

A PC-Based Controller for Dexterous Arms

Paolo Fiorini Homayoun Seraji Mark Long

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109, USA

Abstract

This paper describes the architecture and performance of a PC-based controller for 7-DOF dexterous manipulators. The computing platform is a 486-based personal computer equipped with a bus extender to access the robot Multibus controller, together with a single board computer as the graphical engine, and with a parallel I/O board to interface with a force-torque sensor mounted on the manipulator wrist. The Windows™ environment is enhanced by the iRMX® real-time operating system that runs the position control algorithms for redundancy resolution and force control. The position control algorithm is executed every 2.5 ms and is based on the Configuration Control scheme; motions can be simulated and displayed in real-time by the graphical engine on a separate monitor. The results of several experiments carried out with a Robotics Research Corporation manipulator have shown position and force control capabilities comparable to those obtained with more extensive computing systems, thus validating the use of PC's for dexterous manipulator control.

1 Introduction

In October 1993, the National Aeronautics and Space Administration (NASA) initiated a program of Technology Cooperation Agreements (TCA) aimed at transferring some of the technologies developed at NASA centers to industrial partners. The Jet Propulsion Laboratory (JPL) developed a TCA with the Robotics Research Corporation (RRC) to transfer some of its robotics technology into RRC's new line of commercial products [7].

RRC is the leading manufacturer of dexterous manipulators and has recently enhanced its product line by introducing a new manipulator with improved mechanical design and with servo controlled electronics embedded into the arm [5]. In particular, this new design removes all joint servo loops from the system computer, thus freeing computing resources that

can be used to support an *open* control architecture and a graphical operator interface. JPL has a long track record of R&D in the areas of dexterous manipulator control and operator interfaces. Since 1986, researchers at JPL have developed a novel control scheme for dexterous manipulators, called Configuration Control [12-15], and this method has been used in many laboratories for RRC arm control. Furthermore graphical operator interfaces have been an active research at JPL for a decade [4]. In particular, graphical interfaces have been developed for command entry and data display [9], time delay compensation [1], and high-fidelity calibration [6].

This paper presents the results of the TCA task between JPL and RRC where a PC-based system is developed which integrates a controller for dexterous manipulators with a graphical simulation of the arm. The performance of this controller has been demonstrated in several position and contact experiments, and matches the performance of more powerful, multi-processor, VME based controllers.

The paper is organized as follows. In Section 2, we present the hardware used for the system, followed by the description of the software architecture in Section 3. In Section 4, we summarize the control algorithm, and present the main features of the graphical interface. Plots of position control experiments with a RRC arm are presented in Section 6. Finally, in Section 7 we draw some conclusions from this work and discuss the directions of future research and development.

2 Hardware Description

In this section, we describe the hardware of the PC-based system designed to support the TCA task between JPL and RRC. The complete system consists of a Robotics Research Corporation's model K1207 7-DOF arm/control unit, a 486-based personal com-



Figure 1: The PC-based system

puter with two monitors, and an Assurance Technology Inc. (ATI) 6-axis Force-Torque (FT) sensor, as shown in Figure 1. The RRC manipulator used in this task is equipped with a Multibus-based servo control unit located in a separate cabinet. Therefore, the PC is connected to the Multibus via a shared memory interface using a two-card PC-to-Multibus bus extender from the BIT3 Corporation. This adapter allows a high speed bi-directional shared memory interface between the two buses by mapping the memory locations used by the Multibus directly into the memory space of the PC.

