TRANSFERRING ASSEMBLY SKILLS TO ROBOTS: LEARNING FORCE SENSORY PATTERNS AND SKILLS FROM HUMAN DEMONSTRATION

A Dissertation

by

MARJORIE A. SKUBIC

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 1997

Major Subject: Computer Science

TRANSFERRING ASSEMBLY SKILLS TO ROBOTS: LEARNING FORCE SENSORY PATTERNS AND SKILLS FROM HUMAN DEMONSTRATION

A Dissertation

by

MARJORIE A. SKUBIC

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Richard A. Volz
(Chair of Committee)

John Yen
(Member)

Jeffrey C. Trinkle
(Member)

Louis Everett
(Member)

Richard A. Volz
(Head of Department)

August 1997

Major Subject: Computer Science

ABSTRACT

Transferring Assembly Skills to Robots: Learning Force Sensory Patterns and Skills from Human Demonstration. (August 1997)

Marjorie A. Skubic, B.S., South Dakota State University, Brookings, SD;

M.S., Texas A&I University, Kingsville, Texas

Chair of Advisory Committee: Dr. Richard A. Volz

Robots are distinguished from other automated machines by their capability of being programmed to perform a variety of tasks. However, the difficulty in programming robots remains one of the barriers preventing their full utilization. This is especially true in tasks such as assembly operations, which involve the robot making contact with its environment.

This research addresses the problem by studying methods of transferring human, force-based assembly skills to robots, with the use of a natural interface between the human operator and the robot. The interface is modeled as a teacher-student relationship, where the human demonstrator acts as a teacher and the robot plays the role of student. A hybrid control system (HCS) is used for modeling assembly skills, because it provides a means of interfacing the symbol processing used by the human teacher with the signal processing used by the robot. Using the HCS model, an assembly skill is described as a sequence of qualitative states and the desired transitions between the states. In this case, the qualitative state takes the form of a single-ended contact formation, which describes how a workpiece touches its environment. Skill acquisition involves learning the sequence of qualitative states, the transitions between those states, and the mapping from sensor signals to the qualitative states.

We discuss two methods of classifying single-ended contact formations from force sensory patterns, without using geometric models of the workpieces. In the first method, fuzzy logic is used to model the patterns in the force signals, with membership functions being generated automatically from training data. In the second method, a neural network architecture is used to learn the mapping from force signals to the qualitative states. The training and performance of the classifiers is discussed.

Experimental results are presented which illustrate and validate the approach. The technique is used to extract skill information from human demonstration data. Successful test runs show how the classification technique and the HCS model can be used to provide robust execution of the skill, even without prior position information.

To

Dan, Theresa, and Jake

ACKNOWLEDGMENTS

There are many people who have contributed to the successful completion of this dissertation. Let me first thank my committee chairman, Dr. Richard Volz, for his guidance and support, in spite of his incredibly busy schedule, and for his persistent discussions that pushed me to run that one last test. I have come to appreciate his ability to immediately zoom in on the most significant strengths and weaknesses. I would also like to thank my other committee members, Dr. Jeff Trinkle, Dr. John Yen, and Dr. Louis Everett, for taking the time to review my work and offer helpful comments and suggestions for future work. The interactions I have had with all of my committee members will undoubtedly shape my career for years to come.

There are numerous other people that have helped me in the years I have spent at Texas A&M. I would like to especially acknowledge faculty members Dr. Nancy Amato, Dr. Jennifer Welch, and Dr. Thomas Ioerger for their helpful discussions. And I would like to thank Willis Marti, not only for his computer support, but also for his friendship. Over the years, there have been many students in the Robotics Lab ready to offer help, especially Ayman Farahat and the old USARC team of Sean Graves, John Mollenhauer, and Bryan Morgan. A special thanks goes to Shawn Castrianni for his help with neural networks and his readiness to do anything that needed to be done. Thanks also to Holly Patterson who helped me keep my sanity.

My family has also provided much inspiration over the last several years. My husband, Dan, has been ready to offer any kind of help I needed, even when I thought I didn't want it. My children, Theresa and Jake, have made sacrifices and generally put up with Mom's little adventure. Thanks for helping to keep my life balanced. I also thank my parents for teaching me the value of an education and then being there to help me complete it. And I thank my brothers for their advice on how to survive graduate school and what to do when you're finished.

