To appear in *Proc. of the 1995 IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, May 1995.

Sensorimotor Primitives for Robotic Assembly Skills

J. Daniel Morrow

Pradeep K. Khosla

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Many researchers are interested in developing robust, skill-achieving robot programs. We propose the development of a sensorimotor primitive *layer which bridges the gap between the robot/sensor system and a class of tasks by providing useful encapsulations of sensing and action. Skills can then be constructed from this library of sensor-driven primitives. This reflects a move away from the separation of sensing and action in robot programming of task strategies towards the integration of sensing and action in a domain-general way for broad classes of tasks. For the domain of rigid-body assembly, we are exploiting the motion constraints which define assembly to develop force sensor-driven primitives. We report on the experimental results of a D-connector insertion skill implemented using several force-driven primitives.*

1 Introduction

1.1 Robots: *Flexible* Machines?

The goal of robotics is developing flexible machines which can autonomously (or semi-autonomously) perform a variety of tasks. The current reality is that significant application-specific hardware and software must be developed to support a specific task. This application-specific effort undermines the flexibility goal by requiring significant upfront investment for each new task.

Well-calibrated workcells with precise fixturing are typically required to perform robotic assembly of parts, but this method strongly undermines the flexibility of the robot system. Introduction of sensors is necessary if this structure is to be relaxed, but how to quickly apply them to new tasks remains an open issue. We are developing a sensorimotor layer to bridge the gap between the task space and the robot/sensor space. This sensor integration reduces the requirements for precise fixturing and careful calibration which impede the flexible application of robots. Since assembly tasks are dominated by contact, we are initially focusing on integrating force sensing.

1.2 Key Ideas

The idea of developing robust, skill-achieving programs has been suggested by a number of researchers. Strip [13] has developed an algorithm based on contact-state analysis which will perform convex, prismatic part insertions. Much of his philosophical development centers around the idea of assembly primitives.

Our central idea is to develop an intermediate command layer which integrates sensing and action for a task domain. This layer encapsulates knowledge about the domain as *sensorimotor primitives* (SMP) which are used for implementing task programs. By developing task programs with these primitives, we can leverage our work on previous applications of sensors to similar tasks. This approach differs from other methods which treat sensing and action separately in robot programming. Hopkins et al [5] have suggested pursuing a similar sensor-driven command library. In this section, we outline several of the key philosophical ideas which drive our approach.

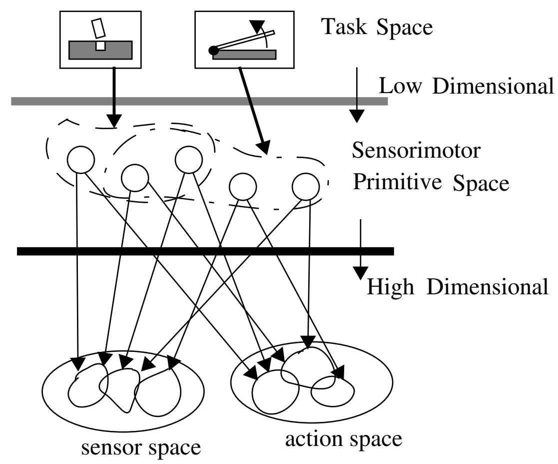


Figure 1: Sensorimotor Space

Reduce dimensionality. A key motivation for developing the sensorimotor layer is to reduce the dimensionality from the task space to the sensor and action spaces of the robot system (Figure 1). The general goal of task-level programming is to reduce the complexity of programming robots by specifying the program in terms of the task [2]. One very difficult part is to bridge the gap between the robot/sensor system and the task description. Deno et al [3] develop control primitives but focus on the dynamical behavior of the robot which has a weak connection to the task description. We propose the sensorimotor layer as the lowest layer in a transition between the robot/sensor system and the task level. Then a human (or computer) task planner can terminate the plan in these primitives instead of low-level, robot/sensor-specific commands. Whereas much task-level programming work [7] emphasizes planning robot motions given a task description, our work focuses on two goals: 1) constructing a new command layer which integrates sensing and action, and 2) implementing skills with commands from this layer.

