Proceedings of the 1996 IEEE
International Conference on Robotics and Automation.
Minneapolis, Minnesota - April 1996.

A Rapidly Deployable Manipulator System

Christiaan J.J. Paredis, H. Benjamin Brown, Pradeep K. Khosla

Department of Electrical and Computer Engineering and
The Robotics Institute,
Carnegie Mellon University,
Pittsburgh, PA 15213

Abstract:

A rapidly deployable manipulator system combines the flexibility of reconfigurable modular hardware with modular programming tools, allowing the user to rapidly create a manipulator which is custom-tailored for a given task. This article describes two main aspects of such a system, namely, the Reconfigurable Modular Manipulator System (RMMS) hardware and the corresponding control software.

1 Introduction

Robot manipulators can be easily reprogrammed to perform different tasks, yet the range of tasks that can be performed by a manipulator is limited by its mechanical structure. For example, a manipulator well-suited for precise movement across the top of a table would probably not be capable of lifting heavy objects in the vertical direction. Therefore, to perform a given task, one needs to choose a manipulator with an appropriate mechanical structure. This is only possible when the task requirements are known beforehand, which is often not the case in unpredictable and changing environments, such as space stations or nuclear facilities.

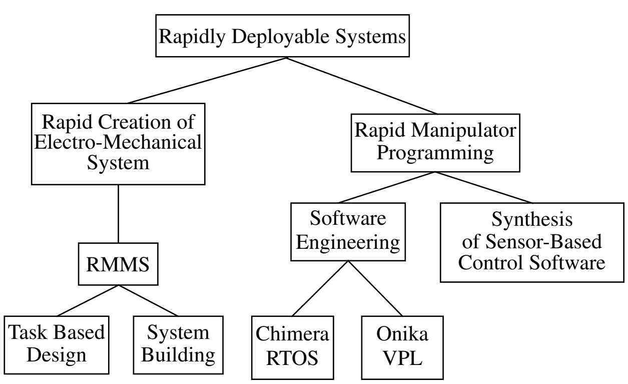

Figure 1: The concept of Rapidly Deployable Systems.

We propose the concept of a *rapidly deployable manipulator system* to address the above mentioned shortcomings of fixed configuration manipulators. As is illustrated in Figure 1, a rapidly deployable manipulator system consists of software and hardware that allow the user to rapidly build and program a manipulator which is custom-tailored for a given task.

The central building block of a rapidly deployable system is a *Reconfigurable Modular Manipulator System* (RMMS). The RMMS utilizes a stock of interchangeable link and joint modules of various sizes and performance specifications. One such module is shown in Figure 2. By combining these general purpose modules, a wide range of special purpose manipulators can be assembled. Recently, there has been considerable interest in the idea of modular manipulators [2, 4, 5, 7, 9, 10, 14], for research applications as well as for industrial applications. However, most of these systems lack the property of reconfigurability, which is key to the concept of rapidly deployable systems. The RMMS is particularly easy to reconfigure thanks to its integrated quick-coupling connectors described in Section 3.

Figure 2: An RMMS pivot joint module.

Effective use of the RMMS requires, *Task Based Design* software. This software takes as input descriptions of the task and of the available manipulator modules; it generates as output a modular assembly configuration optimally suited to perform the given task. Several different approaches have been used successfully to solve simplified instances of this complicated problem [3, 8, 12, 13].

A third important building block of a rapidly deployable manipulator system is a framework for the generation of control software. To reduce the complexity of software generation for real-time sensor-based control systems, a software paradigm called *software assembly* has been proposed in the Advanced Manipulators Laboratory at CMU. This paradigm combines the concept of reusable and reconfigurable software components, as is supported by the Chimera real-time operating system [15], with a graphical user interface and a visual programming language, implemented in Onika [6].

Although the software assembly paradigm provides the software infrastructure for rapidly programming manipulator systems, it does not solve the programming problem itself. Explicit programming of sensor-based manipulator systems is cumbersome due to the extensive amount of detail which must be specified for the robot to perform the task. The software synthesis problem for sensor-based robots can be simplified dramatically, by providing robust *robotic skills*, that is, encapsulated strategies for accomplishing common tasks in the robots task domain [11]. Such robotic skills can then be used at the task level planning stage without having to consider any of the low-level details.