This work has been supported by several grants and fellowships. In particular, the work has been supported by the National Science Foundation under grants CDA-9115123, CDA-9422123, and EEC-9017249, by the Department of Education under grant P200A40713, by the State of Texas under TATP grants 999903-267 and 999903-095, and finally by a fellowship from General Electric. A special note of thanks goes to Dr. Karan Watson, as several fellowships have come through her office, and she has continued to be a source of emotional and financial support for so many students in the College of Engineering.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

A. Motivation

Robots have been used successfully in manufacturing settings, where the environment is very structured and the tasks performed are repetitive and relatively simple. As long as the environment is controlled and the workpieces are confined to precise positions and orientations, the robot can continue to execute its task. Although these robots may be simple automation tools, they perform a useful function and can relieve human workers from boring, repetitive production tasks. In some cases, the robots may even perform the task better than the human operators. But this functionality is very much dependent on maintaining a structured and controlled environment.

This research is inspired by the vision of robots moving beyond the structured, controlled settings, into unstructured and uncontrolled environments and still performing useful functions. The intelligent android sometimes depicted in science fiction movies and books is a long way from reality. That level of autonomy and learning in robots still has many challenging problems. However, it is realistic to think of robots being used as intelligent tools which make the job easier for the human worker or allow a human to avoid a hazardous environment.

This dissertation follows the style and format of the *IEEE Transactions on Robotics and Automation.*

Depending on the nature of the work, the robot may be asked to perform repetitive or non-repetitive tasks. If it is a new task not performed previously, the robot may need some quick instruction or task refinement (i.e., programming), most likely done at the work site with instruction provided by the actual user rather than a specialized robot engineer. The robot may be positioned next to its human operator or it may be located at a distance. In either case, the robot will be more functionally useful if it has some level of local autonomy, such as the ability to safely perform low-level sensorimotor skills. This would allow the human operator to use supervisory control over the robot, thereby reducing his workload and extending his capabilities.

As more sophistication is added, the robot may eventually serve as an assistant to the human operator. The robot may be used to perform the more menial or more dangerous tasks as the human worker and the robot work side by side (functionally if not physically). The human-robot relationship might be compared to that of a seasoned foreman working with an inexperienced human assistant. In such a situation, the robot may require on-the-job training just as a human worker might.

In summary, the motivation behind this research is simply to make robots more useful – to provide strategies which allow them to perform a greater variety of tasks in more environments, working more efficiently and to be more easily reprogrammed for new jobs by the people that use them. There are several stumbling blocks which may prevent this vision from becoming a reality. One of the critical software-related issues will be addressed here.

Robots are distinguished from other automated machines by their capability of being reprogrammed to perform a variety of tasks. However, the difficulty of programming robots remains one of the barriers preventing the kind of interactive, non-repetitive use described above in our vision of intelligent robot tools and assistants. This is especially true in tasks which involve the robot making contact with its environment.

Even in the structured manufacturing setting, robot programming can be time-consuming and may require special expertise. Generally, programming of contact tasks relies on very precise positioning of all the workpieces, often achieved by specialized tools or fixtures. Different tasks may require different parts, different tools and fixtures, and different positioning information. Obtaining this precise positioning information for each task can be difficult. Teach pendants can be used so that an operator guides the robot to the proper position and then records each position. This can be helpful but may not provide the entire program. It may also be tedious because a precise position point must be captured. Off-line programming using a robot-control language or a graphical interface can also be helpful, but this generally requires special expertise and can be time-consuming. Newer methods using model-based motion planning are usually computationally-intensive and can also be time-consuming. They also require accurate models and precise registration between the models and the real parts.

Now imagine extending these programming techniques to an unstructured environment, where the position and orientation of parts is not known. To safely and successfully complete contact tasks, additional sensory information is required. One approach is to use on-line data from a force sensor and program the robot for compliant motion. Current programming techniques require a specialized robot engineer to devise compliant motion strategies that depend on the task to be performed. New tasks would require new compliant motion strategies and, thus, reprogramming. Another possibility is for the robot to completely learn its new environment, as in a vision system extracting models of the workpieces in the environment. Model-based motion planning can then be used. Some progress has been made in this endeavor; however, it adds even further computations and time to model-based approaches, because the robot has to first learn about its environment and adjust its models. Neither of these methods seem practical for fast, interactive programming of intelligent robot tools in an unstructured environment.

