Experiments in Cooperative Manipulation: A System Perspective

Stanley A. Schneider
Robert H. Cannon Jr.

Stanford University Aerospace Robotics Laboratory
Stanford, California 94305

Abstract

This paper outlines an experiment in cooperative robotic manipulation conducted at Stanford's Aerospace Robotics Laboratory. In addition to cooperative dynamic control, the system incorporates real-time vision feedback, a novel programming technique, and a graphical high-level user interface. By focusing on the vertical integration problem, we are examining not only these subsystems, but also their interfaces and interactions.

The control system implements a multi-level hierarchical structure; the techniques developed for operator input, strategic command, and cooperative dynamic control are presented. At the highest level, a mouse-based graphical user interface allows an operator to direct the activities of the system. Strategic command is provided by a table-driven finite state machine; this methodology provides a powerful yet flexible technique for managing the concurrent system interactions. The dynamic controller implements "object impedance control"—an extension of Nevill Hogan's impedance control concept to cooperative-arm manipulation of a single object.

Experimental results are presented, showing the system locating and identifying a moving object, "catching" it, and performing a simple cooperative assembly. Results from dynamic control experiments are also presented, showing the controller's excellent dynamic trajectory tracking performance, while also permitting control of environmental contact forces.

1 Introduction

This paper presents an overview of the Dynamic and Strategic Control of Co-Operating Manipulators (DASCCOM) project at Stanford's Aerospace Robotics Laboratory. Due to space constraints, this paper can only present a very brief overview of the system capabilities; more technical detail and experimental results can be found in [9,8,2]. Space considerations also prohibit an extensive literature review; this work draws most heavily on [5,7,3,6]; related research can be found in [1,4,11].

Research goals Space construction requires the manipulation of large, delicate objects. Single manipulator arms are incapable of quickly maneuvering these objects without exerting large local torques. Multiple cooperating arms do not suffer from this limitation. Unfortunately, utilizing multiple manipulators introduces many additional problems, among them dynamic complexity and difficult strategic command.

The goal of this project is to study simultaneously the dynamic and strategic issues of cooperative manipulation, and to demonstrate experimentally a cooperative robotic assembly. We aim not only to master the dynamic control problem, but also to provide for simple, conceptual *direction* of motion by an untrained operator.

Summary of results The DASCCOM project is now essentially complete, and is capable of performing simple assembly operations. Implementation of a complete multiple-robot hierarchy has resulted in several new insights into manipulation and interacting real-time systems. This system integrates for the first time:

- An "object-only" task-specification graphical user interface.
- Finite state table programming, a structured technique for managing asynchronous real-time events and interacting processes.
- Object impedance control, a cooperative dynamic controller that enforces a specified *object behavior*.
- A real-time "point-tracking" vision system, capable of identifying and tracking multiple objects.

2 Experimental Dual Manipulator System

The experimental system is designed to emulate a dual-armed space robotic vehicle. Dual two-link robotic arms (fixed-base in these initial experiments) can manipulate small, freely-floating air-cushion vehicles. The vehicles are equipped with connectors that can mate with several docking ports in the manipulators' workspace. The system thus simulates, in two dimensions, the weightless space manipulation and assembly problem.

Mechanical Hardware The experimental facility consists of a pair of two-link manipulators, affixed to the side of a "small" granite table (4 feet × 8 feet). Each arm is of the popular SCARA configuration—basically anthropomorphic, with vertical-axis, revolute "shoulder" and "elbow" joints. The arms are equipped with joint angle sensors and endpoint force sensors. An overhead television camera provides global vision. A photograph of the experimental setup appears in Figure 1.

