November 1989

A Two-Arm Exploratory System for Identifying Moving and Removable Parts

Xiaoping Yan
University of Pennsylvania

Robert King
University of Pennsylvania

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

R. Vijay Kumar
University of Pennsylvania, kumar@grasp.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-75.

A Two-Arm Exploratory System for Identifying Moving and Removable Parts

Abstract

When working in an unstructurcd environment, a robot has partial or no *a priori* knowledge of the environment. The purpose of exploratory robotics is to provide robots with the ability to learn and automatically construct models of the environment by exploring and interacting with the environment. This paper describes a two-arm exploratory system whose purpose is to identify movable and removable parts of an object, and the mobility of the parts. The system is implemented by integrating RCI (Robot Control Interface) with Timix (a real-time kernel). The identification is accomplished through *exp1oratory procedures* which are generated from a numbcr of robust *motion primitives*.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-75.

A TWO-ARM EXPLORATORY SYSTEM for IDENTIFYING MOVABLE and REMOVABLE PARTS

Xiaoping Yan
Robert King
Insup Lee
Vijay Kumar

MS-CIS-89-75
GRASP LAB 198

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

November 1989

ACKNOWLEDGEMENTS:
The authors wish to thank Ruzena Bajcsy and Mario Campos for useful discussions. This work was in part supported by Airforce grant AFOSR 88-0244, AFOSR 88-0966, Army/DAAG-29-84-K-0061, NSF-CER/DCRS2-19196 Ao2, NSF CCR-8716975, NASA NAG5-1045, ONR SB-35923-0, NIH 1-RO1-NS-23636-01, NSF INT85-14199, ARPA N0014-88-K-0630, NATO grant No. 0224/85, DuPont Corp., Sandia 75 1055, Post Office, IBM Corp. and LORD Corp.

American Society of Mechanical Engineers
Winter Annual Meeting, December 1989.

A Two-Arm Exploratory System for Identifying Movable and Removable Parts

Xiaoping Yun, Robert B. King, Insup Lee, and Vijay R. Kumar

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

ABSTRACT

When working in an unstructured environment, a robot has partial or no *a priori* knowledge of the environment. The purpose of exploratory robotics is to provide robots with the ability to learn and automatically construct models of the environment by exploring and interacting with the environment. This paper describes a two-arm exploratory system whose purpose is to identify movable and removable parts of an object, and the mobility of the parts. The system is implemented by integrating RCI (Robot Control Interface) with Timix (a real-time kernel). The identification is accomplished through *exploratory procedures* which are generated from a number of robust *motion primitives.*

1 Introduction

In the past several years, there has been rapidly increasing research effort on cooperative multiple robot manipulators. Many advantages of multiple manipulators have been justified. These advantages include large payload capacity, ease of handling large and/or flexible objects, assembly tasks, deep-sea exploration, service tasks in outer space [5], etc. A large number of issues on coordination of multiple manipulators have been identified and addressed such as constraint relations [30], load balance [19,7], modeling and control [16,17,18,1,12,25,3,6]. In dealing with the design of coordinated control systems for multiple manipulators, it is generally assumed that a complete model of the robots and the environment is available. For instance, if motions of multiple manipulators are constrained by the environment, it is normally assumed that certain parameters of the environment such as the geometrical model and the coefficient of friction are available for analysis and design of controllers. If two manipulators grasp an object which has mobile parts (e.g. a pair of scissors), the degrees of freedom and types of moving joints of the parts are assumed to establish constraint relations [30,24]. Such assumptions will not be valid for robot manipulators that operate in an unknown environment.

In this paper, we describe an experimental robotic system that is designed to identify properties of the environment. There are many parameters associated with the environment. To make the problem manageable, we restrict ourselves to kinematic properties of the environment. More specifically, for an object that is possibly made up by multiple parts, we are interested in identifying the relationship among the parts (hereafter referred to as the parts relationship). We will determine if a part is removable from the rest of the object. If not, we will identify the relative mobility of the parts including types of possible motions (e.g., sliding or rotary) and degrees of freedom.

Vision is less useful for identifying parts relationship, though it provides a starting point. Our early study [8] shows that two cooperative manipulators form an effective sensory device for identifying the parts relationship through manipulating and interacting with the object. It is important to view the manipulators themselves as part of the sensory device. To detect changes in displacement and forces, it is necessary to equip the manipulators with position and force sensors.