The workstation is equipped with a 486DX2 66 MHz Intel microprocessor, 11 MBytes of RAM, and serial and parallel input/output ports for data exchange with the FT sensor and the graphical processor. The unusual choice of memory space is a trade-off between the memory space addressable by the bus extender and the 1-MB minimum RAM increment allowed by the PC hardware. Furthermore, because of the limitations imposed by the CPU speed and by the real-time software, we decided to implement the graphical part of the operator interface on a separate single-board computer displaying the simulation on a dedicated color monitor. The Interlogic 486 DX4 100 MHz Single Board Computer (SBC) is chosen to allow the installation of the graphical processor in the controller at a later time. The SBC is configured with 16 MB of RAM and 1 MB of local video memory. The communication between the PC

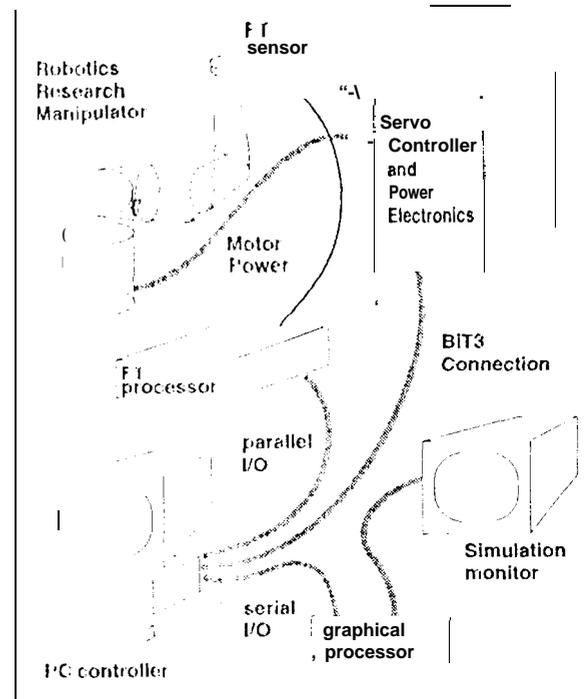


Figure 2: Schematic layout of the PC controller.

controller and the SBC is carried out via a RS232 serial port. The system layout is shown schematically in Figure 2.

3 Software Description

Because of RRC's large investment in PL/M software developed for its current product line, it was decided to adopt the Intel iRMX[®] operating system, since it supports PL/M. The iRMX[®] is a real-time multi-tasking environment, where Windows[™] and DOS can execute as independent tasks. The iRMX[®] memory management requires the CPU in *protected mode* and therefore Windows[™] can only run in *standard mode*, with a *reduced memory space* and limited display capabilities. Thus the need for a separate graphical processor, since high quality display requires Windows operating in *enhanced mode*.

This limitation, together with RRC's plan to rewrite their software in C, convinced us to use only few features of iRMX[®], and develop a software architecture that can easily be ported to a different real-time environment. Therefore, we use the programmable interval-counter of the PC CMOS memory chip as the real-time clock of the control loop. This timer produces a hardware interrupt that dots

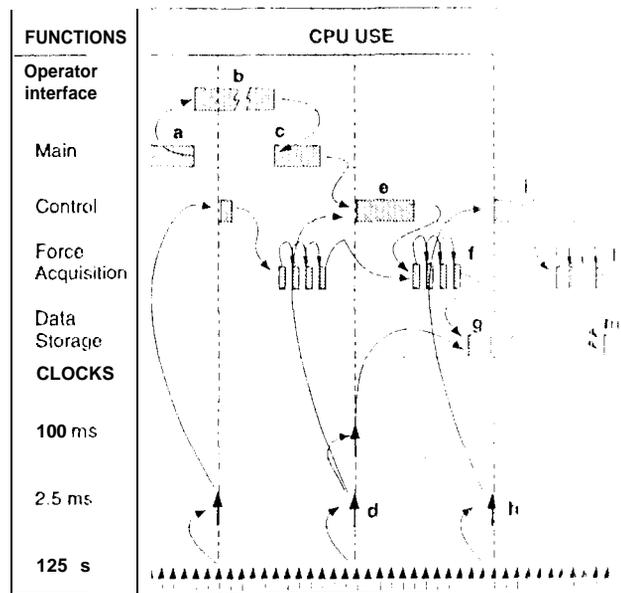


Figure 3: Diagram of the software timing.

not interfere with the PC clock, nor does it incur any iRMX[®] overhead. At an interval of $125 \mu\text{s}$ is chosen to ensure a good stability within the 2.5 ms position control cycle and a prompt acquisition of the FT data.

