiple-robot control of a manipulation object (a floating beam). In these experiments, an off-board vision system tracks the positions and velocities of the beam and robots (as well as the robot manipulator endpoints), and sends this information via the network to each of the robots. The robots regulate the beam at a location specified by the operator. The on-board air thrusters and momentum wheels maintain each robot's position and orientation within manipulation range of the beam.

The basis of a control scheme that facilitates cooperative manipulation by multiple robots should center on the desired motion of the manipulated object (in our case, the beam). The operator's concern is the proper positioning of the object– not the control torques and forces on the robots. Following this philosophy, the motion of the beam used in this research is effected as follows. The operator commands a desired beam position to each of the two robots. Each robot then independently determines (using the same algorithm) the desired acceleration of the beam that will guide the beam to the commanded

position. These desired beam accelerations are then used to determine the accelerations of the grasp points on the beam for the given robot (the present beam has four useable manipulation ports) as well as the required manipulation forces at those grasp points. Each robot has knowledge of how both robots are currently grasping the beam (the grasp configuration), allowing each robot to determine its share of the control effort. Each robot uses the accelerations and the manipulation forces at the grasp points together with station-keeping commands to determine the required joint and base torques and forces, as described in the Eighth semi-annual report [2].

## 4.9 Future Work

The major hardware issues have been resolved. An improved momentum wheel system has been designed and the necessary new parts for the two robots are currently being machined. After the installation of the new momentum wheels, one of the robots will be completed, and the following hardware issues will remain for the other:

- Mounting an angular rate sensor.
- Mounting fiber-optic Ethernet transceiver and cable to replace the standard coaxial cable.

Several software modules also need further testing or development:

- Incorporate and test the Coordinator module– allowing the operator to issue commands to a single agent rather than the individual robots.
- Incorporate and test the Condition module.
- Expand a utility allowing the user on a remote host to execute functions on the real-time systems.

# Chapter 5

# Experiments in Thrusterless Robot Locomotion for Space Applications

Warren J. Jasper

## 5.1 Introduction

Much of the recent work in robotics has assumed *a priori* that the desired task is within the workspace of the robot. This assumption is clearly evident for all fixed-base robots, whether they be bolted to the factory floor or to the cargo bay of the space shuttle. A fixed-base robot is ideally suited to perform repetitive tasks in a highly structured environment.

Unfortunately, this is not the situation in space, where the environment is unstructured and the tasks are varied. For example, building or repairing a space-station requires the ability to perform a myriad of different tasks at different locations. Many missions that the astronauts are called upon to perform require mobility, such as repairing a damaged satellite. Therefore, a space robot must be mobile to perform different tasks at various locations. If space robots are to be mobile, then one must ask the question: "What are good ways for a robot to move from one place to another in space".

The ways a robot can move through space can be divided into two types: Those that use propulsion as the primary means for imparting momentum to the robot, and those that do not. This research addresses the issue of locomotion without propulsion.

### 5.1.1 Research Goals

The goal of this project is to study the dynamics and control issues involved in thrusterless locomotion, and to demonstrate experimentally that a free-flying multi-arm robot can accurately reposition/reorient itself while pushing off from another body (e.g. a space station) rather than using thrusters. The aim is not to construct a space qualified robot, but rather

to demonstrate in two dimensions in the laboratory a design philosophy that will easily extend to three dimensions in space.

More specifically, the goals of this project are to:

- Develop a control paradigm and simple algorithms that aid in thrusterless locomotion.
- Achieve smooth switching between control laws during abrupt changes in kinematic configuration. A challenging aspect of this project is to control a nonlinear plant with changing degrees of freedom.
- Construct an autonomous free-flying multi-arm robot that simulates the zero-g drag-free environment of space.
- Verify the design experimentally.

## 5.2 Status

The initial research into thrusterless locomotion was completed during this report period. A PhD thesis [4] documenting this work is attached to this report.

## 5.3 Conclusions

A new method for controlling the dynamic behavior of a rigid-body system was developed. Called *system momentum control*, this method allows one to specify momentum trajectories instead of Cartesian-space or joint-space trajectories. The method was verified experimentally in demonstrations of thrusterless locomotion.