The entire process of identifying the parts relationship is in two steps: *manipulation* and *recognition.* The former requires movement of manipulators, application of forces in certain directions, and recording position and force data. The latter involves the representation of the parts relationship and algorithms inferring the parts relationship from the data. These two steps are repeated until the parts relationship of the object is identified. The focus of this paper is on the first step, that is, on building an experimental control system for two manipulators which is able to generate all exploratory movements.

2 Experiment Set-up

Figure 1 illustrates our experiment set-up. We utilize two robot manipulators at this time to implement a cooperative control system for performing exploratory tasks. For this purpose, two existing PUMA 560 robot manipulators are used in this experiment. They are installed close enough to provide a reason-

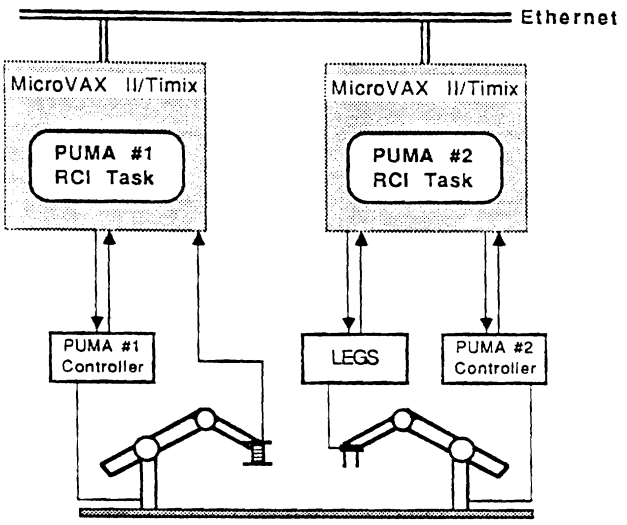

Figure 1: Experiment Set-up of Two Manipulators

ably large common workspace. Based on previous experiments carried out on a single manipulator in the laboratory [27], an instrumented compliant wrist is extremely useful to introduce necessary passive compliance. The wrist also provides the controller with reaction forces and moments by means of measuring deflection. The PUMA manipulator on the left (PUMA #1) is equipped with such a compliant wrist while the other manipulator (PUMA #2) has a Lord Experimental Gripper System (LEGS) attached. Each manipulator has a dedicated MicroVAX II for control development. The controller for PUMA #1 is attached to its MicroVAX via a parallel interface and the analog signals from the wrist are connected to an analog-to-digital converter in the same MicroVAX. For PUMA #2, two parallel interfaces are used: one for the controller and one for LEGS.

The creation of real-time control programs is made possible by supporting Robot Control Interface (RCI) [13] on both MicroVAX's. For real-time coordination of the two manipulators, it is essential for two processors/controllers to communicate with each other in real-time. For this purpose, the two processors are connected through an isolated Ethernet and are installed with Timix, a real-time kernel developed in the GRASP laboratory, instead of Unix. Timix provides real-time communication which makes it easy to coordinate and synchronize motions of the two manipulators. Integration of Timix and RCI is addressed below.

The goal of this experiment is to establish experimental data on exploratory tasks by using two cooperative manipulators. It is hoped that this experiment allows us to gain insight into problems in exploratory robotics. Since force control plays an essential role in interaction with environments and in coordination of multiple manipulators, emphasis is placed on further understanding force control of manipulators in absence of models of environments. Further, a set of motion primitives will be implemented, which are expected to span a basis for arm/hand movements needed for exploration.

2.1 Two-Arm Control Systems

Two-arm applications are naturally distributed and can be implemented in a distributed fashion across multiple processors. This allows the integration of additional sensors and arms, as needed, to build more complex systems. There are two basic architectures when developing a multi-robot system: tightly coupled processors and loosely coupled processors.

Two approaches have been undertaken using tightly coupled processors. Both are hardware dependent and have shared memory so that communication and synchronization occurs at a high rate. The Robot Instruction Processing System (RIPS) [15] allows multiple arms to be controlled by an hierarchical multiprocessor unit. The robot control problem is subdivided into separate tasks which are then mapped onto various custom and general purpose processors. A VME-bus is used to connect RIPS to a Sun 3 which serves as a host and executes planning programs.

Lloyd [13] developed RCI to provide a programmer with primitives that are used to write simple control procedures to operate the individual joints of a robot. It consists of two components: a *control* task which periodically executes at a high priority in a non-interruptable context to produce commands for each joint and a *planning* task which provides high level directives to the control task. The control task is restricted from executing UNIX system calls and its code and data pages are locked into memory. Using multiprocessor systems, RCI has been expanded to support multiple robots [14]. This approach uses a multiprocessor version of a MicroVAX II system where its configuration is augmented by at most three additional MicroVAX processors. One processor runs UNIX and responds to all external interrupts. The other three processors run a specialized minikernel which cannot respond to any external interrupts. Each processor executes one RCI control task.