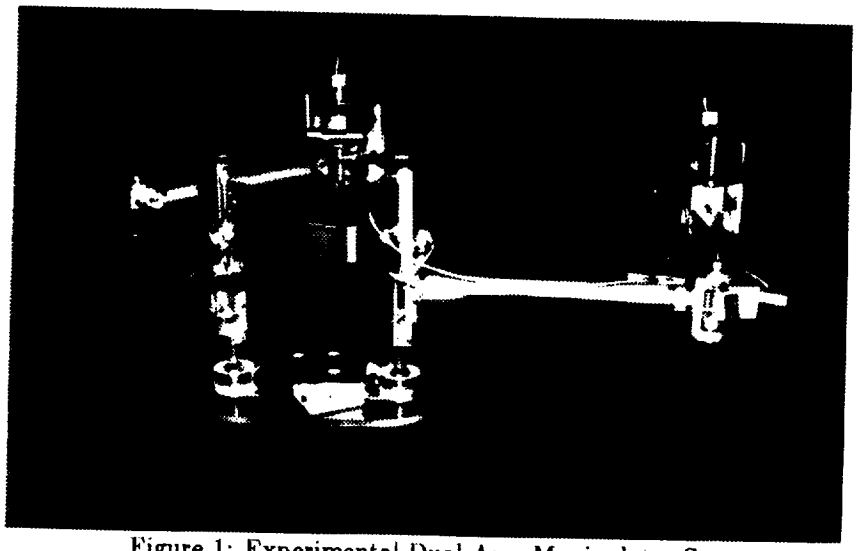


Figure 1: Experimental Dual Arm Manipulator System

Computer System Our real-time computer system combines a proven UNIX development environment with high-performance real-time processing hardware. Motorola 68020/68881 single board processors running the pSOS real-time kernel provide inexpensive real-time processing power. VME bus shared-memory communications permit efficient multiprocessor operation. The real-time processors are linked, via the VME bus, to our Sun/3 engineering workstations. Thus, we benefit from Sun's superb programming environment, while providing the capacity for relatively cheap, unlimited processing expansion.

Real-time software environment Each real-time processor runs pSOS; a small, fast, priority-driven multi-tasking kernel [10]. The features used most heavily by our software structure are the multi-tasking scheduler, the inter-process message facility, and the event-signal facility. We have also developed a large array of in-house real-time software to support control-systems research, [8] presents details.

3 System Structure

The cooperating arms application software consists of four major modules: user interface, strategic control, dynamic control, and vision. The dynamic control module is further divided into three sub-modules: the object impedance controller and dynamic controllers for each of the two arms (see Figure 2). Execution of these modules is spread over three real-time 68020 processors and the Sun workstation.

The command hierarchy The user interface collects conceptual-level commands from the operator, and communicates them to the strategic controller.

The strategic control module is responsible for the overall command of the system. It fields high-level requests from the user interface module, and translates them into sequences of primitives that the dynamic control module can implement. It also monitors the various conditions and activities of the system and directs appropriate response actions.

The dynamic control module is responsible for reading the various sensors and calculating the actuator torques required to produce the desired system behavior.

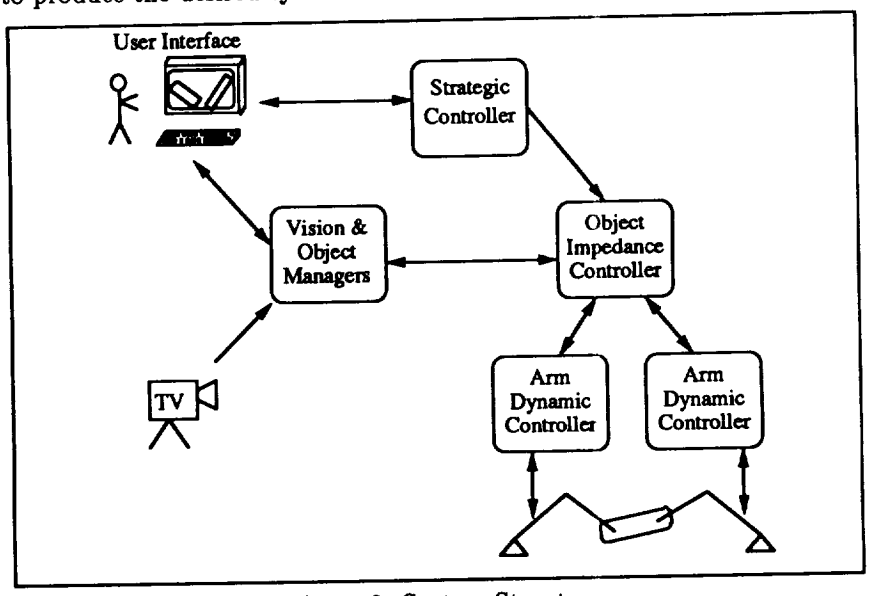


Figure 2: System Structure

System data-flow Data communications between modules is accomplished via simple global data structures. An interprocessor lock gate is provided with each structure. Two major databases are maintained: the "object" database contains physical properties and current states of the objects in the workspace (world model), and the "arms" database contains the desired and measured states (positions, endpoint forces, etc) of each arm.

Since command or temporal information flow is not efficient with shared data structures, bidirectional byte-stream "channels" are also provided for interprocessor (or interprocess) communications.

4 User Interface

The purpose of the user interface is to gather conceptual commands from the user, and communicate them to the strategic control module. The user interface should present a clear and intuitive means for the user to specify his wishes.

Modes of operation Users often wish to communicate with the system at different conceptual levels. To provide this, the user interface of DASCCOM provides two modes of operation: autonomous execution of common operations, and manual "teleoperated" manipulation for unusual tasks. This combination allows the completion of most assembly tasks.

In automatic mode, two "views" of the object being manipulated are displayed. The actual position of the object is displayed by a solid-lined iconic figure. In addition, a "desired" position of the object is drawn with broken lines. The user can move the broken-line (ghost) object by clicking on it and dragging it around the screen.