# Chapter 6

# Experiments in Control of a Flexible Arm Manipulating a Dynamic Payload

Lawrence J. Alder

## 6.1 Introduction

This chapter introduces some of the issues involved in controller design for a flexible robot arm grasping a nonrigid payload. The effect of a nonrigid payload on dynamics of the system are discussed. There is also a description of the experimental hardware and the future direction of the research.

### 6.1.1 Motivation

In space applications, the RMS (remote manipulator system) will be manipulating satellites that may contain fuel or have flexible appendages. The sloshing of the fuel and vibrations of the payload will effect the performance of the RMS. These vibrations can conceivably cause the manipulator to go unstable. Even if the arm is able to position a satellite, in order to maintain position accuracy the arm will need to apply force every time the fuel sloshes back and forth, thus wasting power. If the arm releases a payload that is still vibrating, the payload will move and possibly damage other structures. It is thus desirable to design a controller that can handle such 'dynamic' objects and damp out their internal vibrations.

### 6.1.2 Research Goals

The goals of this project are:

- Demonstrate stable end-point control of a flexible arm manipulating both rigid and nonrigid objects

- Demonstrate the ability of a flexible robot arm to damp internal vibrations in a payload
- Demonstrate the ability to move a nonrigid object from an initial point to a final point with no residual vibrations
- Demonstrate all of the above without knowledge of whether the payload is rigid

### 6.1.3 The Experimental System

In order to study the effects of a dynamic payload without unnecessary complexity, the experiment will be conducted with a one-link planar flexible manipulator. For this initial work the arm will manipulate dynamic objects that are 'simply modelable.' Simply modelable refers to objects such as pendulums that can be described by equations of motion. Future research will be necessary to extend this work to complex objects such as sloshing fluid which are described by the Navier-Stokes equations. It should be noted that some experts feel that a simple pendulum is a reasonable model for slosh dynamics [1].

## 6.2 System Dynamics

It has been shown in previous work at the ARL [8] that endpoint controllers are very sensitive to the mass at the tip of the arm. In fact if the arm controller is designed for a large mass and a lighter one is grasped, the arm will go unstable with conventional endpoint control. The reason that the controllers are so sensitive to the tip mass can be summarized in the frequency domain. The conventional endpoint feedback controller uses LQR[1] designs that in effect notch out the vibration modes of the arm. If the vibration modes of the arm are not what the LQR design algorithm assumed, the controller will perform poorly and possibly be unstable. The point being that it is the vibration modes of the system (arm and payload) that the LQR controllers are sensitive to.

Thus one must ask what effect does a nonrigid payload have on the system vibration modes?

### 6.2.1 The Pseudo-Finite-Elment Model

In order to study the vibration modes of the system one must select a method to model the beam and payload. There are many techniques available to model flexible beams such as the assumed-modes method [7] or finite elment codes. The goal here is to get a first pass analysis of how the dynamics of the system change with the addition of a dynamic payload. With this goal in mind, a pseudo-finite-element (PFEM) model has been developed

The PFEM model entails dividing the beam up into many small rigid beams connected by torsional springs. Figure 6.1 shows a diagram of a beam modeled with three finite elements. Obviously, one uses more than three elements when doing the analysis. With this method all of the torsional springs have the same spring constant as done in [6]. The