Leverage sensor use. An important goal of developing this sensor-driven primitive layer is to leverage previous applications of sensors by re-using them. Current efforts to apply sensors are often very task specific and we seek to generalize the use of sensors to a class of related tasks. Erdmann [4] determines the minimum information required to perform a task by designing an abstract sensor for that task. Ahn et al [1] apply a learning algorithm to associate sensor signals with pre-defined error-correcting actions. Both of these approaches seek to connect sensing signals to error-reducing actions of a *specific* task. While Erdmann's method is currently task-specific, his approach could lead to the development of sensors *and algorithms for their use* which are general across a domain of tasks.

Re-using sensors requires the existence of similarities in the way that sensors are applied within a domain. Assembly tasks are similar in that they involve the imposition of constraints between two parts. Morris and Haynes [9] list the common rigid-body constraints and discuss related ideas about developing assembly solutions based on these constraints. It is critical to determine what is common across a task domain to provide a framework within which to develop domain-general, sensor-driven primitives. The common element across rigid-body assembly tasks is the set of motion constraints which define the assembly.

Seek contact. Many current methods of performing assembly tasks (e.g. insertions with remote center of compliance) are fundamentally position-based approaches which are designed to compensate for incidental contact. Flexible execution of contact tasks will require actively seeking contact as a source of information and means of reducing uncertainty during the execution of an assembly task. This reflects a move away from the traditional position-based programming paradigm which seeks to minimize contact and avoid it if possible. Contact provides a rich (if murky) source of information about the task which can be exploited.

Reactively execute. The classic methods of planning robot fine-motions [8] are ill-conditioned for contact tasks because the task model details which strongly influence the behavior of the task program are likely to be wrong. Attempts to consider uncertainty help, but are often too conservative and can fail to generate plans which exist to solve the task.

Smithers and Malcolm [12] discuss "behavior-based assembly" as robust skill-achieving behaviors which can perform assembly tasks. They argue that planners must be used at the symbolic level to form rough plans, and reactive strategies should resolve uncertainty at run-time. While they provide persuasive arguments why this is a good approach compared with traditional planning methods (like fine-motion planning), they do not address the central issue of how to create these behaviors.

While fine-motion planning relies on a diagonal admittance matrix and tries to develop the sequence of nominal motions to perform the task, Schimmels and Peshkin [11] design the admittance matrix according to task requirements. Viewed as admittance matrix design, sensorimotor primitives would be sub-matrices which frequently recur in the programming of tasks.

Related to the reactive-based methods discussed above, Paetsch and von Wichert [10] implement 4 parallel behaviors to perform insertion of both round and square pegs. The behaviors are derived in an ad hoc manner from observation of human insertion strategies. We follow a similar task-strategy generation method in this paper.

Our initial work does not seek to create planners, but rather to develop a sensorimotor layer and demonstrate its application to the programming of skills. We need to base this layer on task models which can reasonably be expected to agree with the real-world. Reactive strategies will be required because the models will not contain all of the necessary detail.

2 Sensorimotor Primitives

2.1 What is a Sensorimotor Primitive?

A sensorimotor primitive is an encapsulation of sensing (processing of sensor feedback) and action (trajectories) which can form *domain-general* building blocks for task strategies. A primitive is not an autonomous agent which can intelligently intervene at appropriate places in the task, but rather a more powerful and task-relevant command. The goal is to provide a library of sensor-driven commands which effectively integrate sensors into a robot system for a particular class of tasks.

We differentiate between skills and primitives. A skill is a parameterized, stand-alone, robust solution to a specific task. A sensorimotor primitive is a parameterized, domain-general command which integrates sensing and action and can be applied to many skills within a task domain. The goal is to capture important domain information or capabilities so that robust skills can be quickly developed.

2.2 Primitive Classes

The initial work in developing sensorimotor primitives has resulted in several primitive classes. These classes help to identify the different types of primitives. If we view primitives as having two inputs and one output, we can enumerate the possibilities.