At all times, the ghost object represents the state that the system will attempt to produce if the mouse button is released. It is thus a one-move preview of the new state of the system. For instance, to perform a connection operation, the user can simply point to any connector on an object, and drag it to any matching connector in the workspace. The display will show the ghost object in the final connected position. When the move is confirmed, the system performs the sequence of actions required to make the connection, and reports the status back to the user. This allows quick, simple assembly operations.

In recognition of impossibility of anticipating all actions, a manual operation mode is also provided. In manual mode, no ghost object is displayed. Instead, manual mode allows direct access to the object impedance dynamic controller.

Both modes utilize an *object-only* interface; the system takes responsibility for all arm motions.

Interface to the strategic controller The user interface executes on the Sun workstation. It communicates with the strategic control module on the real-time system via a bidirectional byte-stream channel through VME-bus shared memory.

The protocol utilizes a "request/response cycle" pattern; the user interface requests an action, and the strategic controller returns a status response when the action completes. The simplicity of this communication paradigm allows considerable flexibility.

A brief operational description Several frames from a typical session are presented as Figure 3. The screen is divided into three sections. The large lower section depicts the manipulator workspace in iconic form; the activities of the system are visually displayed here. The upper left window displays the system status; the first line in this window gives a short verbal description of the systems activity. A system control panel forms the upper right section.

In the upper right (second) frame, for example, there are two objects in the vision system's field of view. Scooter is the floating air-cushion object. Scooter has two gripper attachment ports and two male connectors. Multibase is a stationary object with female connectors. The arms are currently holding Scooter, as evidenced by the presence of the Scooter ghost image. The user has just dragged the ghost's right connector over to Multibase's rightmost connector. The status line indicates that the insertion of one of Scooter's connectors into one of Multibase's connectors is in progress.

Note that the arms do *not* appear in the display. The operator commands only object motions; the arm actions required to effect these motions need not be specified.

An example task: install a part For the sake of this example, suppose that "Multibase" is affixed to a mobile robot, and represents a series of attach points for holding miscellaneous items. "Scooter" is a part that is to be installed into a remote module, represented by "Dock".

In the upper left frame, the part is approaching the robot system[1]. The operator has just indicated Scooter is to be grasped, thus the "Acquiring Scooter" status message in the top left corner. In the following frame, the operator indicates that Scooter should be affixed to the robot's base. Next, the robot is directed to navigate to the vicinity of Dock (navigational control is not discussed here). When Dock is in view, the operator indicates that the new part (Scooter) is to be installed. The lower left frame shows the first stage of that action. Finally, in the last frame, the part has been released, and the installation is complete.

This entire procedure was accomplished with only four simple mouse motions. (Click on the approaching Scooter, connect Scooter to Multibase, connect Scooter to Dock, release.) Most of the assembly details—such as how to attach and detach connectors, how fast to approach the docking connector, etc.—are completely automated.

Summary: User interface In summary, the DASCCOM user interface features:

- A simple mouse-based "point and click" graphical interface.
- "Object-only" task specification; manipulation details are left to the system.
- One-step operation previewing.
- Automatic execution of most assembly operations.
- Optional "manual" manipulation at the dynamic object control level.

[1] Or, equivalently, the robot is approaching the part.