An alternate solution is to let the human worker help with the programming job, but to provide an interface that makes it easy for him to help. This falls into the general category of *robot programming by human demonstration* and has become an active research area. The related research will be discussed in detail in Chapter 3. Much of the work is a variation of the old idea of using a teach pendant. Additional sensory information is added so that a complete program can be learned and/or compliant motion strategies can be learned, simply by having the user demonstrate the desired task.

The goal is to provide a method of programming which can be done quickly and easily and which does not require the talents of a trained engineer. This would certainly be a benefit in the manufacturing setting, allowing robots to be easily reprogrammed and, therefore, economically feasible for even small-quantity jobs. The programming by demonstration method may also meet our vision of interactive programming for non-structured environments, provided there is a way to deal with position and orientation uncertainties.

B. Research Goals

Although much work has been done, research has not yet provided a fast, interactive method of programming robots for contact tasks in unstructured environments. The research discussed in this dissertation will focus on this problem by learning low-level force-based assembly skills (rather than position-based skills) from human demonstration. The ability to quickly transfer force-based skills to robots will dramatically increase the functional capabilities of our envisioned intelligent robot tools and assistants.

The research goals of this research are as follows:

- To learn low-level force-based sensorimotor skills from human demonstration
- To provide a natural interface for the human demonstrator, making the demonstration process easy
- To use a strategy for the learned skill so that it can be executed robustly, even with uncertainties, and not be dependent on position or orientation
- To be able to execute the learned skill (including sensory perception) in real-time

The focus here is not on trying to learn an entire program but rather to concentrate on the part of the program which involves making contact with the environment. That is, we want to learn the force-based sensorimotor skill which makes up part of a larger program. Ultimately, some other technique would be used to get the rest of the program (e.g., vision, human demonstration using a sensing modality other than force). Also, during execution of the learned program, we will assume that some other means is used to position the robot close to the point of contact, not necessarily in a known position but simply close to any point of contact with the environment. A vision system (e.g., visual servoing) might serve this function, or it might result from supervisory control where a human operator is in charge.

To achieve a *natural* interface for the human demonstrator, we will design the human-to-robot interaction to be similar to a human-to-human interaction. That is, we will use an anthropomorphic model of the interaction between the human demonstrator and the robot. In this case, a teacher-student model is used, where the human demonstrator plays the role of the teacher, and the robot plays the role of the student. One might compare the human-robot interaction to that of an experienced worker training a neophyte or that of a parent teaching a child.

To overcome uncertainties and be independent of position and orientation, we will use an event-based approach. The sensorimotor skills will be modeled using force-based discrete states, which describe qualitatively how contact is being made with the environment. The discrete states will be referred to as qualitative states. We will investigate the applicability of a hybrid control model, in which a continuous-time system is controlled by a discrete event controller. A change in qualitative state constitutes an event and triggers a new control command to the robot. In this way, the skill execution is not dependent on absolute position but rather responds to changes in the force-based qualitative state.

To be able to execute the learned skill in real-time, the computational load must be kept to a minimum. We will, therefore, avoid the use of detailed geometric models and computationally-intensive model-based approaches. Instead, we will investigate sensor-based methods, which use force sensor data only. It is the belief here that the ultimate interface would include multi-modal information, incorporating vision, force and voice input, as well as utilizing some geometric information. However, this research will concentrate on using force signals to see what can be done with force alone. We will explore the possibilities and the limitations, with the ultimate hope of gaining further insight on how to combine force information with geometric information and/or other sensing modalities.

C. Outline

The remainder of this dissertation is organized as follows. Definitions will be given in Chapter II, and the evolution of related research will be discussed in Chapter III. Chapter IV will present a description of the methodology used, including the skill model, the sensory perception of the qualitative states, and the transitions between qualitative states. In Chapter V, the hardware and software of the testbed will be described for both the demonstration environment and the control environment. The prototype experiments, along with results and analysis will be reported in Chapter VI. Representative results will be presented in Chapter VI, and the complete set of results are provided in Appendix A. We will conclude with Chapter VII, citing the contributions of the research, as well as the limitations and suggestions for future work.