Table 1: Primitive Classes

input 1	input 2	output	function
sensor	sensor	sensing	sensor fusion
sensor	sensor	action	reactive planning
sensor	action	sensing	**active sensing**
sensor	action	action	**perturbation**
action	action	sensing	active sensing
action	action	action	complex trajectory

We are specifically interested in the shaded primitive classes which encapsulate sensing and action inputs to generate their output; we name these types *sensorimotor primitives* to indicate their integration of sensing and action.

Consider the "active sensing" SMP. The key is to use the command which generated the sensor signal as added information in processing the sensor signal. This allows, for example, correlation to reject noise and bias from the sensor signal. Using the command in processing the sensor signal effectively provides context which is otherwise missing. Lee and Asada [6] have applied correlation techniques to robotic insertion of a heat exchanger tube.

Next, consider the "perturbation" SMP: this primitive is often referred to as a *behavior* in the robotics literature, since it directly produces an action from a sensor signal. Within this type of primitive class there are two subclasses: linear and nonlinear action SMP's. In a linear SMP, the perturbation is generated entirely by sensor feedback, without regard for the input action. An example of this type of primitive is the linear accommodation which maps a force to a velocity change (which is summed with the current command to produce the output). Another sensor-driven primitive is a threshold termination condition in a guarded move. The computation of the perturbation is independent of the input action and depends only on the sensor signal.

The second subclass, nonlinear action, is a combination of the two types where sensing and action are combined to interpret sensing and modify the output action. The key is that the computation to determine the output action involves the input action. An example of this type of primitive is a hole centering primitive which modifies the parameters of the input action. Here the goal is to modify the "zero" point and amplitude of a sinusoidal trajectory of the hand. Computing these parameters requires knowledge of both the input trajectory and the sensor feedback.

2.3 Example Primitives for Assembly

Some of the primitives listed below rely on the presence of a damping controller to supply compliant (accommodation) motion of the robot.

Guarded move. This primitive is one of the only common sensor-integrated commands available in robot programming languages today. It is ubiquitous in assembly programs for removing uncertainty. Our variation is a multi-state module which can transition to a "contact" velocity to maintain contact with the surface and which detects lost contact by monitoring the position change after contact is made.

Sweep. This primitive is a variation of the guarded move in which a coordinated cartesian trajectory is executed and, upon detection of contact, the transition to a maintenance velocity offset is performed. The sweep trajectory combines sinusoidal linear and angular velocities to effect a trajectory specifically designed to acquire contact of an edge.

Correlation. This primitive is a tight coupling between sensing and action where the commanded action is used to interpret the sensor information. In this case, the command is a sinusoidal cartesian velocity which is correlated with the forces and torques from the force sensor. This provides robust information from a biased and noisy sensor.

Accommodation. Various sub-matrices of the damping controller are linear perturbation primitives. The linear compliance submatrix is common as a fundamental component of a damping controller, but the other submatrices are not always applicable (e.g. those mapping forces to rotary motion to effect a remote center of compliance).

3 Assembly Skill Strategies

There is great interest in developing robust skill strategies for performing common assembly tasks. The difficulty is that a general method of synthesizing robust solutions to these tasks has not been developed. We are suggesting that a *sensorimotor* layer, which is a lower level than skills but higher than the sensor and action spaces, should be developed to ease the development of robust skills. This layer will form a basis for skill strategy encoding and will reduce the dimensionality from the skill space to the sensor and action spaces by encapsulating useful combinations of sensing and action.

3.1 Skill Synthesis Methods

For rigid-body assembly tasks, the key question is how to synthesize robust solutions. Synthesizing fine-motion solutions based on detailed geometric models is limited in robustness because of the inevitable model error.

The most important features in assembly are the constraints. In fact, all assembly skills are fundamentally the introduction of constraints between two parts. We are trying to develop skill strategies which are constraint-based: that is, try to acquire and maintain constraints as landmarks to the task strategy. Our initial experiments use heuristic strategies based on the constraints of the assembly. These strategies use a successive acquisition of constraints to achieve the assembly.