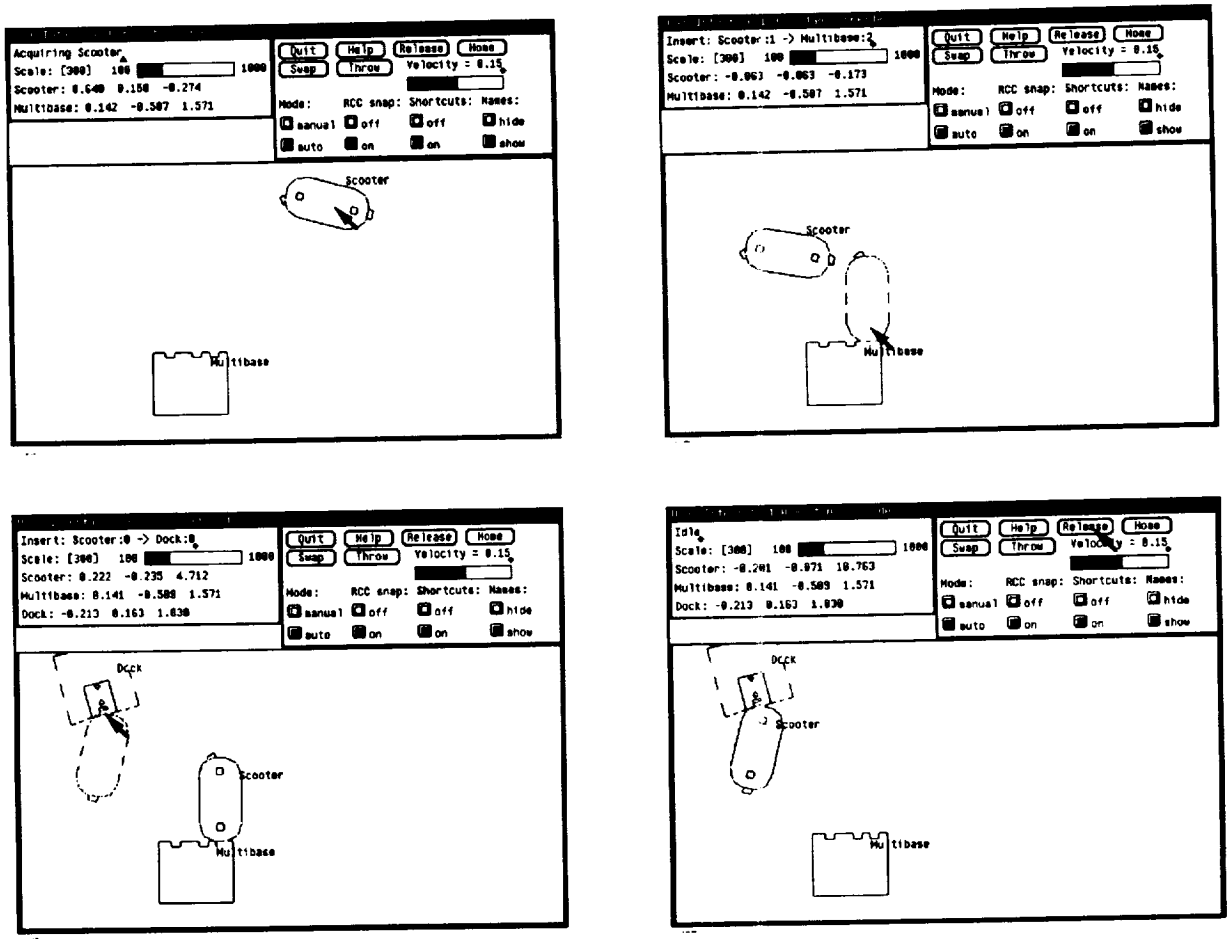


Figure 3: Installation Demonstration Example

5 Strategic Control

The purpose of the strategic controller is to provide an interface between the conceptual commands provided by the user interface and the dynamic control module. It furnishes the user interface with a set of simple automated commands, allowing it to perform many tasks.

An integrated system involving multiple manipulators, real-time vision, and interactive operator control is a complex, event-driven environment. These systems are naturally concurrent in nature; complex asynchronous interactions must be managed. With a fundamentally sequential underlying programming paradigm, the burden of managing these asynchronous events is left to the programmer. A sequential execution stream model can not, for instance, deal effectively with multiple-arm synchronization, especially if the arms are running different programs on different processors.

The state table programming technique This section presents an alternative programming methodology, referred to as state table programming. It provides a naturally event-driven structure to guide the programmer in producing code that is easily interfaced to other real-time system modules. It directly exploits the facile multiple-process generation and communications provided by modern real-time kernels. In fact, management of multiple asynchronous events is central to the structure of the system; the programmer is actively encouraged to divide the problem into small, independently executing programs. In addition, it is based on a very intuitive task description technique.

The state table programming technique is neither a robot programming language nor a replacement for a library of robot control routines. For instance, trajectory generation utilities and world modeling utilities are also required. The technique merely provides a framework that weaves the multiple streams of execution in the system into an easy-to-use structure.

State transition graphs State transition graphs provide a simple, intuitive means of visualizing the sequence of actions required to effect a task. A "state" is usually characterized by the system performing a single "step" of an operation, and waiting for some indication of its completion. State transitions are indicated by arrows labeled with the event that causes the transition. When an event occurs, a transition routine is executed. The result of that routine determines the next state entered.

For example, Figure 4 presents a simplified series of steps required to catch a moving object. The system starts out in the "Idle" state. The receipt of the "Acquire" stimulus, presumably sent from a higher level (e.g. the user interface), causes the system to enter one of two states: "Reaching" if the object is within reach, or "Waiting", if not. While the system is in the "Reaching" state, the arms are executing an intercept trajectory. When the trajectory completes ("TrajComplete" event) , the arms track the object until the gripper endpoints match the targeted grip ports precisely. At this point, the system enters the "Gripping" state while the grippers engage. Finally, the catch is complete, and the "Manipulating" state is active until a "Release" command is received.

This is a rather simple example. Much more complex series of actions are easily representable.