[1] Linear quadratic regulator

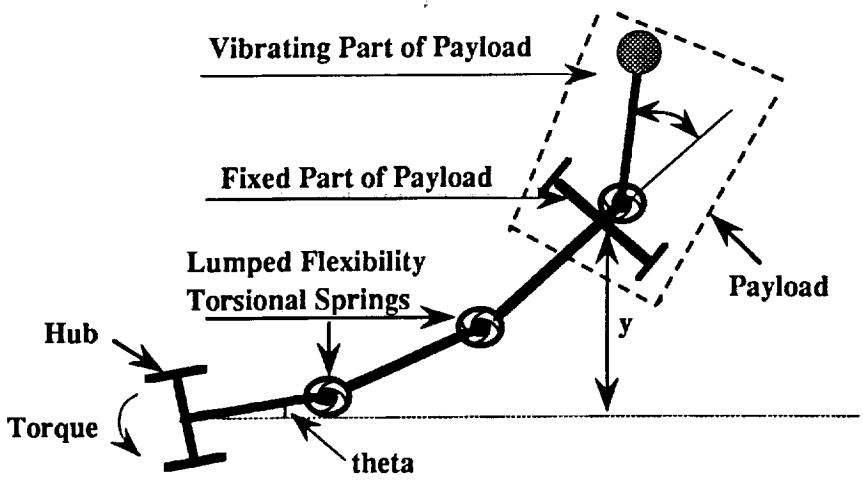



Figure 6.1: **Pseudo-Finite-Element Model of a Flexible Beam**

*The flex arm is grasping a dynamic object which is modeled as an torsional spring and inertia rod*

PFEM model has the advantage that it is easy to produce linear equations of motion for simulation.

Figure 6.1 also shows the payload which consists of two parts. There is a fixed part of the payload which is rigidly attached to the arm and a vibrating part which is represented in the model by a mass on a rod attached to a torsional spring. It is important to note the torsional spring in the payload is not the same as those used to model the arm. The torsional spring and mass of the payload represent the slosh of fluid. Modeling the payload as a torsional spring and inertia rod is easily incorporated into the system model when the PFEM method is used.

Note that the PFEM is not the ultimate way to model a beam but simply a quick method to gain insight into the problem.

### 6.2.2 Modeling Results

Now that the modeling technique has been developed, the effects of the dynamic payload on the system dynamics. There are two cases of interest can be analyzed. Case one is a rigid payload where we simple make the payload spring constant infinite. Case two is when the payload is not rigid and has a finite spring constant. This represents the case where someone has a black box which can be weighed, but the contents are unknown. We will study the effect that the payload has on two transfer functions. First, the collocated transfer function is the transfer function from the torque at the hub to the angle the hub makes with an inertial reference (shown as *theta* in Figure 6.1). Next, the noncollocated

| Parameter | Value |
|---|---|
| Number of Elements | 6 |
| EI | 0.77 $Nm^2$ |
| Length | 0.75 m |
| Beam mass | 0.3986 kg |
| Hub inertia | $7.85x10^{-4}$ $kg/m^2$ |
| Modal damping | 1% |
| Fixed tip mass | 1 kg |
| Oscillating tip mass | 0.5 kg |
| Freq of payload | 3 Hz |