The controller software consists of the following tasks:

- (i) The *main* task handles housekeeping functions and the message dispatching to the other tasks.
- (ii) The *operator interface* task handles the command menus and displays the trajectory parameters. This task is data driven, being activated by the position data generated at the end of a trajectory. The operator is presented with a series of menus, one for (i) control mode of the robot: joint or Cartesian position control, force or compliance control. Furthermore, the operator can choose to record position, orientation or FT data.
- (iii) The *control* task performs the trajectory generation and kinematic computations, and is activated directly by the hardware interrupt. This task takes approximately 2 ms to complete, faster than the minimum rate of 400 Hz at which the servo controller of the manipulator can accept new set-points. The task performs all the necessary hardware handshakes with the servo controllers and does a final check on the position and velocity set-points before sending them to the servo controllers. Sensor acquisition is also a part of the control task, since FT measurements must be synchronized with the control algorithm. Every $100 \mu\text{s}$, the position and velocity of the arm, and the FT val-

ues, are saved into memory. At the end of the motion, these data are stored in an ASCII file for later display.

The interaction among the three tasks is schematically represented in Figure 3. The tasks are mutually exclusive, except for the FT acquisition, which is always active. Figure 3 shows also the timing of the control functions. The real-time clock runs continuously at a $125 \mu\text{s}$ interval, and increments the counters for the 2.5 ms control algorithm and the 100 ms data recording. In Figure 3, the execution flow of the program is shown by the arrows linking the various tasks and functions, and representing causal relations. The controller software starts with the *main* task (a), followed by the operator command entry (b) and by the message dispatch (c). A new motion is then started in the following 2.5 ms interval (d) by the *control* task (e), that decodes the command message, computes the parameters for the trajectory interpolation, and starts the data recording (f). The FT acquisition (g) is always active and starts every two control cycles. The recording function is spread over several 2.5 ms intervals to avoid interfering with the trajectory generation. In the following 2.5 ms interval (h), the *control* task computes the joint set-points (i), continues the FT acquisition (l), and continues the data recording (m). At the end of the motion, the *operator* task displays the current position of the arm, and the trajectory data are copied to disk.

The FT sensor processor acquires the eight strain-gauge values in approximately 2 ms , and transmits them to the PC at a rate of $250 \mu\text{s}$. The acquisition/transmission cycle thus provides new FT data every two cycles of the control algorithm.

The control task implements the Configuration Control approach for motion control of dexterous manipulators. The operator interface includes a graphical simulation of the RRC manipulator, thus enabling off-line motion display and verification. These aspects are discussed in the next two sections.

4 Arm Control System

Advanced control of manipulators has always been an active area of research at JPL. In particular, JPL has developed a class of motion control algorithms for redundant manipulators called Configuration Control (CC) [2, 12-15], where the user can specify task-dependent constraints for the motion that utilize the robot redundancy and allow efficient end-effector trajectory control. This approach was implemented originally for RRC manipulators controlled by VME-based multiprocessors [10], and the resulting algorithms

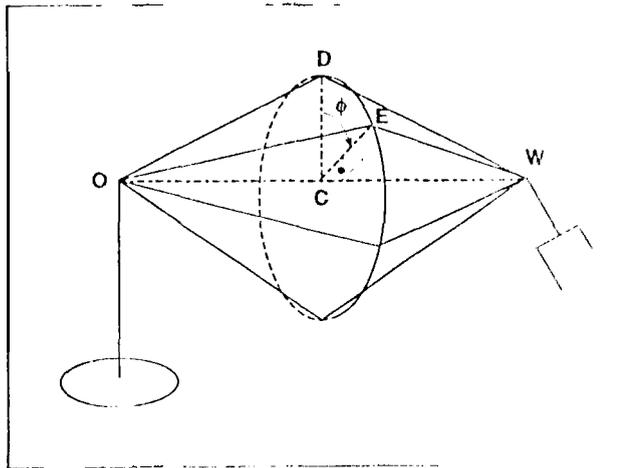


Figure 4: Definition of the arm angle ϕ .