As an example of the use of a rapidly deployable system, consider a manipulator in a nuclear environment where it must inspect material and space for radioactive contamination, or assemble and repair equipment. In such an environment, widely varied kinematic (e.g., workspace) and dynamic (e.g., speed, payload) performance is required, and these requirements may not be known a priori. Instead of preparing a large set of different manipulators to accomplish these tasks—an expensive solution—one can use a rapidly deployable manipulator system. Consider the following scenario: as soon as a specific task is identified, the task based design software determines the optimal modular assembly configuration to perform the task. This optimal configuration is then assembled from the RMMS modules by a human or, in the future, possibly by another manipulator. The resulting manipulator is rapidly programmed by using the software assembly paradigm and our library of robotic skills. Finally, the manipulator is deployed to perform its task.

Although such a scenario is still futuristic, the development of the reconfigurable modular manipulator system, described in this paper, is a major step forward towards our goal of a rapidly deployable manipulator system.

Our approach could form the basis for the next generation of autonomous manipulators, in which the traditional notion of sensor-based autonomy is extended to *configuration-based autonomy*. Indeed, although a deployed system can have all the sensory and planning information it needs, it may still not be able to accomplish its task because the task is beyond the system's physical capabilities. A rapidly deployable system, on the other hand, could adapt its physical capabilities based on task specifications and, with advanced sensing, control, and planning strategies, accomplish the task autonomously.

2 Design of self-contained hardware modules

In most industrial manipulators, the controller is a separate unit housing the sensor interfaces, power amplifiers, and control processors for all the joints of the manipulator. A large number of wires is necessary to connect this control unit with the sensors, actuators and brakes located in each of the joints of the manipulator. The large number of electrical connections and the non-extensible nature of such a system layout make it infeasible for modular manipulators. The solution we propose is to distribute the control hardware to each individual module of the manipulator. These modules then become self-contained units which include sensors, an actuator, a brake, a transmission, a sensor interface, a motor amplifier, and a communication interface, as is illustrated in Figure 3. As a result, only six wires are required for power distribution and data communication.

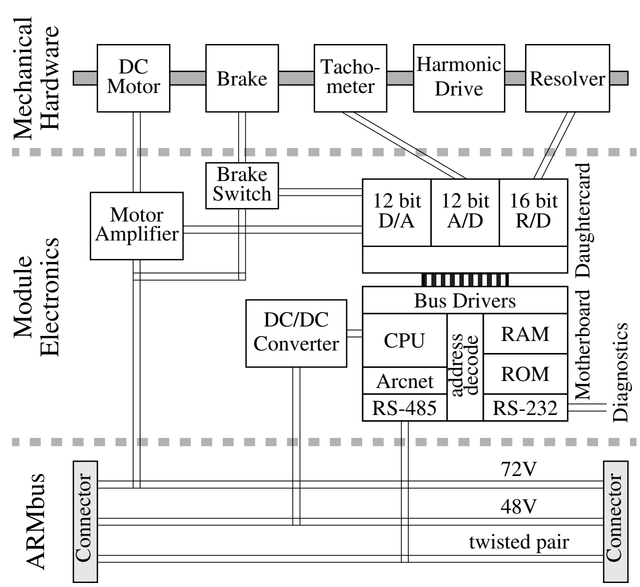

Figure 3: Diagram of a self-contained RMMS module.

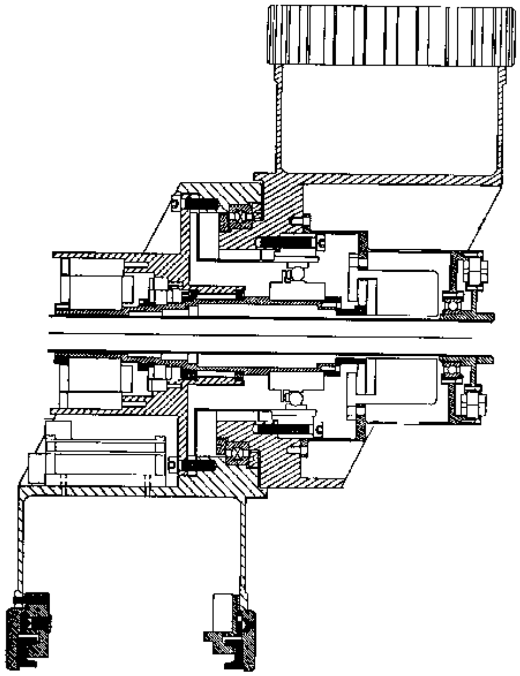

Figure 4: Assembly drawing of a pivot joint module.