A second approach to developing a multi-robot system uses loosely coupled processors. This approach is more portable and may be implemented on any communication medium (such as parallel lines or Ethernet). However, the communication overhead is greater than that of tightly coupled systems.

Our initial approach was to use two loosely coupled single processor MicroVAX II systems, each running RCI with UNIX and connected via an Ethernet. However, when experimenting with this approach, the communication delay was unpredictable since system calls (send and receive messages) can not be issued from within the control task. The unpredicatable communication is not acceptable for the coordination of two robot manipulators. To obtain the necessary predictable communication, we propose to replace the UNIX kernel with Timix [10]. Timix runs on a network of MicroVAX II processors which may be connected by Ethernet or ProNET (token ring). With this loosely coupled method, we may integrate other dedicated processors to provide additional sensory input – such as image processors.

2.2 Timix

Timix [10,11] supports processes with independent address spaces that execute, communicate and handle devices within timing constraints. *Signals* and *asynchronous message passing* are the two basic communication paradigms supported in Timix. New devices, which are directly controlled by application processes, can be integrated into the system without changing the kernel. In addition, dynamic timing constraints are used for scheduling processes and interprocess communications.

Signals Signals are the most basic way to communicate in Timix and are used by various other components of the kernel such as asynchronous messages, alarms, scheduling, devices and

system error reporting. Associated with each of the thirty-two possible signals is an integer value that can be used to pass a value. Whenever a signal is sent to a process, the appropriate signal handler of the receiving process is invoked to process the signal. To facilitate the use of signals, Timix allows them to be delayed, ignored, preempted, and prioritized.

Message Passing The notion of a port has been used widely for interprocess communication since it provides an easy to use communication abstraction [23]. Our kernel extends ports for real-time communication by allowing the sender to pass timing constraints with messages and the receiver to control message queuing and reception strategies. There are two types of ports: *reception* and *multicast*. A message sent to a reception port is received by one process, whereas a message sent to a multicast port is delivered to all reception ports connected to the multicast port. A process can either receive a message explicitly by invoking a receive system call on a port, or be notified of the arrival of a message through a signal associated with a port. For each port, the receiver may specify the ordering of pending messages and the size of the message queue as in [22].

Devices There are two kinds of devices: system devices, which are an integral part of the kernel; and applications devices, which are only pertinent to a particular application. System devices, such as clocks and network adapters, are managed by the kernel and used indirectly by many application processes. Each application device is directly controlled by a single application process. Application devices include the analog-to-digital conversion board required for the wrist, the parallel interface board required for each manipulator arm, and a parallel interface board for the Lord gripper. To control a device, a process requests the device from the device server. After the request is granted, the device and the process can share memory and device registers. In addition, a process may request to the device server that device interrupts be converted to signals.

2.3 Integration of Timix and RCI

A set of library routines that mimic the functionality of RCI have been implemented for Timix. This package requires no modifications to the kernel as was required to implement RCI on UNIX: each process is always resident in memory and the device interface allows new applications to be easily added. The same communication program that runs on the PUMA controller for RCI on UNIX is used for RCI on Timix.

Communication with the dedicated control processors (the PUMA controllers and LEGS) is accomplished with parallel interfaces. Each interrupt from the interface is mapped into a Timix signal to the appropriate RCI process. Whenever the signal occurs, its signal handler will execute. Since the handler is executing as part of a normal process context, it may execute any system call that is available to any application process. In particular, each RCI process can communicate with the other one by sending and receiving messages within its control task. This allows quick feedback of the state of each manipulator arm to the other one.

3 Two-Arm Exploration

The two-arm system that we are developing is for the purpose of exploring and interacting with the environment with the immediate goal of identifying the parts relationship of an object. This will generally require disassembling the object in contrast to assembly operations which are the focus of many publications. The key difference between assembly and disassembly in this context lies in availability of knowledge about the object such as size, shape, and the parts relationship. In assembly, the knowledge about the object is generally available up to certain degree and compliant control is applied to compensate the relatively small discrepancy. On the other hand, there is no *a priori* knowledge of the object required for exploratory disassembly. In this experiment, we develop a set of robust motion primitives for exploring and interacting with environments. These motion primitives are data driven and do not assume any knowledge of environments.