were extensively tested in several tasks

4.1 Configuration Control

Configuration Control allows specification of additional tasks for redundancy resolution. In this implementation, we have chosen the seventh task to be the control of the arm angle for elbow placement. The arm angle ϕ is defined as the angle between the arm plane OEW and the vertical plane ODW passing through the line OW, where O, E and W refer to the origin of the shoulder, elbow and wrist frames, respectively [8], as shown in Figure 4.

The major elements of the CC approach are: the generation of the end-effector and arm angle trajectories, the computation of the forward kinematics and augmented Jacobian, and the computation of the inverse kinematics.

The trajectory generator produces smooth continuous cycloidal functions to make the transition from the initial value to the final value in the specified time.

The computations of the forward kinematics and augmented Jacobian of the 7-DOF manipulator utilize Craig's interpretation of Denavit-Hartenberg (DH) parameters for frame assignment [3, 8]. This method provides direct computation of the manipulator Jacobian in the world frame of the robot. The computation of the inverse kinematics is implemented by a damped-least-squares method [16, 11, 14], which trades large joint velocities against large task-space errors. The resultant computation of the joint velocities has the following form [14]:

$$\dot{\theta}_d = [J^T W_t J + W_v]^{-1} J^T W_t [\dot{X}_d + KJ] \quad (1)$$

where W_t and W_v are the task-space error weights and joint velocity damping weights, $E = X_d - X$ and K is a diagonal matrix with positive elements. Cholesky decomposition is used to find $\dot{\theta}_d$. It can be seen that as the Jacobian becomes singular, the velocity weight dominates in the inverse matrix term in (1), reducing the commanded joint velocities. The reduced joint velocities, in turn, act to retard the arm from reaching the singular configuration.

During the trajectory execution, the control task sends the measured joint angles to the graphical processor. This ensures that the user views the actual arm configuration. The control task computes also the joint angles for the trajectory simulation, by sending the joint set-points to the graphical engine only. The implementation of the simulation mode in the control task assures that the simulation will effectively duplicate the real trajectory since the same code is executed in both cases.

4.2 Contact Control

The contact control schemes operate in the user-defined tool frame where the task is defined and executed. The dynamic modeling and stability analysis of the proposed position-based contact control systems are studied in [13].

4.2.1 Compliance Control

A manipulator under Cartesian position control in contact with a reaction surface can be adequately described in each Cartesian direction of contact by the second-order transfer-function [13]

$$\frac{F(s)}{X_r(s)} = \frac{ck_c}{s^2 + as + b} \quad (2)$$

where k_c is the surface stiffness, $[a, b, c]$ are the manipulator constants, and the origin of the frame-of-reference is the point of contact between the end-effector and the reaction surface with the x-axis in the direction of the applied force.

The basic concept of the compliance control system is to use force feedback in order to reduce the forward gain or stiffness $k_f = \frac{c}{b}$ in the end-effector/reaction surface interaction. This will then allow the reference position X_r to be used as a command input to control the contact force F as the output. Under compliance control, the reference trajectory is used differently for the two categories of tasks: as the desired motion trajectory in unconstrained tasks, and as the input command to control the contact force in constrained tasks. This is in contrast to force control schemes in which a

force setpoint is used to explicitly command the contact force.

The compliance controller used in this paper consists of the first-order lag $k_1/(\tau s + 1)$ in parallel with the feed-forward gain k_2 , where $\{k_1, k_2, \tau\}$ are constant parameters. This yields the compensator transfer function

$$K(s) = \frac{k_1}{\tau s + 1} + k_2 = \frac{\alpha s + \beta}{\tau s + 1} \quad (3)$$

where $\alpha = \tau k_2$ and $\beta = k_1 + k_2$.