2.1 Mechanical design

The goal of the RMMS project is to have a wide variety of hardware modules available. So far, we have built four kinds of modules: the manipulator base, a link module, three pivot joint modules (one of which is shown in Figure 2), and one rotate joint module. The base module and the link module have no degrees-of-freedom; the joint modules have one degree-of-freedom each. The mechanical design of the joint modules compactly fits a DC-motor, a fail-safe brake, a tachometer, a harmonic drive and a resolver.

The pivot and rotate joint modules use different outside housings to provide the right-angle or in-line configuration respectively, but are identical internally. Figure 4 shows in cross-section the internal structure of a pivot joint. Each joint module includes a DC torque motor and 100:1 harmonic-drive speed reducer, and is rated at a maximum speed of 1.5rad/s and maximum torque of 270Nm. Each module has a mass of approximately 10.7kg. A single, compact, X-type bearing connects the two joint halves and provides the needed overturning rigidity. A hollow motor shaft passes through all the rotary components, and provides a channel for passage of cabling with minimal flexing.

2.2 Electronic design

The custom-designed on-board electronics are also designed according to the principle of modularity. Each RMMS module contains a motherboard which provides the basic functionality and onto which daughtercards can be stacked to add module specific functionality.

The motherboard consists of a Siemens 80C166 microcontroller, 64K of ROM, 64K of RAM, an SMC COM20020 universal local area network controller with an RS-485 driver, and an RS-232 driver. The function of the motherboard is to establish communication with the host interface via an RS-485 bus and to perform the low-level control of the module, as is explained in more detail in Section 4. The RS-232 serial bus driver allows for simple diagnostics and software prototyping.

A stacking connector permits the addition of an indefinite number of daughtercards with various functions, such as sensor interfaces, motor controllers, RAM expansion etc. In our current implementation, only modules with actuators include a daughtercard. This card contains a 16 bit resolver to digital converter, a 12 bit A/D converter to interface with the tachometer, and a 12 bit D/A converter to control the motor amplifier; we have used an of-the-shelf motor amplifier (Galil Motion Control model SSA-8/80) to drive the DC-motor. For modules with more than one degree-of-freedom, for instance a wrist module, more than one such daughtercard can be stacked onto the same motherboard.

3 Integrated quick-coupling connectors

To make a modular manipulator be reconfigurable, it is necessary that the modules can be easily connected with each other. We have developed a quick-coupling mechanism with which a secure mechanical connection between modules can be achieved by simply turning a ring hand-tight; no tools are required. As shown in Figure 5, keyed flanges provide precise registration of the two modules. Turning of the locking collar on the male end produces two distinct motions: first the fingers of the locking ring rotate (with the collar) about 22.5 degrees and capture the fingers on the flanges; second, the collar rotates relative to the locking ring, while a cam mechanism forces the fingers inward to securely grip the mating flanges. A ball-

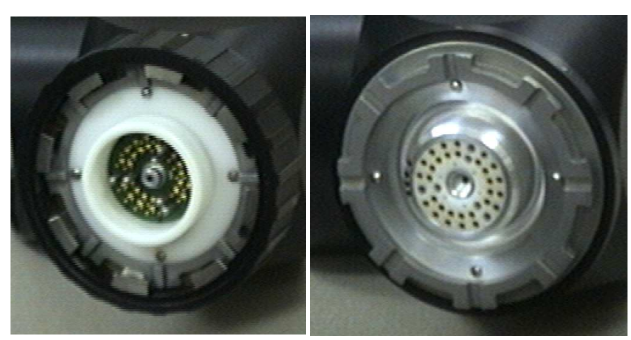

Figure 5: A male and female RMMS connector.

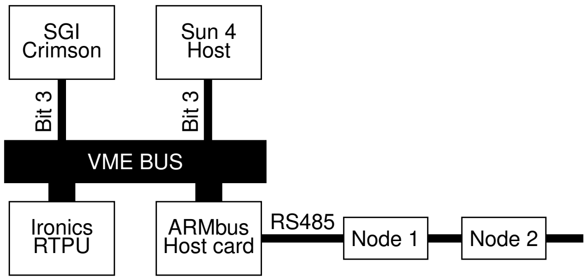

Figure 6: RMMS computing hardware.

transfer mechanism between the collar and locking ring automatically produces this sequence of motions.