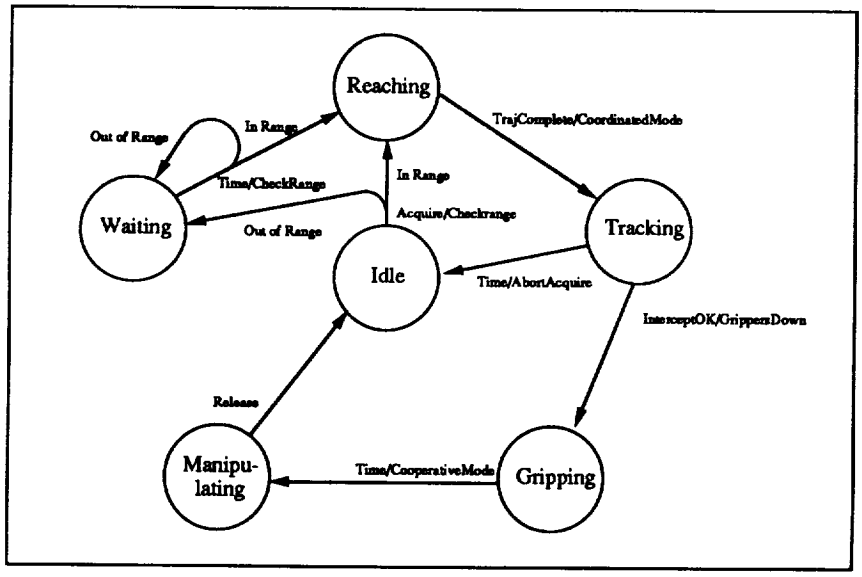


Figure 4: State Transition Graph - Catch Task

State table programming The state table entries corresponding to the states of Figure 4 are presented in table 1. Note that multiple pending conditions are handled very naturally, for example, the "Tracking" state waits for either "InterceptOK" or "Time".

State	Stimulus	Transition Routine	Next States	
Idle	Acquire	CheckRange	Reaching	Waiting
Waiting	Time	CheckRange	Reaching	Waiting
Reaching	TrajComplete	CoordinatedMode	Tracking	
Tracking	InterceptOK	GrippersDown	Gripping	
	Time	AbortAcquire	Idle	
Gripping	Time	CooperativeMode	Manipulating	
Manipulating	Release	GrippersUp	Idle	

Table 1: State Transition Table

Stimuli may be generated by any of the modules of the system, but most originate from one of two sources: command stimuli from the user interface, and condition-event stimuli from independently executing "helpers". The "Acquire" and "Release" stimuli are examples of the former. The "Time" stimulus is created by a simple helper process that sleeps for a specified time before sending its message. Other helpers (not shown in the table) perform "safety checks"; for example, the "OutOfRange" stimulus is sent by a process that checks every so often to insure that the object being acquired is not suddenly moved out of range.

Implementation The heart of the system is the Finite State Machine (FSM) driver. The FSM driver is an independently executing process. It acts as a central command post, receiving messages from many sources in the system and taking the appropriate action. Each message contains a stimulus code; the FSM driver uses that code to reference the state table and select a state transition routine to execute. The state transition routines perform the actual work: they change controller modes, start and stop helper processes, and interact with the trajectory generation module.[2]

The DASCCOM strategic control module supports automatic object capture, docking (connector insertion), withdrawal, and throwing functions. Each task is executed as a multi-step chain of state transitions.

Summary: Strategic control The DASCCOM strategic controller is based on an event-driven finite state machine. This technique features:

- An intuitive, graphical task specification.
- Direct tabular implementation.
- A natural, event-driven structure.

6 Cooperative Dynamic Control

The dynamic equations of motion of multiply-armed robotic systems are complex. Strategic control of the system interactions is also difficult; a consistent set of desired motions must be specified. As of yet, no satisfactory method of precise dynamic control coupled with a simple strategic command interface has been developed and experimentally demonstrated.

This section outlines a strategy for the control of a cooperative robotic system that permits high performance dynamic motion control, while also allowing direct control of environmental interactions. This is accomplished by controlling the manipulated *object* to react to external environmental stimuli with a programmable impedance. This facilitates motion direction by presenting a simple yet powerful interface; the strategic controller need only specify the impedance. Although "exact" inertial force compensation is achieved, the control structure does not require explicit formulation of the closed-chain dynamic equations of motion, and is amenable to parallel computation. Object internal forces are explicitly controlled. The object impedance controller has been implemented on a multi-processor real-time computer system. Experimental results are presented in section 8 to verify the controller's performance, both for free-motion slews and environmental contact.

Control objective Hogan's impedance control policy [5] causes the endpoint of the manipulator to react to external forces with a programmable impedance. The simplest example both to understand and implement is a simple second order linear impedance—the endpoint behaves as a mass attached via a virtual spring-damper to the environment. DASCCOM utilizes a dynamic controller that enforces a controlled impedance not of the arm *endpoints*, but of the manipulated *object* itself. Intuitively, the object behaves as if it were attached to its environment by linear spring-damper systems in the linear degrees of freedom, and also by uncoupled torsional spring-dampers to control rotational orientation.