CHAPTER II

DEFINITIONS

The following definitions are provided, as they relate to robotics systems:

- **actuator** - a mechanism for moving a part of a robot
- **anthropomorphic user interface** - a communication mechanism between a human and a machine which is based on a model of human-to-human communication [18].
- **automaton** - a control mechanism designed to follow automatically a predetermined sequence of operations [73].
- **contact formation** - the set of configurations with identical, elemental contacts between pairs of topological elements [16]. The contact formation provides a qualitative description of how 2 or more objects make contact with each other, reflecting the geometric relationship.
- **contact state** - as used here, identical to *contact formation.*
- **control** - as used here, automatic feedback control of a machine (e.g., robot), in which a system tries to maintain a specified relationship between a reference input and the system output [53].

- **hand controller** - a mechanism which is used by a human operator to issue motion commands to a telerobot.

- **hybrid control system** - a control model in which a continuous-time plant (i.e., system) is controlled by a discrete event controller [68].

- **learning** - extracting knowledge which can be used to change behavior. A robot might learn how to plan, how to program, how to control, or how to sense something.

- **plan** - a specification of the activity of an agent which directs it in some purposeful manner [22].

- **planning** - automatically generating a plan from a high-level description of the task.

- **programming** - converting a plan into robot commands (either off-line or on-line).

- **qualitative state** - the equivalence class resulting from a mapping of sensor signals into discrete, perceptually homogeneous regions [12].

- **robot** - a machine characterized by actuators and sensors connected via one or more control loops, which can be programmed to perform a variety of functional tasks.

- **robot command** - a directive issued to the robot which causes an action to occur.

- **robot programming by human demonstration** - any of a variety of demonstration modes provided by a human operator (e.g., guiding the robot by using a hand controller, off-line simulation-based programming, robot learning by observing human operation).

- **sensor** - a device which responds to a physical stimulus and transmits a signal. A robot may use one or more sensors to determine its status or environmental conditions.

- **single-ended contact formation** - a one-sided contact formation (i.e., the set of configurations with elemental contacts between a grasped object and any point in its environment). The single-ended contact formation provides a qualitative description of how an object makes contact with its environment.

- **skill** - a robust, reusable strategy which can be used for similar recurring, complex tasks.[1]

- **state** - The state at t_0 of an n^{th} order system is described by a set of n variables which, along with the input to the plant for $t \geq t_0$, is sufficient to determine the behavior of the plant for all $t \geq t_0$. [47].

- **supervisory control** - a control mode in which a human operator issues commands (including motion commands) to a machine (e.g., computer, robot) and

[1]The objective of this work is to find a concrete model of a skill in a mathematical form, which successfully achieves a robust, reusable strategy for assembly tasks. A precise model is presented and discussed in Chapter IV.

monitors status. The machine closes an autonomous control loop through actuators and sensors to execute the commands [60].

- **task** - a piece of work that is to be performed, either by a human operator, a robot, or a combination of the two (e.g., completing an assembly operation).

- **telerobot** - as used here, a robot controlled by a human operator via a computer intermediary, using direct manipulation and/or supervisory control.

- **user interface** - the two-way communication mechanism and/or environment between a human operator and a machine (e.g., computer, robot).

CHAPTER III

RELATED RESEARCH

This chapter offers a brief overview of related research in three main areas of study as they relate to this dissertation: understanding and using force signals, understanding the human, and robot programming by demonstration. The intent is to show the evolution of research which has resulted from both limitations and improvements in technology. This evolution will be used to provide a foundation on which a new methodology can be built.

The research discussed in this dissertation also encompasses pattern recognition techniques based on neural networks and fuzzy logic. These topics will not be discussed in detail in this chapter; however, related references will be introduced throughout the dissertation as the methodology is presented. Several good references can be used for a general introduction to the topics, including [6, 7, 77] for fuzzy logic and [20] for neural networks. Also, [36] provides insights on using both neural networks and fuzzy logic.