Behind this approach is the desire to base skill strategies on our most robust description of the task (its constraints). But we must still command specific motions of the manipulator so we need reactive behaviors to help us deal with the errors associated with our higher-level (constraint-based) goal representation. We believe that basing our SMP's on constraint acquisition and maintenance is a promising approach which will yield general primitives with which to build assembly skills.

3.2 Example Task: D-Connector Insertion

As example assembly task, we have chosen a standard "D" connector insertion (both 9-pin and 25-pin). We assume that we have a reasonably good idea of the connector position and orientation in the hand, and we align the insertion axis as well. We begin with the connector biased to one side of the mating part.

The strategy begins with a guarded move along Y to remove the bias and find the long edge of the connector. Upon finding the edge, the guarded move primitive transitions to a “maintain” state and monitors for loss of contact. The next step is to move up along the insertion axis until contact is lost. The guarded move primitive identified contact loss by noting a position movement in the direction it was trying to “stick.” We have set this distance for approximately the width of the connector. At the end of this move the connector is poised approximately above the mating connector. Now we begin two sinusoidal dithers: 1) rotation about the insertion axis to twist the connector; 2) translation along the long axis of the connector. A translation along the insertion axis (while dithering) is usually successful at mating the part.

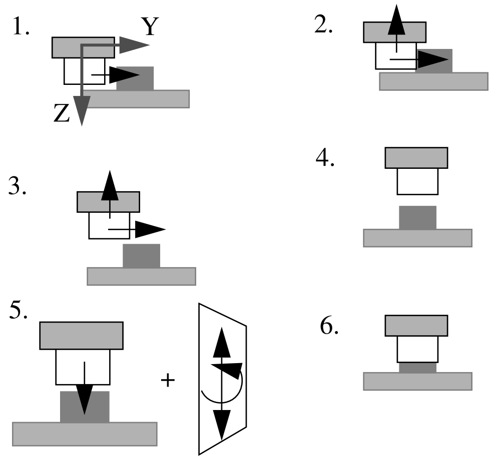


Figure 2: D-connector Insertion (short-side view)

An important element of the connector insertion is detecting success. Note that since we have not taught an absolute position to attain, we cannot simply identify proximity to a particular position. Instead, we employ an active sensing method to detect the acquisition of the proper constraints. The nominal input velocity will be perturbed by the linear accommodation controller and result in a different reference velocity when the connector has seated. By performing a normalized correlation of these two signals, we are able to robustly identify the task goal state. Early attempts to use the change in cartesian position after contact in the insertion axis proved to be very unreliable due to the noise and variation in that signal.

The D-connector insertion strategy reflects the encapsulation of useful activity into sensor-integrated primitives which are then invoked by the skill. The enhanced guarded move SMP which also monitored for loss of contact is valuable at encapsulating the acquisition and maintenance of contact. In addition, the active sensing SMP proved a very robust method of “differentially” detecting the goal state by verifying the loss of motion freedom in the x (dither) direction. Both of these primitives are based on constraints in the assembly task. Finally, the dithering signals were added to effect a complex trajectory from simple elements.

4 Active Sensing

The use of active sensing primitives is critical to fully exploiting our sensors by providing a context in which to interpret the signals. In the case of the D-connector, we used active sensing to identify the acquisition of the goal state by identifying the constraints which it imposed. Note that this constraint-detection method is very general and applies to many other assembly skills, not just D-connector insertion.

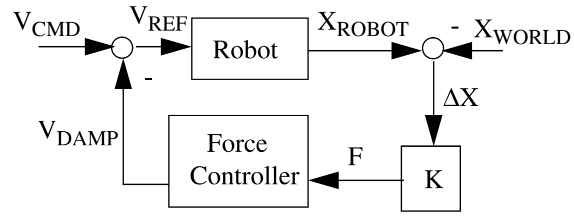


Figure 3: Damping Controller

Referring to Figure 3, in the absence of strong force coupling between the world and the robot, the reference velocity closely tracks the commanded velocity. However, when constrained motion occurs, we expect to see significant deviation between the commanded velocity and reference velocity caused by the forces of the constraint. For finite stiffness K, we expect the forces to be 90 degrees out of phase with the commanded velocity.