At the same time the mechanical connection is made, pneumatic and electronic connections are also established. Inside the locking ring is a modular connector that has 30 male electrical pins plus a pneumatic coupler in the middle. These correspond to matching female components on the mating connector. Sets of pins are wired in parallel to carry the 72V-25A power for motors and brakes, and 48V–6A power for the electronics. Additional pins carry signals for two RS-485 serial communication busses and four video busses. A plastic guide collar plus six alignment pins prevent damage to the connector pins and assure proper alignment. The plastic block holding the female pins can rotate in the housing to accommodate the eight different possible connection orientations (8@45 degrees). The relative orientation is automatically registered by means of an infrared LED in the female connector and eight photodetectors in the male connector.

4 ARMbus communication system

Each of the modules of the RMMS communicates with a VME-based host interface over a local area network called the ARMbus; each module is a node of the network. The communication is done in a serial fashion over an RS-485 bus which runs through the length of the manipulator. We use the ARCNET protocol [1] implemented on a dedicated IC (SMC COM20020). ARCNET is a deterministic token-passing network scheme which avoids network collisions and guarantees each node its time to access the network. Blocks of information called packets may be sent from any node on the network to any one of the other nodes, or to all nodes simultaneously (broadcast). Each node may send one packet each time it gets the token. The maximum network throughput is 5Mb/s.

The first node of the network resides on the host interface card, as is depicted in Figure 6. In addition to a VME address decoder, this card contains essentially the same hardware one can find on a module motherboard. The communication between the VME side of the card and the ARCNET side occurs through dual-port RAM.

There are two kinds of data passed over the local area network. During the manipulator initialization phase, the modules connect to the network one by one, starting at the base and ending at the end-effector. On joining the network, each module sends a data-packet to the host interface containing its serial number and its relative orientation with respect to the previous module. This information allows us to automatically determine the current manipulator configuration.

During the operation phase, the host interface communicates with each of the nodes at 400Hz. The data that is exchanged depends on the control mode—centralized or distributed. In centralized control mode, the torques for all the joints are computed on the VME-based real-time processing unit (RTPU), assembled into a data-packet by the microcontroller on the host interface card and broadcast over the ARMbus to all the nodes of the network. Each node extracts its torque value from the packet and replies by sending a data-packet containing the resolver and tachometer readings. In distributed control mode, on the other hand, the host computer broadcasts the desired joint values and feed-forward torques. Locally, in each module, the control loop can then be closed at a frequency much higher than 400Hz. The modules still send sensor readings back to the host interface to be used in the computation of the subsequent feed-forward torque.

5 Modular and reconfigurable control software

The control software for the RMMS has been developed using the Chimera real-time operating system, which supports reconfigurable and reusable software components [15]. The software components used to control the RMMS are listed in Table 1. The *trjjline, dls,* and *grav_comp* components require the knowledge of certain configuration dependent parameters of the RMMS, such as the number of degrees-of-freedom, the Denavit-Hartenberg parameters etc. During the initialization phase, the RMMS interface establishes contact with each of the hardware modules to determine automatically which modules are being used and in which order and orientation they have been assembled. For each module, a data file with a parametric model is read. By combining this information for all the modules, kinematic and dynamic models of the entire manipulator are built.

After the initialization, the rmms software component operates in a distributed control mode in which the microcontrollers of each of the RMMS modules perform PID control locally at 1900Hz. The communication between the modules and the host interface is at 400Hz, which can differ from the cycle frequency of the *rmms* software component. Since we use a triple buffer mechanism [16]

name	function	description
rmms	RMMS interface	Initializes RMMS; Computes and sends raw joint positions to the joint modules; Receives and converts raw sensor readings.
grav_comp	gravity compensation	Computes the gravity compensation torques based on the kinematic and dynamic manipulator models.
fkjac	forward kinematics and Jacobian	Computes the forward kinematics and Jacobian, based on the kinematic manipulator model.
dls	damped least-squares kinematic controller	Computes the desired joint space position according to the damped least-squares kinematic control algorithm.
trjjline	joint trajectory generator	Generates a linearly interpolated joint space trajectory.
trjcgen	Cartesian trajectory generator	Generates a linearly interpolated Cartesian space trajectory.

Table 1: Chimera software module description.

for the communication through the dual-port RAM on the ARMbus host interface, no synchronization or handshaking is necessary.