A. Understanding and Using Force Signals

The use of force signals dates back to the 1950's and 1960's with force feedback for remote manipulators and artificial arm control [75]. More recently, force signals have been used for force control in manipulation tasks which require the robot to make contact with its environment (e.g., assembly tasks). Coupled with compliant motion strategies, force control enables the robot to perform contact tasks more safely and reliably, overcoming stability problems and small uncertainties in position. Various force control techniques have been proposed, including impedance control [28], stiffness control [57], and hybrid position/force control [56]. Whitney provides an interesting historical review of force control methods, as well as unsolved issues [75].

The hybrid position/force control method has become one of the most widely-used control strategies for robotic manipulators. In this method, each Cartesian degree of freedom is controlled by either force control or position control. Mason has proposed a theoretical framework for contact tasks, which works well with hybrid position/force control [41]. The geometry of the workpieces is analyzed according to the nature of the contacts encountered in the desired task. Mutually orthogonal directions are defined as either naturally constrained or artificially constrained. These directions specify a task frame, centered at the point of contact, which is used to specify desired force and position trajectories.

Force signals have also been used to determine the topological contact, or contact formation, between workpieces in assembly tasks. The contact formation can be viewed as a qualitative force-based discrete state which describes how 2 or more objects make contact with each other. The concept was initially proposed by Desai and Volz as a means of achieving more robust execution of programs [16]. The original use assumed the automatic generation of assembly programs from CAD models. As such, most methods of identifying contact formations use detailed geometric models of the assembly parts, as well as force signals.

Desai and Volz identified the contact formation by first formulating a hypothesis and then verifying that it was consistent with the sensed force signals [16]. They used static equilibrium equations and solved for the contact forces and moments. In cases where the contact forces could not be determined, or ambiguities existed, active force sensing was used. Motions were made along the separation cone (a separation cone is set of all motions that result in a break in contact) and more sensing data was collected.

Hirai and Asada presented a method of deriving contact classifiers from the geometric model using polyhedral convex cones [25]. The ranges of possible forces and displacements that can be measured at each contact formation were represented as the union of polyhedral convex cones (PCC). The face vectors of the PCC's were used to determine the contact formation from sensor signals.

Other model-based methods have also been proposed for identifying contact formations. Farahat, Graves and Trinkle used linear programming to determine the feasibility of a contact formation, based on the force signals and the geometric model [19]. Kitagaki. et al used active force sensing to estimate the point of contact between objects [34]. Multiple data sets were obtained by changing the applied force rather than the orientation of the grasped object. They also presented a method for detecting the transition from a point contact to an edge contact, by monitoring the error term of the relative moment (i.e., the difference between the expected relative moment and the observed relative moment).

Mimura and Funahashi used active force sensing to determine the contact condition, without knowing the shape of the grasped object or the point of contact [48]. They present an iterative algorithm which first assumes a point contact, and then tries successively more complex contacts (soft finger, edge, and plane). Their method reduces the number of sensing data sets required and the number of equations. However, they do assume that the contact conditions remain unchanged during the identification process (i.e., no slip occurs).

McCarragher and Asada used an analysis of rigid body dynamics to identify the contact formation between objects [44]. Constraints were added to the equations of motion with Lagrange multipliers and a velocity constraint matrix. For each discrete contact formation, all possible quality states[1] were enumerated, where a quality state is defined as a unique combination of positive, negative, or zero values for each position, velocity, and acceleration term in the equation. Qualitative templates were then precalculated as the sequence of quality states that represented the type of motion expected to achieve the corresponding contact state. Using a similar analysis of the equations of motion, qualitative templates were also generated for the transitions between each contact state (i.e., the loss and gain of contacts). The qualitative templates were used to understand the force signals produced during the assembly process. By monitoring the force profile, with loss and gain of contacts, the sequence of contact formations could be determined.

McCarragher and Asada's qualitative reasoning approach provides a fast identifi-

[1]The term *quality states* was used in this reference; however, the intent is probably that of *qualitative states*.

cation scheme for the contact formations, although there are some practical problems. Threshold values must be determined to distinguish between large and small positive, large and small negative and zero qualitative states. Also, the force signals must be sampled fast enough to detect the sequence of quality states. (McCarragher and Asada used a sampling rate of 300 Hz.) In addition, geometric models of the workpieces and the robot environment are needed to precalculate the templates.