In order to detect the acquisition of a constraint, we perform a normalized correlation between the sinusoidal commanded velocity and the reference velocity. The normalization is necessary to compensate for any amplitude changes which may occur after constraint acquisition.

$$C = \frac{\int_0^T f(t)\, g(t)\, dt}{\int_0^T |f(t)|\, dt \int_0^T |g(t)|\, dt} \tag{1}$$

The discretized implementation of this correlation function is:

$$C = \frac{\left(\sum_{i=0}^{N} f\left(\frac{i2\pi}{N}\right) g\left(\frac{i2\pi}{N}\right)\right) N}{\sum_{i=0}^{N} \left|f\left(\frac{i2\pi}{N}\right)\right| \sum_{i=0}^{N} \left|g\left(\frac{i2\pi}{N}\right)\right|} . \tag{2}$$

When f and g are in-phase sine functions, the resulting correlation is $\pi^2/8$. We implement this by commanding a sine function velocity trajectory and computing the correlation using a sliding sampling window. For the first cycle, no correlation is computed so at least 1 cycle is performed of the command before the constraint can be recognized.

5 Experiments

5.1 Experimental Setup

The experimental setup is a PUMA 560 with a Lord 15/50 6-axis force/torque sensor with VME-based computing running the Chimera 3.2 real-time operating system. Implemented as Chimera port-based modules, the SMP’s can be easily reconfigured into a wide variety of skills. Figure 4 shows the SMP-based skill architecture. The numbers in the skill states refer to the numbers in the D-connector insertion strategy illustrated in Figure 2. Notice that the primitives provide an intermediate layer between the skill and the robot and sensor in this architecture. The skill strategy, implemented as a finite-state machine (FSM), is easily specified through text "configuration" files and all the C code is embedded in the sensorimotor layer. The idea is to relieve the task programmer from dealing directly with the robot/sensor space by providing a set of task-relevant commands which facilitate the implementation of a task strategy (skill).

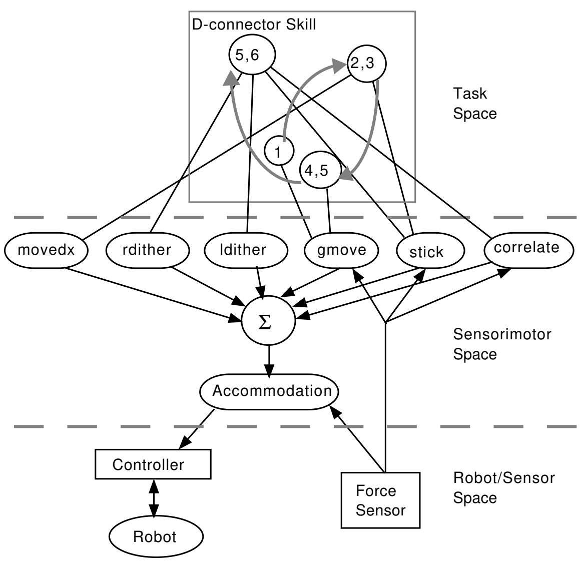
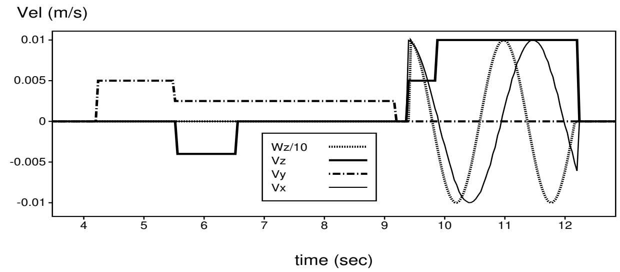