3.1 Motion Primitives

Observing how humans find the parts relationship of an object with the two hands, it has been recognized that hand movements fall into a number of patterns. To remove the cap of a pen, the two hands will pull, or twist, or combine the two (screw motion). Lederman and Klatzky [9] have called such patterns of hand movements as exploratory procedures and documented a set of exploratory procedures for identifying structure properties (shape, size, etc.) and material properties (texture, hardness, etc.) of objects. To cope with communication delays in telerobotics, Wilcox [26] has suggested to use a small number of "mechanical primitives" such as "move to contact," "rotate to edge contact," "slide to wall," etc. Since these mechanical primitives can be made robust, they allow a remote operator to perform many tasks despite communication delays and uncertainties.

Our approach to the design of control algorithms for two exploratory manipulators is similar to that of Wilcox's and also incorporates the concept of exploratory procedures. Therefore, analogous to human arms/hands, the manipulators identify a parameter of the environment by employing an appropriate exploratory procedure. For instance, *lateral motion* is an exploratory procedure for identifying the texture of a surface, *static contact* for thermal properties, *enclosure* and *contour following* for shape and size, etc. More exploratory procedures can be found in [2,9].

To realize various exploratory procedures, it is necessary to control the manipulators to produce the corresponding patterns of arm and hand movements. For this purpose, we create a set of motion primitives from which all the exploratory procedures can be generated. A motion primitive is an elementary control algorithm with a number of binding parameters. An exploratory procedure is obtained by assigning appropriate parameters to a single motion primitive and by combining multiple motion primitives.

For a single robot manipulator, the following four motion primitives are developed.

- Free motion: The manipulator moves from one location to another in the workspace. The binding parameters are the initial location, final location, type of trajectory (joint

or Cartesian space), and motion velocity. It is used to approach the object or environment. The manipulator is position controlled and forces are monitored. When approaching an object, the final location may not be known. In this case, a motion direction will replace the final location in the binding parameters. The manipulator will end its motion trajectory upon contacting the object.

- Surface-following: The manipulator moves along a surface. The binding parameters are the direction and velocity of motion, and amount of force exerted against the surface. This may be used, for example, to generate lateral motion for the identification of surface characteristics, such as texture and friction, or for contour following to determine the geometry. Hybrid position/force control is used in this motion primitive.

- Edge-following: The manipulator moves along an edge of the object. The velocity of motion and amount of force exerted against the edge are the binding parameters to be specified. Similar to the surface-following, hybrid position/force control is necessary.

- Pressing: The manipulator applies a specified force (pressure) against a surface. The binding parameters are the direction and amount of forces to be applied. This primitive is used to identify the hardness, stiffness or elasticity of the object or the environment.

The compliant wrist is used in implementation of these four motion primitives. An additional "T" shaped tool is used in the edge-following. The left-hand side segment of the horizontal part of the tool is connected to the end of the wrist and the right-hand side corner is used to contact the two sides of an edge as to exert forces in two directions. Experiments indicate that this simple device works reasonably well except on rough surfaces with large coefficient of friction.

For two cooperative manipulators, we have the following motion primitives which are especially designed for the identification of the parts relationship.

- Pushing/Pulling: *Pushing* is the two-arm version of *pressing* — the two manipulators apply force in the same line but opposite directions. The binding parameters for this primitive are the direction and amount of pushing forces. There are two possible cases for executing this primitive. The first one is that the two manipulators have grasped an object at two points and immediately start to apply the specified forces. At the same time, the motion displacements are monitored to determine the relative mobility of parts. The second one is that the two manipulators have no contacts with the object when issuing the *pushing* primitive. In this case, the two manipulators start a free motion primitive to bring the end effectors into contact with the object and then apply the specified forces. Pulling is the same as Pushing except that forces applied by each manipulator is reversed.

- Twisting: This primitive applies a moment to the object. The binding parameters to be specified are the twisting axis and amount of twisting moment. Its primary purpose is to reveal rotary mobility between two parts of the object.

- Bending: It is used to identify flexibility of a part and rotary mobility between parts. The binding parameters include the axis and amount of moment to be applied.

By using the real-time kernel Timix, we are able to send the position and force data of one manipulator to the other with a minimum delay of sampling period. This makes it possible to synchronize and coordinate two manipulators in real-time. Once again, the compliant wrist installed on one manipulator has its dual role of introducing passive compliance and measuring deflection. It is worth noticing that identifying the parts relationship does not necessarily require two manipulators. For instance, if a part is much smaller than the rest of an object, one manipulator may be able to identify the relationship of the part to the rest. But two manipulators are needed in most cases, especially when an object is made of parts of comparable size.