Now, the steady-state relationship between the reference position X_r and the contact force F is obtained as:

$$F = \frac{ck_c}{b + ck_c\beta} X_r = \frac{1}{k_f^{-1} + \beta} X_r = k_{ap} X_r \quad (4)$$

It is seen that under compliance control, the reaction surface *appears* as a spring with the apparent stiffness $k_{ap} = [k_f^{-1} + \beta]^{-1}$. This is the equivalent stiffness of the series combination of the two springs k_f and β representing the stiffnesses of the open-loop system and the compliance controller, respectively.

4.2.2 Force Control

In contrast to compliance control which is an *implicit* force control scheme, force control is an *explicit* scheme in which the force setpoint F_r is used as the command input to control the **contact**, force F . The controller $K(s)$ now uses the force error information $e = F_r - F$ to generate the necessary control action so that the contact force F tracks the force setpoint F_r .

The force controller used in this system is a proportional-plus-integral (PI) controller:

$$X_f(t) = k_p e(t) + k_i \int_0^t e(t) dt \quad (5)$$

where k_p and k_i are the constant proportional and integral force feedback gains, respectively, and X_f is the position perturbation produced by the force controller as shown in Figure 5.

The contact force tracks the force setpoint accurately with zero steady-state error. Furthermore the system is robust to parameter variations so that the steady-state setpoint tracking and disturbance rejection characteristics are preserved, provided that the closed-loop system remains stable.

5 Operator Interface

The manipulator control interface enables the user to specify the command and control modes of operation,

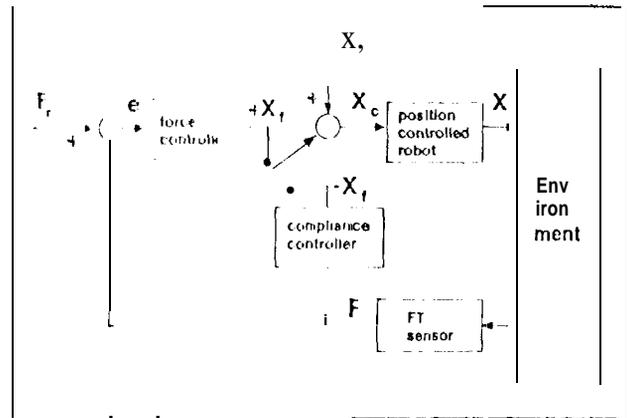


Figure 5: Block diagram of the contact control scheme.

set (control system parameters, and display the manipulator status.

5.1 Command Interface

The user first selects the command mode by entering the appropriate command. The following choices are available:

- (i) *Joint Control Mode*: Commands are issued to the seven joint angles of the arm.
- (ii) *Cartesian-World Control Mode*: Commands are expressed relative to a fixed user-defined frame of reference (the world frame).
- (iii) *Cartesian-World Relative Control Mode*: Motion commands are expressed relative to coordinate frame of the hand.
- (iv) *Explicit Force Control Mode*: Force commands are expressed relative to the coordinate frame fixed to the manipulator hand.
- (v) *Implicit Force (Compliance) Control Mode*: Compliance controller parameters are specified which set the apparent stiffness k_{ap} of the hand.

The various command modes provide considerable flexibility for operating the robotic arm. The control mode can be changed on-line by the user at the end of each motion.

Data recorded during the motion is stored and can be displayed in a window on the operator interface monitor of the PC controller.