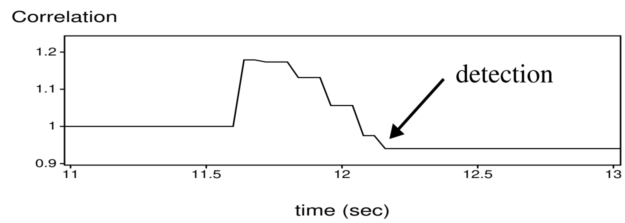
Figure 4: SMP-based Skill Architecture

5.2 Experimental Results

Approximately twenty different D-connector insertion experiments were performed to test the SMP-based strategy. The following data is representative of the results. Figure 5a shows a history of commands for a complete (successful) D-connector insertion. All the steps outlined earlier in the strategy description are visible on this plot. Notice that the rotation and linear dithers are different frequencies so that different trajectories are executed over time rather than the same one repeatedly. Figure 5b shows the corresponding use of the correlation computation with a threshold (0.95) to detect the successful mating of the connector. Note that the correlation function is only computed once a full linear dither cycle has executed. Thus, it applies only toward the end of the velocity commands. Note that the "steady-state" value of the correlation when the reference and command velocities agree is around our predicted value of 1.2.



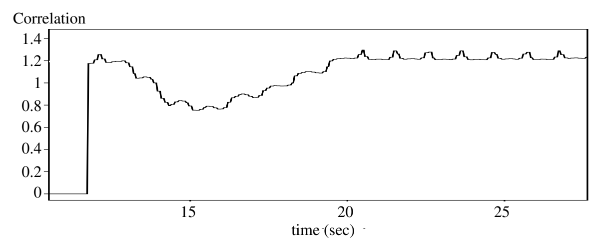
(a) Velocity Commands



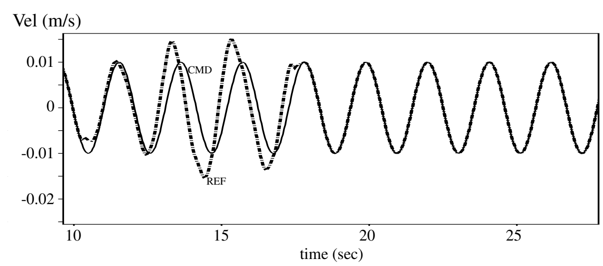
(b) Goal Detection with Correlation

Figure 5: Successful Connector Insertion

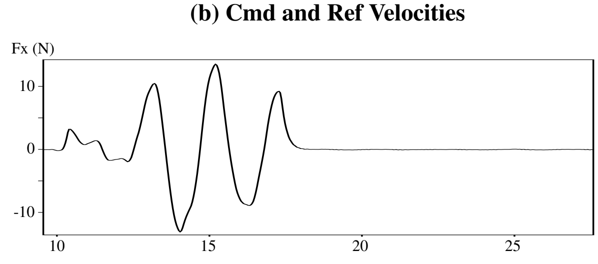
In order to view the effects of correlation, Figure 6 shows results from a run in which the goal-detection SMP was turned off. The mating occurred, but since the motion commands did not stop, the connector was "worked off" through the motion.



(a) Correlation Value



(b) Cmd and Ref Velocities



(c) Force Feedback

Figure 6: No Goal Detection

Figure 6a clearly shows the strong signal which the correlation algorithm computes to identify the region where the commanded and reference velocities differ. Figure 6b shows how the reference velocity closely tracks the commanded velocity when the force feedback is negligible (Figure 6c). Note that after the connector disengages, the guarded move SMP interprets the movement as a loss of contact and terminates the downward offset velocity. Thus, no contact forces are generated for the last half of the dithering in this example.

6 Discussion

6.1 Comments on Experimental Results

The algorithm developed for insertion of the D-connector is based on sensor-driven primitives such as a sticking guarded move and a correlation function to detect the constraint once the connector is seated. The algorithm works for both 9-pin and 25-pin D-connectors and is fairly robust to certain classes of errors (e.g. displacement perpendicular to insertion axis). However, it is quite sensitive to errors in the insertion axis. Biasing the insertion axis direction by rotating slightly about the x-axis to "open" the approaching connector improves the robustness significantly.

One problem that we saw was motion of the part relative to the gripper once the connector seated. We used a pneumatic gripper with limited gripping force and a friction hold on the part. These significant motions violate the constraints that we hope to detect, since the constraints attempt to enforce no motion. Even so, we can detect the interaction forces which is enough to detect the successful insertion. Future work will develop more secure grasp strategies and hardware to alleviate this problem.