3.2 Force Control in Exploration

Robotic exploration requires that robot manipulators interact with the environment. Especially, to identify the parts relationship of an object, it is essential to apply forces to the object as to change the current state of parts. Force control plays an important role in robotic exploration. As a result, every motion primitive involves force control. The *free motion* primitive does not require force control, but forces are monitored to detect possible contacts with the environment. Due to the nature of exploratory tasks, force control used in exploration does not have any knowledge of the environment.

Most of force control methods developed early such as those described in [20,21] do not require a model of the environment. None of them however guarantees stability. Recent developments in force control methods [28,4,29] have resulted in control schemes that can be theoretically proven to be stable. On the other hand, these schemes assume *a priori* knowledge of the geometrical model of the environment.

In this experiment, force control is implemented with the aid of a compliant wrist developed in the GRASP laboratory [27]. The wrist is made of rubber which provides necessary compliance when the manipulator interacts with stiff environments. The wrist is also instrumented by potentiometers which measure deflection data of the wrist. The measured data is utilized in feedback control algorithms. This combination of passive compliance and active feedback is shown to give the best performance in realizing the two major motion primitives: *surface-following* and *edge-following*.

4 Conclusion

A two-arm experimental system for identifying movable and removable parts of an object was described. It is intended to be a subsystem of a complete exploratory robotic system which will eventually be able to identify physical parameters representing various properties of the environment. Motivated from psychological studies on haptic sensing, two cooperative manipulators are most suited for tasks of identifying relative mobility of the parts of an object. It is also proposed that the identification can be accomplished through a number of exploratory procedures. A set of motion primitives are implemented to generate all the exploratory procedures.

Synchronization of the two manipulators is achieved by integrating the real-time kernel Timix and Robot Control Interface (RCI). The replacement of UNIX by Timix provides predictable communication through the existing local area network. This loosely coupled approach allows us to easily add an additional processor to or delete any existing processor from the overall system.

5 Acknowledgement

The authors wish to thank Ruzena Bajcsy and Mario Campos for useful discussions. This work was in part supported by: Airforce grant AFOSR 88 0244, AfOSR 88-0966, Army/DAAG-29-84-K-0061, NSF-CER/DCR82-19196 Ao2, NSF CCR-8716975, NASA NAG5-1045, ONR SB-35923-0, NIH 1-RO1-NS-23636-01, NSF INT85-14199, ARPA N0014-88-K-0630, NATO grant No.0224/85, DuPont Corp., Sandia 75 1055, Post Office, IBM Corp. and LORD Corp.

References

[1] S. Arimoto, F. Miyazaki, and S. Kawamura. Cooperative motion of multiple robot arms or fingers. In *Proc. of 1987 IEEE International Conference on Robotics and Automation*, pages 1407–1412, March 1987.

[2] R. Bajcsy, S. Lederman, and R. L. Klatzky. *Machine Systems for Exploration and Manipulation: A Conceptual Framework and Method of Evalustion.* Technical Report MS-CIS-89-03, Department of Computer and Information Science, University of Pennsylvania, Jan. 1989.

[3] Y. Hu and A. A. Goldenberg. An adaptive approach to motion and force control of multiple coordinated robot arms. In *Proc. of 1989 IEEE International Conference on Robotics and Automation*, pages 1091–1096, May 1989.

[4] O. Khatib. A unified approach for motion and force control of robot manipulator: the operational space formulation. *Journal of Robotics and Automation*, RA-3(1):43–53, Feb. 1987.

[5] A. J. Koivo and G. A. Bekey. Report of workshop on coordinated multiple robot manipulators: planning, control, and application. *IEEE Journal of Robotics and Automation*, 4(1):91–93, Feb. 1988.

[6] K. Kosuge, M. Koga, K. Furuta, and K. Nosaki. Coordinated motion control of robot arm based on virtual internal model. In *Proc. of 1989 IEEE International Conference on Robotics and Automation*, pages 1097–1102, May 1989.

[7] K. Kreutz and A. Lokshin. Load balance and closed chain multiple arm control. In *Proc. of 1988 American Control Conference*, pages 2148–2155, June 1988.

[8] V. R. Kumar, X. Yun, and R. Bajcsy. Exploration of unknown mechanical assemblies through manipulation. In *Advances in Intelligent Robotics Systems Symposium, SPIE*, Nov. 1989.