5.2 Graphical Simulation

Full three-dimensional graphical display of the robot is an indispensable tool for system development and operation. Developers benefit from kinematic testing on a graphical simulation by saving time, resources

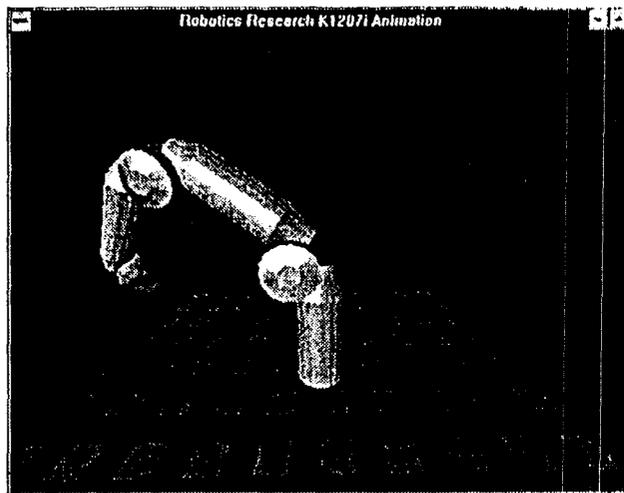


Figure G: Graphic simulation of the RRC manipulator

and potential damage to the real robot. Operators can accurately preview and test motion trajectories.

The use of the SBC as a graphical processor made the choice of the graphic package independent from the operating system used in the controller. Since much of JPL 3-D animation of the RRC manipulators have been developed using IRIS-GL on Silicon Graphics computers, it was decided to develop the simulation using the emerging graphics standard *OpenGL* compatible with IRIS-GL, and to use Windows NT as the operating system of the SBC, since it supports OpenGL.

Figure 6 is a screen dump of the graphical display, and shows the solid model of the arm, with hidden surface removal, shading, surface reflective properties, and user-deflected lighting conditions. The user can select the orientation and distance of the viewpoint and switch between drawing modes, by using pull-down menus, and by click-and-drag with the mouse over the simulation display.

The animation currently runs at 5 Hz rate, and is driven by the seven values of the manipulator joint angles, sent by the Controller via the RS232 communication link.

6 Experimental Results

In this section, we present the results of experiments in position control of the dexterous manipulator, aimed at demonstrating the quality of the performance achievable with a PC-based controller.

In these experiments, the arm is operated in Cartesian-world control mode using the cycloidal tra-

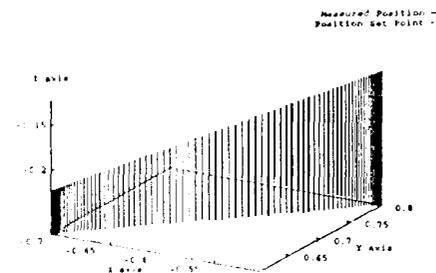


Figure 7: Plot of a position control experiment.

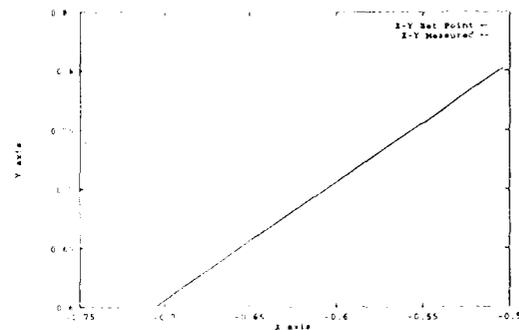


Figure 8: Plot of the trajectory on the XY plane.

jectory generator. An example of motion is shown in Figure 7, where the arm moved 200 mm in the X and Y directions, and 100 mm in the Z direction. The initial position was ($x = -704.12 \text{ mm}$, $y = 601.86 \text{ mm}$, $z = -223.34 \text{ mm}$), the initial orientation was ($roll = 179.60^\circ$, $pitch = 0.84^\circ$, $yaw = -139.130$), and the initial arm angle was $\phi = -0.23^\circ$. The arm reached the terminal position ($x = -504.08 \text{ mm}$, $y = 801.88 \text{ mm}$, $z = -123.31 \text{ mm}$), with orientation ($roll = 179.60^\circ$, $pitch = 0.85^\circ$, $yaw = -139.13^\circ$), and arm angle $\phi = -0.23^\circ$. The projection of the trajectory on the XY plane is shown in Figure 8. The maximum error is comparable with other motions with the arm starting at different postures.