Using this identification scheme, McCarragher and Asada implemented a discrete event controller for use in autonomous assembly operations (e.g., peg insertion) [45]. The assembly process was modeled as a discrete event dynamic system. Each assembly task was described as a sequence of discrete events (i.e., a sequence of contact formations), using a petri net. The controller detected the current discrete event and generated the velocity trajectory to achieve the next desired discrete event.

More recently, refinements have been made to McCarragher's qualitative reasoning method of identifying contact formations [61]. Instead of using threshold values to distinguish between contact states, discriminant functions are used. Force data is first partitioned into classes using a clustering algorithm. Linear discriminant functions are then determined from the classified data.

As an alternative to using geometric information, Hovland and McCarragher proposed using fast Fourier transforms to capture the dynamic nature of a contact change and Hidden Markov Models to model the information [29]. Although they achieved a high success rate (98%), this approach is not easily accomplished in real-time.

Another method which does not require geometric information was demonstrated by Hara and Yokogawa [23]. Fuzzy sets were used to model contact forces and recognize the contact formation. However, only a small number of contact states for one case were considered here. No attempt was made to provide a generalized methodology for applying the approach to other workpieces or contact tasks.

Telerobotics systems have used force signals to provide autonomy at the robot site. In [27], Hirzinger et al describe a system where sensor signals (including force) were simulated at the operator's local station. The robot's controller then moved the robot in an attempt to match the real sensor signals to the simulated signals. Funda et al use a similar strategy, which they called teleprogramming [21]. The robot environment was simulated at the operator's station (including simulating the contacts). Based on the simulation, symbolic commands were issued to the robot (e.g., slide, rotate, guarded move). Cho et al extended the teleprogramming concept by using the desired contact formation as the symbolic link between the operator's control station and the robot's controller [10]. However, because of the difficulties in identifying the contact formation, only a face-to-face mating task was implemented.

B. Understanding the Human

Many studies have been done, in an effort to better understand the human. Only a few will be discussed here, which are particularly relevant to this dissertation. In [67], Stansfield discusses a human model of grasping in relation to vision vs. haptic sensing. The human primarily uses vision to seek and grasp an object until contact is made; at that point, the haptic sense dominates the visual sense. If we apply this reasoning to the use of sensing in robotics, it provides a rationale for using force signals in the control of contact operations.

In an effort to better understand how humans use force sensory perception, McCarragher investigated the force signals generated by a human demonstrator when performing a complex assembly task [43]. He looked at the force profiles and identified the points of loss and gain of contacts, using the analysis of dynamics described in the previous section. A justification is presented for using the contact formation as a qualitative discrete event in an assembly process, thereby segmenting the human-performed task. As a continuation of this effort, McCarragher also analyzed the data using stiffness control and impedance control, to see whether either could provide an adequate model for a human's assembly skills [42]. He concluded that neither provides a complete model of human skill; however, until an adequate model is determined, using a strategy which maps sensory input to commanded output is viable for getting robots to imitate human motions.

C. Robot Programming by Demonstration

This section provides a review of the previous attempts at programming robots using human demonstration. A description of the very early systems is included for a historical perspective. The more recent work is separated into three categories: systems not using force information (primarily vision-based systems), simulation- based systems, and force-based systems. The common goal in this research is to create an easy-to-use programming method that can be used by non-experts. Researchers attempt to avoid the requirement of special expertise needed for textual programming and to overcome the combinatorial complexity problems found in automatic programming methods.

1. Early Systems

Early demonstration systems primarily used position information. The operator used a simple hand controller to guide the robot through a sequence of points. The robot recorded the points and repeated the sequence. The responsibility for precise positioning rested on the human operator, and the program often failed if parts were not in those precise positions. The XPROBE system [69] introduced simple sensing strategies which could compensate for minor deviations. However, the limited use of force signals prevented the programming of complex tasks. Also, the operator interface used a cumbersome combination of buttons and keyboard input. The LARTS system [58] integrated a task-level language and teaching via teleoperated manipulation. However, the lack of support for force sensing again limited the practical use. Also, trying to demonstrate a complete, perfect program with a teleoperator can be exhausting and difficult for the human operator.