The object impedance controller enforces (for a simple linear second-order impedance) the relationship:

$$m_d(\ddot{x} - \ddot{x}_{des}) + k_v(\dot{x} - \dot{x}_{des}) + k_p(x - x_{des}) = f_{ext}$$

Here x denotes the coordinate of any one degree of freedom of an *arbitrary* point fixed in the object's frame. The constants m_d, k_v, and k_p are specifiable. The reference signal x_{des} denotes the desired position (or orientation) of the chosen point. The $\ddot{x}_{des}$ term represents acceleration feed-forward. Thus, the programmable impedance force corrects deviations from the desired trajectory.

Intuitively, this control policy completely supplants the actual dynamics of the object with a "virtual" object, with specifiable mass and inertia properties[3]. The "virtual" object is attached at its (apparent) center of mass via an orthogonal set of imaginary damped springs to a selectable point in the environment. Thus,

[2] The addition of callable sub-chains would add considerable power to the implementation; it is under development.

[3] Of course, this can only be done within the bandwidth and actuation limits of the system.

the object can be manipulated by simply moving the virtual spring endpoint. Controlled force interactions with environmental obstacles can be done by simply pressing the "spring" against the obstacle. Thus, both free motion slews and manipulation requiring contact can be done with the same strategic interface.

The position and orientation of the "virtual" object with respect to the actual object is also selectable. This selectable "command frame" allows simple specification of many operations. A particularly useful example is for performing connector insertions—by placing the "virtual" object frame at a fixed location in the connector frame, all assembly operations can be specified as connector motions only. Multiple connectors arranged on an object in arbitrary orientations can then be handled by the same simple connector insertion algorithm. Since the stiffness is selectable, this controller is also capable of pseudo remote center of compliance (RCC) operation [12], permitting simple and efficient part mating and insertion operations.

Summary: Dynamic Control Space constraints prohibit derivation of the controller here, see [9] for a more complete treatment. The controller features:

- A simple, powerful *object behavior* specification interface.
- Good dynamic performance, both in free motion and in contact, without switching controllers.
- Exact dynamic compensation, without requiring closed-chain equations of motion.
- A selectable command frame, facilitating assembly operations.

7 Real-time Vision System

To track moving objects, DASCCOM employs a high-speed television camera-based point-tracking vision system. The vision system uses a 60 Hz shuttered CCD television camera to track special variable reflectivity targets. Each object to be tracked is outfitted with a unique pattern of these targets. The vision system software is then able identify and track individual objects in real-time. Simple linear state estimators also provide velocity estimates. The arm endpoints are also fitted with targets, and the information is used to allow endpoint feedback control.

Space considerations prohibit further description here; the system is described in detail in [2]. In summary, the vision system capabilities are:

- Real-time (60Hz) *object* position and orientation.
- Sub-pixel resolution (about $1/20^{th}$ of a pixel).
- Multiple object identification and tracking.

8 Experimental Results

A moving object "catch" A "strobe" sequence picture of a typical catch is presented as the left side of Figure 5. The vision system provides high-speed data for three subsystems to perform this task: the two arms, and the moving body. Each arm is under vision-guided endpoint control. The desired trajectory for each arm takes it from its initial "home" position, to an intercept state that matches the object's gripper port in both position and velocity.

To perform a successful capture, the arm endpoint must then be held over the gripper port for the duration of the time required for the gripper mechanism to engage. The positioning must be fairly accurate.

The right side of Figure 5 shows the vertical positions and velocities of the right arm and right gripper port during the catch. The object is being accelerated toward the arm system for the first second. During the next second, the arm tip accelerates to match the port position and velocity at about the 3.2 second mark[4]. The gripper's downward motion occupies the next second, after which the object is brought to a halt. After the grippers have engaged (at about 4.4 second mark) the observer has an incorrect plant model; this accounts for the apparent difference in position and velocity after the capture.

Before the grippers engage, they are under independent endpoint impedance control; after the object is caught the object impedance controller is in effect. Thus, compliant control is active at all times—this prevents large acceleration forces during and after the acquisition.

[4] The arm trajectories are actually updated several times as the object position and velocity estimates improve.