Because closed form inverse kinematics do not exist for all possible RMMS configurations, we use a damped least-squares kinematic controller to do the inverse kinematics computation numerically..

6 Seamless integration of simulation

To assist the user in evaluating whether an RMMS configuration can successfully complete a given task, we have built a simulator. The simulator is based on the TeleGrip robot simulation software from Deneb Inc., and runs on an SGI Crimson which is connected with the real-time processing unit through a Bit3 VME-to-VME adaptor, as is shown in Figure 6. A graphical user interface allows the user to assemble simulated RMMS configurations very much like assembling the real hardware. Completed configurations can be tested and programmed using the TeleGrip functions for robot devices. The configurations can also be interfaced with the Chimera real-time software running on the same RTPUs used to control the actual hardware. As a result, it is possible to evaluate not only the movements of the manipulator but also the real-time CPU usage and load balancing. Figure 7 shows an RMMS simulation compared with the actual task execution.

7 Summary

We have developed a Reconfigurable Modular Manipulator System which currently consists of six hardware modules, with a total of four degrees-of-freedom. These modules can be assembled in a large number of different configurations to tailor the kinematic and dynamic properties of the manipulator to the task at hand. The control software for the RMMS automatically adapts to the assembly configuration by building kinematic and dynamic models of the manipulator; this is totally transparent to the user. To assist the user in evaluating whether a manipulator configuration is well suited for a given task, we have also built a simulator.

Acknowledgment

This research was funded in part by DOE under grant DE-F902-89ER14042, by Sandia National Laboratories under contract AL-3020, by the Department of Electrical and Computer Engineering, and by The Robotics Institute, Carnegie Mellon University.

The authors would also like to thank Randy Casciola, Mark DeLouis, Eric Hoffman, and Jim Moody for their valuable contributions to the design of the RMMS system.

References

[1] ARCNET Trade Association, *ANSI/ATA 878.1—Local Area Network: Token Bus (version 1.10)*

[2] B. Benhabib and M. Q. Dai, “Mechanical Design of a Modular Robot for Industrial Applications,” *Journal of Manufacturing Systems*, Vol. 10, No. 4, pp. 297–306, 1991.

[3] I.-M. Chen, and J. W. Burdick, “Determining Task Optimal Modular Robot Assembly Configurations,” in *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 21–27, 1995.

[4] R. Cohen et al. “Conceptual Design of a Modular Robot,” *Transactions of the ASME: Journal of Mechanical Design*, Vol. 114, pp. 117–125, March 1992.

[5] T. Fukuda, et al. “A Study on Dynamically Reconfigurable Robotic Systems Assembling, Disassembling and Reconfiguration of Cellular Manipulator by Cooperation of Two Robot Manipulators,” in *Proceedings of the 1991 IEEE/RSJ International Workshop on Intelligent Robots and Systems*