Table 6.1: Beam and Payload Parameters

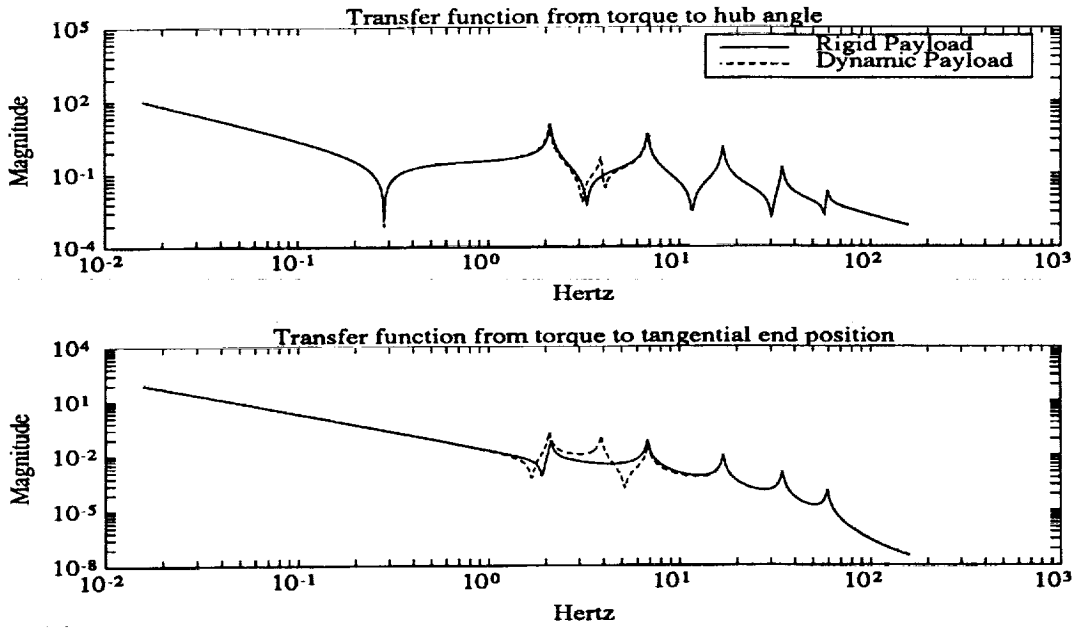



Figure 6.2: **System Frequency Response**

*Frequency responses are shown for the flex arm with rigid and nonrigid payloads*

transfer function is the transfer function from the torque at the hub to the tangential end position of the tip (labeled $y$ in Figure 6.1).

The results are shown in Figure 6.2. The parameters used to make this plot are shown in Table 6.1. Note that the frequency of the payload is slightly higher than the fundamental frequency of the arm.

The dynamic payload introduces an extra mode at approximately the frequency of the payload which one expects. The introduction of an unmodeled mode at a frequency near

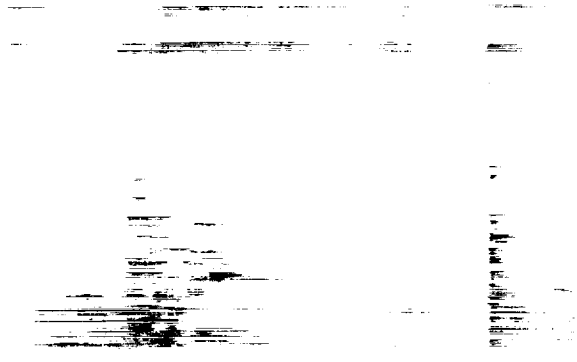
the controller bandwidth can lead to an unstable LQR design. In fact for the case shown an LQR controller designed for a rigid payload was seen to go unstable with the dynamic payload. It is important to remember that the masses and inertias of the objects are the same. The introduction of the payload vibrational frequency alone causes the above mentioned effects.

In summary, it has been shown that a dynamic payload significantly alters the frequency response of a flexible beam. This can cause end-point controllers to become unstable if the dynamics of the payload are not accounted fro in the controller design.

## 6.3 Future Work

There is a lot of work ahead. The following is just a partial list of things to do.

- Develop other models such as assumed-modes model or finite-element model.
- Develop an estimator in order to test end point controllers.
- Design and construct the hardware for the experiment. Some of which has already been done.
- Design a controller to control a dynamic object with full knowledge of the dynamics.
- Study and review possible adaptive or robust control techniques for applicability to the problem.



~

# Bibliography


[1] Arthur E. Bryson, Jr. *Control of Spacecraft and Aircraft.* To be published.

[2] Robert H. Cannon, Jr., Marc Ullman, Ross Koningstein, Stan Schneider, Warren Jasper, Roberto Zanutta, and William Dickson. NASA Semi-Annual Report on Control of Free-Flying Space Robot Manipulator Systems. Semi-Annual Report 8, Stanford University Aerospace Robotics Laboratory, Stanford, CA 94305, February 1989.

[3] John J. Craig. *Introduction to Robotics Mechanics and Control.* Addison-Wesley, Reading, MA, 1986.

[4] Warren Jasper. *Experiments in Thrusterless Robot Locomotion Control for Space Applications.* PhD thesis, Stanford University, Stanford, CA 94305, September 1990. Also published as SUDAAR 595.

[5] Ross Koningstein. *Experiments in Cooperative-Arm Object Manipulation with a Two-Armed Free-Flying Robot.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, October 1990. Also published as SUDAAR 597.

[6] Raymond H. Kraft. *Experiments in End-Point Control of a Flexible Robot with a Mini-Manipulator.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, May 1989. Also published as SUDAAR 581.

[7] Leonard Meirovitch. *Analytical Methods in Vibrations.* Macmillan Series in Applied Mechanics. Macmillan Company, New York, NY, 1967.

[8] Daniel Mark Rovner. *Experiments in Adaptive Control of a Very Flexible One-Link Manipulator.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, August 1987.