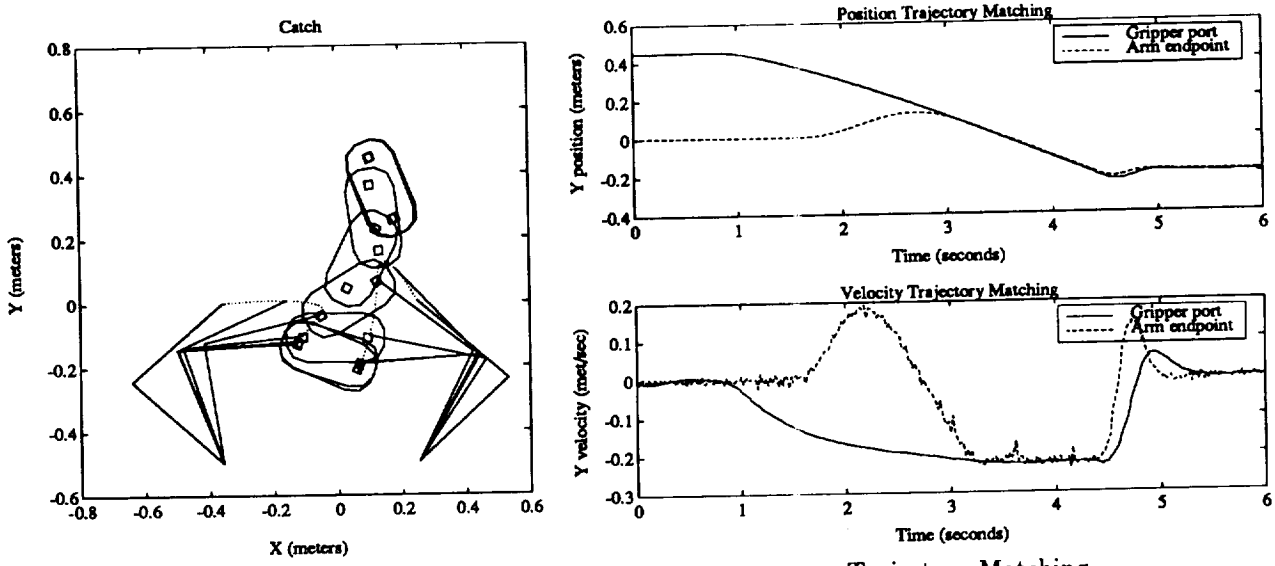


Figure 5: A Two-handed Catch

Transport trajectory tracking Five cooperative control algorithms were compared for a cooperative transport task. The five algorithms are: co-located proportional-derivative (PD) joint-space coordinated position control, dynamic and kinematic coordinated endpoint impedance control, and dynamic and kinematic object impedance control. Space constraints only allow presentation of results from two of the controllers here; see [9,2,8] for more details. The commanded reference is a fifth-order trajectory of the center of mass of the object in each object degree of freedom: x, y, and θ. All algorithms were provided with the correct coordinated position, velocity, and acceleration references for the entire slew path.

Figure 6 compares the PD controller to the dynamic object impedance controller. The upper-left plot for each controller depicts the motion of the center of mass of the y direction. The lower-left is the corresponding velocity. The upper-right plots x vs. y, and indicates the desired and actual object positions at 0.5 second intervals during the motion. The lower-right plot shows the magnitude of the "tension" between the arms, after being corrected for dynamic forces. Both controllers are attempting to maintain zero tension.

The PD controller does a poor job of following the desired trajectory and offers no control of the internal forces on the object. This controller also does not compensate for inertial forces.

Since the object impedance controller correctly compensates for the object dynamics, the trajectory tracking performance is quite impressive. The inter-arm tension is controlled well also.

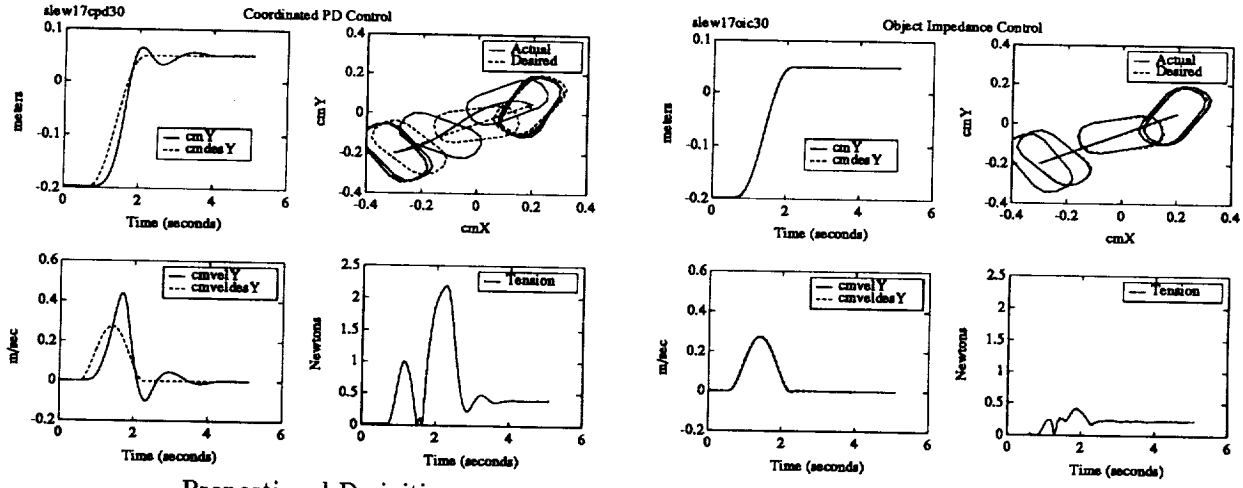
Force control performance Force control data (Figure 7) were obtained by simply placing a hard, stationary object in the path of the object during a slew. The unfiltered data exhibits some ringing—this is an impact between two very hard objects—but the force level quickly settles to the desired. The important thing to note is that the object impedance controller successfully controls the forces of interaction, without switching control modes, even when it comes into contact with a very stiff environment.