2. Demonstration Systems without Force

Recent attempts use passive modalities to observe the human performing tasks in a natural setting. Kang and Ikeuchi use vision to observe human hand movement, while attempting to determine the segmentation of a task [33]. Kuniyoshi et al use vision to extract a high-level task strategy in their 'learning by watching' technique [76]. Similarly, Tung and Kak use a dataglove and Polhemus sensor to monitor the human hand movement, and deduce an assembly plan [71]. Assumptions are made about where and how the object is grasped. Finally, Tso and Liu provide a wrist device for the human to hold an object. The wrist has LED markers which are tracked by a vision system [70]. In all of these demonstration systems, the main focus is on learning a high-level plan rather than on learning sensorimotor skills.

Resolution limitations in the sensing make it difficult to extract the fine motion plan needed to perform contact operations such as assembly, especially when uncertainties exist in the position and orientation of objects. Paul and Ikeuchi have attempted to overcome these resolution limitations by using thresholds to make assumptions about contact points, and then computing the motion path in C-space [55]. This necessitates the use of geometric models which increases the computational load. Also, the motion plan is still based on position information, and, in order to succeed, requires accuracy in the robot motion and in obtaining object positions. The learned plan would be more robust if force references were calculated and the execution of the plan was performed using force control; however, this has not been done.

In general, it is the view here that the demonstration systems discussed in this section are complimentary to the methodology presented in this dissertation. The task strategies learned with these systems would be more robust if they were combined with force-based skills.

3. Simulation-Based Systems

An alternative to using a real demonstration environment is to use a simulated one. A virtual environment can be created and presented to the human teacher using 3D graphics, allowing the demonstration to be performed in an interactive, simulated world. An example of such a system is presented by Ogata and Takahashi in [52]. The goal of this system is to learn a high-level assembly task strategy. As such, it is similar to the vision-based systems described in the previous section and has the same limitations.

In contrast, Onda et al present a simulation environment designed to learn fine motion assembly planning strategies by extracting a sequence of contact states (i.e., contact formations) [54]. The contact formation is determined by the collision of the geometric models that occurs as the demonstrator moves the virtual workpieces. The advantage to this approach is that it creates a symbolic representation of the assembly process. The operator can also refine the plan by pruning out contact formations deemed unnecessary. However, problems can still occur when the plan is executed in a real robot environment, unless the workpiece positions are known accurately. The sequence of contact formations can provide a symbolic link between the learned task (in the virtual environment) and the execution of the task (in the real world). Manipulation skill primitives which move the robot from one contact formation to another are offered as a solution to this problem [49], but these primitives are also based on position. Again, incorporating force-based primitives would result in more robust execution.

4. Force-Based Systems

One of the pioneers in creating force-based programming by demonstration systems has been Haruhiko Asada. Let us first review the systems designed for non-assembly tasks. In [3] , Asada and Asari describe skill transfer of a grinding task. The human holds a grinding tool, which contains a force sensor, for measuring force exerted on the surface, and a potentiometer, for measuring the speed of the grinding wheel. Forces, positions, and wheel speed are monitored, and impedance control parameters are learned for the 1 degree of freedom (DOF) grinding task.

Asada and Yang look at the application of deburring in [5]. Sensor signals are processed using pattern recognition methods to accomplish sensory perception (i.e., to recognize the state of the process). Control laws are extracted in rule-form: IF (signal pattern) THEN (control action). In [38], Liu and Asada attempt to learn how the human changes the control strategy in response to changes in the task process. The human demonstrates a deburring task, with the force sensor mounted under the fixture. The deburring task is modeled as a metal-cutting process; impedance control parameters are used to model the control strategy. A neural network is used to find the nonlinear mapping between the task process parameters and the control parameters.

Asada has also researched systems designed for assembly-type tasks. In [4], Asada and Izumi use teaching data to generate a program for a palletizing task, in which an object is placed at the corner of a rectangular box. The human directs the robot with a hand controller, demonstrating control modes and trajectories for hybrid position/force control. Constraint surfaces are restricted to being planes and must be perpendicular to each other. No rotational movement is supported, and the system is not generalized beyond this simple task.