6.2 Conclusions

The guarded move SMP effectively encapsulates the acquisition, maintenance, and detection of a simple contact constraint. The correlation SMP encapsulates the detection of a motion constraint. Given these types of encapsulations of sensing and action, the skill-level is free to consider what behavior is desired and to implement this behavior with the primitives.

We have proposed the development of a sensorimotor layer specifically designed to bridge the gap between task-achieving skills and the sensor and action spaces of a robot/sensor system. The key idea is to develop a library of useful encapsulations of sensing and action which can hide some of the complexity from programming sensor-based skills. We are pursuing this goal from the direction of exploiting the motion constraints which appear in all assembly tasks.

6.3 Future Work

A number of challenges exist to extend this work. First, we must develop a richer set of general sensor-driven primitives for application to assembly tasks. These primitives need to be applied to a broad selection of related tasks to show that they are indeed general and not specific to individual tasks. And finally, we need to quantify the improved robustness and/or speed which programming with such primitives provides over conventional methods.

7 Acknowledgements

This work was supported in part by the Computational Science Graduate Fellowship (CSGF) Program of the Office of Scientific Computing in the Department of Energy. The experimental work was started during a CSGF practicum (summer of 1994) at the Intelligent Controls Group of the National Institute of Standards and Technology. The authors would like to thank Al Wavering, Tom Wheatley, and Ron Lumia for their assistance during this time.

8 References

[1] D.S. Ahn, H.S. Cho, K. Ide, F. Miyazaki, and S. Arimoto, "Strategy Generation and Skill Acquisition for Automated Robotic Assembly Task," *Proceedings of the 1991 IEEE International Symposium on Intelligent Control*, Arlington, VA, 13-15 August, 1991.

[2] J.J. Craig, *Introduction to Robotics: Mechanics and Control*, Ch. 12, Addison-Wesley Publishing Co., Reading, MA, 1986.

[3] D.C. Deno, R.M. Murray, K.S.J. Pister, and S.S. Sastry, "Control Primitives for Robot Systems," pp. 1866-1871, *Proceedings of IEEE International Conference on Robotics and Automation*, Cincinnati, OH, 1990.

[4] M. Erdmann, "Understanding Action and Sensing by Designing Action-Based Sensors," *Int. J. of Rob. Res.*, In press.

[5] S.H. Hopkins,.C.J. Bland, and M.H. Wu, "Force sensing as an aid to assembly," *Int. J. Prod. Res.*, vol. 29, no. 2, pp 293-301, 1991.

[6] S. Lee and H. Asada, "Assembly of Parts with Irregular Surfaces Using Active Force Sensing," pp. 2639-2644, *Proceedings of IEEE International Conference on Robotics and Automation*, San Diego, CA, 1994.

[7] T. Lozano-Perez, "Task Planning," Ch. 6, *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982.

[8] T. Lozano-Perez, M.T. Mason, and R.H. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots", *Int. J. of Rob. Res.*, Vol 3, No 1, pp. 3-23, Spring, 1984.

[9] G.H. Morris and L.S. Haynes, "Robotic Assembly by Constraints," pp. 1507-1515, *Proceedings of IEEE International Conference on Robotics and Automation*, Raleigh, NC, 1987.

[10]W. Paetsch, and G. von Wichert, "Solving Insertion Tasks with a Multifingered Gripper by Fumbling," pp. 173-179, *Proceedings of IEEE International Conference on Robotics and Automation*, Atlanta, GA, 1993.

[11]J.M. Schimmels and M.A. Peshkin, "Admittance Matrix Design for Force-Guided Assembly," *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 2, pp. 213-227, April, 1992.

[12]T. Smithers and C. Malcolm, "Programming Robotic Assembly in terms of Task Achieving Behavioural Modules," DAI Research Paper No. 417, University of Edinburgh, Edinburgh, Scotland, December 1988.

[13]D. Strip, "Technology for Robotics Mechanical Assembly: Force-Directed Insertions," *AT&T Technical Journal*, Vol. 67, Issue 2, pp. 23-34, March/April 1988.