9 Conclusions

This paper has presented an overview of the DASCCOM system. This system integrates strategic and dynamic control of cooperating manipulators with a real-time vision system and a graphical user interface. The techniques developed have been experimentally proven, and will provide a basis for the Stanford Aerospace Robotics Laboratory's future space robotics research.

References

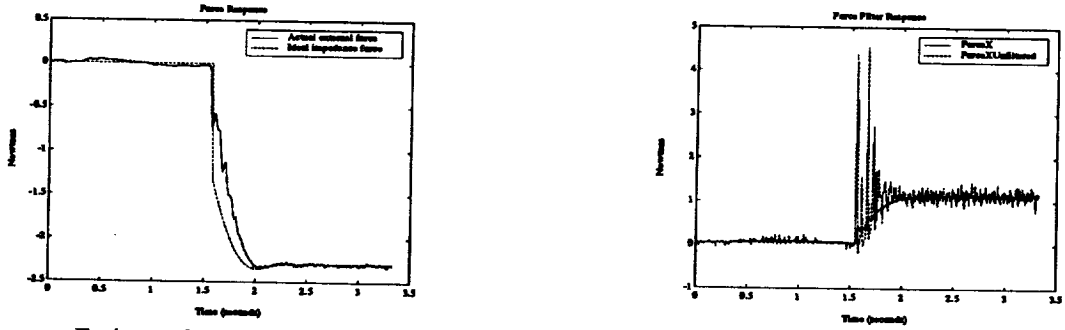
[1] T. E. Alberts and D. I. Soloway. Force control of a multi-arm robot system. In *Proceedings of the International Conference on Robotics and Automation*, pages 1490 - 1496, Philadelphia, PA, April 1988. IEEE.



Proportional-Derivitive

Object Impedance Control

Figure 6: Controller Slew Performance Comparison



Estimated External Force

Right Arm Force Sensor Output

Figure 7: Impact Force Control Performance

[2] R. H. Cannon, Jr., M. Ullman, R. Koningstein, S. Schneider, W. Jasper, R. Zanutta, and B. Dickson. NASA Semi-Annual Report on Control of Free-Flying Space Robot Manipulator Systems. Semi-Annual Report 7, Stanford University Aerospace Robotics Laboratory, Stanford, CA 94305, August 1988.

[3] S. Hayati. Hybrid position/force control of multi-arm cooperating robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 82 – 89, San Francisco, CA, April 1986. IEEE.

[4] G. Hirzinger and J. Dietrich. Multisensory robots and sensor-based path generation. In *Proceedings of the International Conference on Robotics and Automation*, pages 1992 – 2001, San Francisco, CA, April 1986. IEEE.

[5] N. Hogan. Impedance control: An approach to manipulation, parts i, ii and iii. *Trans of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 107:1–24, March 1985.

[6] O. Khatib. Object manipulation in a multi-effector robot system. In *ISRR*, Santa Cruz, CA, 1987.

[7] Y. Nakamura, K. Nagai, and T. Yoshikawa. Mechanics of coordinative manipulation by multiple robotic mechanisms. In *Proceedings of the International Conference on Robotics and Automation*, pages 991 – 998, Raleigh, NC, April 1987. IEEE.

[8] S. Schneider. *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators.* PhD thesis, Stanford University, Stanford, CA 94305, May 1989.

[9] S. Schneider and R. H. Cannon. Object impedance control for cooperative manipulation: Theory and experimental results. In *Proceedings of the International Conference on Robotics and Automation*, Tempe, AZ, April 1989. IEEE.

[10] Software Components Group, Inc., 4655 Old Ironsides Drive, Santa Clara, CA 95054. *pSOS - 68K Real-Time, Multiprocessing Operating System Kernel User's Manual*, 4.1 edition, December 1986.

[11] M. Uchiyama, N. Iwasawa, and K. Hakomori. Hybrid position/force control for coordination of a two-arm robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 1242 – 1247, Raleigh, NC, April 1987. IEEE.

[12] D. E. Whitney and J. L. Nevins. What is the remote centre compliance (rcc) and what can it do? *Robot Sensors*, 2:3–15, 1986.