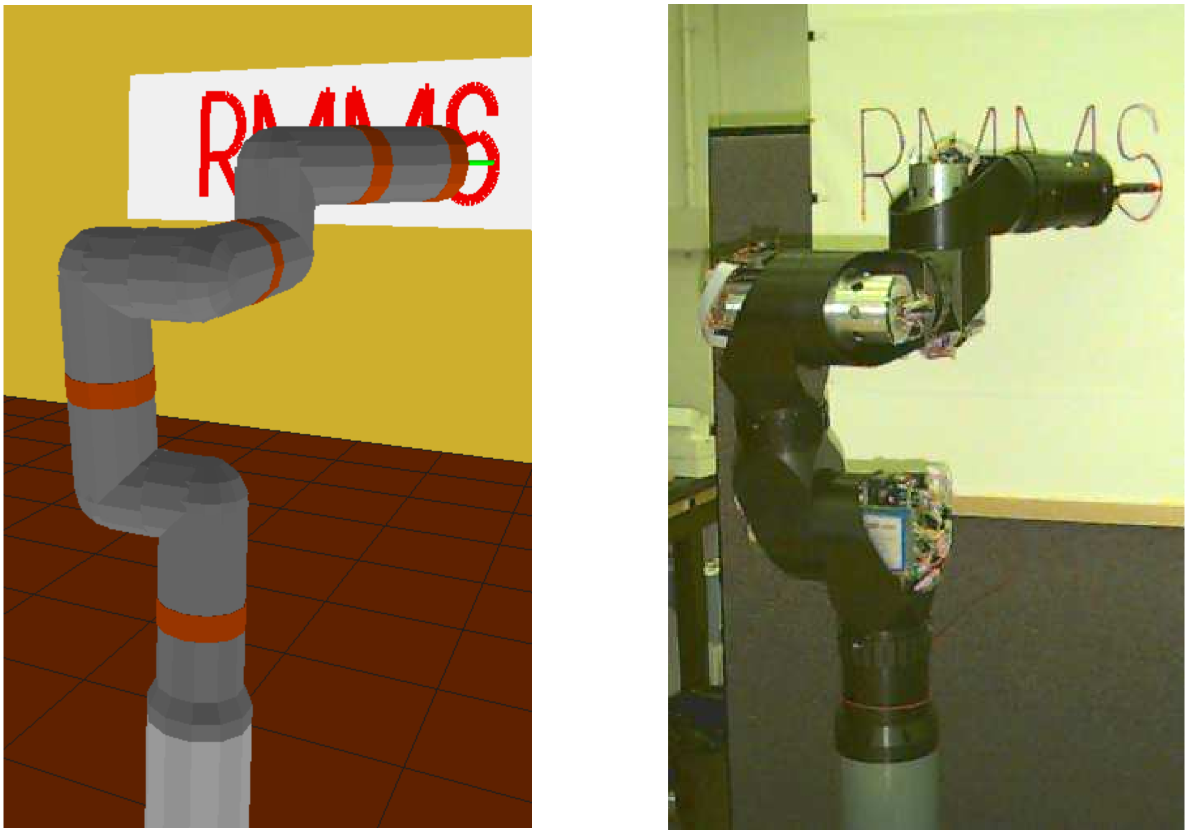

Figure 7: An RMMS configuration: simulation and hardware.

(IROS'91), Osaka, Japan, November 3–5, 1991, pp. 1184–1189.

[6] M. W. Gertz, and P. K. Khosla, "Onika: A Multilevel Human-Machine Interface for Real-Time Sensor-Based Robotic Systems," in *Proceedings of the 1994 Annual Meeting of the American Nuclear Society*, June 1994, New Orleans, Louisiana.

[7] R. Hui et al. "Design of the IRIS Facility—a Modular, Reconfigurable and Expandable Robot Test Bed," in *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, Georgia, May 2–6, 1993, pp. 155–160.

[8] J.-O. Kim, and P. K. Khosla, "A Multi-Population Genetic Algorithm and Its Application to Design of Manipulators," in *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'92)*, Raleigh, NC, July 7–10, 1992.

[9] S. Kotosaka, et al. "Development of a Functionally Adaptive and Robust Manipulator," in *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, Wako, Saitama, Japan, Sep. 21–22, 1992, pp. 85–90.

[10] T. Matsumaru, "Design and Control of the Modular Robot System: TOMMS," in *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 21–27, 1995, pp. 2125–2131.

[11] J. D. Morrow and P. K. Khosla, "Sensorimotor Primitives for Robotic Assembly Skills," in *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 21-27, 1995.

[12] S. Murthy, P. K. Khosla, and S. Talukdar, "Designing Manipulators from Task Requirements: An Asynchronous Team Approach," in *Proceedings of the 1st WWW Workshop on Multiple Distributed Robotic Systems*, Nagoya Japan, July 1993.

[13] C. J. J. Paredis and P. K. Khosla, "Kinematic Design of Serial Link Manipulators From Task Specifications," *International Journal of Robotics Research*, Vol. 12, No. 3, pp. 274–286, June 1993.

[14] D. Schmitz, P. Khosla, and T. Kanade, "The CMU Reconfigurable Modular Manipulator System," in *Proceedings of the 19th International Symposium on Experimental Robotics*, Sydney, Australia, November 6–10, 1988, pp 473–488.

[15] D. B. Stewart, and P. K. Khosla, "Rapid Development of Robotic Applications using Component-Based Real-Time Software," in *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'95)*, Pittsburgh, Pennsylvania, August 5–9, Vol. 1, pp. 465–470.

[16] D. B. Stewart, *Real-Time Software Design and Analysis of Reconfigurable Multi-Sensor Based Systems*, Ph.D. Dissertation, Carnegie Mellon University, Department of Electrical and Computer Engineering, Pittsburgh, PA, April 1994.