These results are similar to those obtained with the same manipulator controlled by a VME system [10] and to RRC's own manufacturing specifications, thus indicating no performance degradation due to the use of a PC-based controller.

7 Conclusions

A PC-based controller for 7-DOF manipulators is described in this paper. The manipulator is well suited for tasks that demand dexterous positioning of the end-effector, and the controller fully exploits the arm capabilities by controlling the end-effector location and the arm posture simultaneously. This reports a general-purpose manipulator system that possesses generic capabilities usable in many different applications.

The proposed system responds to the needs of the robotic community for a PC-based high-performance controller for dexterous manipulators that can support an open architecture for integrating and trying different control algorithms, and includes a realistic graphical simulation for test and verification of motion trajectories.

The experiments carried out at JPL, have shown that a simple PC-based controller is able to handle successfully the control of a complex dextrous arm. Furthermore, the implementation of the controller software has shown that robot control can take a significant advantage from the large software base of PC-based computers, that many functions such as operator interfaces and data display, can now be designed and implemented rapidly, inexpensively and with no system performance degradation.

In the future, we plan to continue further development of this controller, in particular in the area of graphical user interface, and real-time architecture. The goals of this development are to eliminate any textual input to the controller, and to make a better use of the interrupt capabilities of the PC to achieve a higher level of task parallelism.

8 Acknowledgment

The research described in this paper has been partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautic and Space Administration.

References

J.A.K. Bejczy, W.S. Kim, and S. Venura. The path to a mobile: Predictive display for teleoperator with time delay. In *IEEE International Conference on Robotics and Automation*, pages 646-650. Cincinnati, OH, May 1990.

- [2] R. Colbaugh, H. Seraji, and K. Glass. Obstacle avoidance for redundant robots using configuration control. *Journal of Robotic Systems*, 6(6):721-744, 1989.
- [3] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison Wesley, New York, NY, 1986.
- [4] P. Fiorini, A.K. Bejczy, and P.S. Schenker. Integrated interface for advanced teleoperation. *IEEE Control Systems Magazine*, 13(1): 20, October 1993.
- [5] J. Karlen. Proceedings of the HRC Users Group Meeting. Technical report, RoboticsResearch Corporation, Anaheim, OH, 1993.
- [6] W.S. Kim and A.K. Bejczy. Demonstration of a high-fidelity predictive/preview display technique for teleoperatorics servicing in space. *IEEE Transactions on Robotics and Automation*, pages 698-702, October 1993.
- [7] W.S. Kim et al. Commercialization of jpl virtual reality calibration and redundant manipulator control technology. In *Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space (ISAIRAS'94)*. Pasadena, CA, September 1994.
- [8] K. Krentz-DeGard, M. Long, and H. Seraji. Kinematic analysis of 7-dof manipulators. *International Journal of Robotics Research*, 11(5):469-481, October 1992.
- [9] P. Lee, H. Hammarlund, and L. Wood. Telerobotic configuration control. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 121-126, Los Angeles, CA, 1990.
- [10] D. Jain, T. Lee, and H. Seraji. A real-time control system for a mobile redundant 7-DOF arm. In *IEEE International Conference on Robotics and Automation*, pages 1188-1193, San Diego, CA, 1994.
- [11] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME Journal on Dynamic Systems, Measurement and Control*, (108):163-171, 1986.
- [12] H. Seraji. Configuration control of redundant manipulators: Theory and implementation. *IEEE Transactions on Robotics and Automation*, 5(4):472-490, 1989.
- [13] H. Seraji. Task-based configuration control of redundant robots. *Journal of Robotic Systems*, 9(3):411-451, 1992.
- [14] H. Seraji and R. Colbaugh. Improved configuration control for redundant robots. *Journal of Robotic Systems*, 7(6):897-928, 1990.
- [15] H. Seraji, M. Long, and Lee T. Motion control of 7-dof arms: The configuration control approach. *IEEE Transactions on Robotics and Automation*, 9(2):125-139, 1993.

- [16] C.W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and least-square methods. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16:93-101, 1986.