[9] S. J. Lederman and R. L. Klatzky. Hand movements: a window into haptic object recognition. *Cognitive Psychology*, (19):342–368, 1987.

[10] I. Lee, R. King, and X. Yun. A real-time kernel for distributed multi-robot systems. In *Proc. of the 1988 American Control Conference*, pages 1083–1088, June 1988.

[11] I. Lee, R. B. King, and R. P. Paul. A predictable real-time kernel for distributed multi-sensor systems. *IEEE Computer*, 22(6):78–83, June 1989.

[12] Z. Li, P. Hsu, and S. Sastry. Dynamic coordination of a multiple robotic system with point contact. In *Proc. of 1988 American Control Conference*, pages 505–510, June 1988.

[13] J. Lloyd. *Implementation of a Robot Control Development Environment.* Master's thesis, Computer Vision and Robotics Laboratory, Department of Electrical Engineering, McGill University, 3480 University Street; Montreal, Quebec Canada H3A 2A7, Dec. 1985.

[14] J. Lloyd, M. Parker, and R. McClain. Extending the RCCL programming environment to multiple robots and processors. In *Proc. of 1988 IEEE International Conference on Robotics and Automation*, pages 465–469, Apr. 1988.

[15] A. A. Mangaser, Y. Wang, and S. E. Butner. Concurrent programming support for a multi-manipulator experiment on RIPS. In *Proc. of 1989 IEEE International Conference on Robotics and Automation*, pages 853–859, May 1989.

[16] N. H. McClamroch. Singular systems of differential equations as dynamic models for constrained robot systems. In *Proc. of 1986 IEEE International Conference on Robotics and Automation*, pages 21–28, 1986.

[17] N. H. McClamroch and D. Wang. Feedback stabilization and tracking of constrained robots. *IEEE Transactions on Automatic Control*, 33(5):419–426, May 1988.

[18] Y. Nakamura, K. Nagai, and T. Yoshikawa. Mechanics of coordinative manipulation by multiple robotic mechanisms. In *Proc. of 1987 IEEE International Conference on Robotics and Automation*, pages 991–998, March 1987.

[19] M. E. Pittelkau. Adaptive load-sharing force control for two-arm manipulators. In *Proc. of 1988 IEEE International Conference on Robotics and Automation*, pages 498–503, Apr. 1988.

[20] M. H. Raibert and J. J. Craig. Hybrid position/force control of manipulators. *Transactions of ASME, Journal of Dynamic Systems, Measurement, and Control*, 103(2):126–133, June 1981.

[21] J. K. Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *Proc. of 19th IEEE Conference on Decision and Control*, pages 95–100, Dec. 1980.

[22] K. Schwan, T. Bihari, B. W. Weide, and G. Taulbee. High-performance operating system primitives for robotics and real-time control systems. *ACM Transactions on Computer Systems*, 5(3):189–231, Aug. 1987.

[23] K. G. Shin and M. E. Epstein. Intertask communications in an integrated multirobot system. *IEEE Journal of Robotics and Automation*, RA-3(2):90–100, Apr. 1987.

[24] M. Vukobratovic and V. Potkonjak. *Dynamics of Manipulation Robots: Theory and Application*. Springer-Verlag, Berlin, New York, 1982.

[25] M. W. Walker, D. Kim, and J. Dionise. Adaptive coordinated motion control of two manipulator arms. In *Proc. of 1989 IEEE International Conference on Robotics and Automation*, pages 1084–1090, May 1989.

[26] B. H. Wilcox. An evolutionary strategy for telerobotic operation via geosynchronous relay. In *IEEE International Conference on Robotics and Automation, Workshop on Integration of AI with Robotics*, May 1989.

[27] Y. Xu and R. P. Paul. On position compensation and force control stability of a robot with a compliant wrist. In *Proc. of 1988 IEEE International Conference on Robotics and Automation*, pages 1173–1178, 1988.

[28] T. Yoshikawa. Dynamic hybrid position/force control of robot manipulators — description of hand constraints and calculation of joint driving force. *IEEE Journal of Robotics and Automation*, RA-3(5):386–392, Oct. 1987.

[29] X. Yun. Dynamic state feedback control of constrained robot manipulators. In *Proc. of the 27th IEEE Conference on Decision and Control*, pages 622–626, Dec. 1988.

[30] Y. F. Zheng and J. Y. S. Luh. Constrained relations between two coordinated industrial robots. In *Proc. of 1985 Conference of Intelligent Systems and Machines*, pages 118–123, Apr. 1985.