Kosuge, Fukuda and Asada [37] use the same task and extract two levels of control strategies from the teaching data – (1) a high-level sequence of discrete states (i.e., contact formations), and (2) the compliant motion strategies used by the operator to move between discrete states (modeled as damping control). The sequence of contact formations is generated by using geometric models and the method developed by Hirai and Asada [25]. This same technique provides a strategy for recognizing the contact formations from the monitored force signals(using detailed geometric information). Note that the sequence of contact formations is known a priori.

Another system based on the discrete events (referenced here as the hybrid dynamic model) is that proposed by Hovland, Sikka, and McCarragher in [30]. They use a discrete event controller and assume the existence of a process monitor that detects and identifies changes in contact (such as that in [29]). This reference focuses on the control part of the skill, using a Hidden Markov Model to model the human's motions. The Markov states correspond directly to the discrete states (i.e., contact formations). The goal is to learn the velocity command which allows the robot to achieve the next desired discrete state. The advantage with this approach is that different task strategies can be demonstrated and learned within the same framework. If one strategy fails because of position or orientation uncertainties, another strategy can be determined on-line. The disadvantage is that the HMM requires the velocity commands to be discretized, which may be impractical if extended beyond the planar task used here.

In contrast, another approach is to learn compliant motion strategies or trajectories for a task. Wang et al use a hand controller to demonstrate an assembly task and learn a compliant motion program [72]. Delson and West learn a compliant motion program by using a virtual trajectory to model the human demonstrator's task strategy [13, 14]. The limitation with both of these approaches is that only one strategy is learned, and the task execution will fail if the orientation of the environment is significantly different from that at demonstration time. (Indeed, neither of these works address tasks involving rotational movement.) Delson and West address this problem in part by re-orienting the environment and repeating the demonstration using the same task strategy [15]. However, the success of the task execution is still dependent on the environment not varying more than it did between the various demonstrations so that the demonstrated task strategy can succeed.

In another category of force-based programming by demonstration systems, neural networks are used to model the compliant motion of assembly tasks. Asada was the first to propose using neural networks as a means of learning the nonlinear mapping of measured force signals to corrected motion, in terms of velocity commands [2]. His generalized network had three layers. The first layer mapped force signals to individual contact points on the workpiece. The second layer provided a mapping to discrete contact states (i.e., contact formations). The third layer provided the final mapping to the velocity command outputs. Asada demonstrated the approach on a planar assembly task, using simulated data to train the neural network.

This neural network approach was also implemented by Whalen for a planar edge-mating task [74]. Whalen showed that it was successful when using simulated training data, but he did not find it to be successful when using actual human demonstrated data. One possible reason for its failure might be that human demonstrators can generate very inconsistent motions unintentionally. As a result, the training sets contained greatly varying output data for similar inputs, which are particularly difficult to use when training a neural network. Examples of this inconsistent motion will be presented later in the dissertation.

Koeppe proposed a modified neural network for learning compliant motion strategies from human demonstrations [35]. Velocities, as well as force signals, are inputs to the network, in the form of fuzzy variables (e.g., small, medium, large or negative, zero, positive). Koeppe showed that velocities should be included as inputs to solve the correspondence problem (i.e., delay) between the human's haptic perception and the motion correction. Koeppe's execution results in a simulated environment have been promising. However, this approach also requires that the execution environment is identical to the demonstration environment.

Kaiser et al have also used neural networks for sensorimotor skill acquisition from human demonstrations [32]. To overcome unintentional inconsistency demonstrated by the human teacher, the training data is first preprocessed. Sample (input, output) pairs are removed if the output value is below a threshold. Also, a smoothing algorithm is applied. In addition, reinforcement learning is used for skill refinement after the network has been trained with the demonstrated data.

Finally, an example of programming by demonstration using non-rigid parts has also been tried. In [26], Hirai et al attempted a skill transfer for inserting a deformable hose on a plug. The operator used a tool equipped with a force sensor and a Polhemus tracker (for position and orientation information) to hold the hose. The goal was to learn a task strategy for motion and control based on sensory feedback. Poor performance resulted from a direct transfer of skill. Better results were achieved after the humans (manually) did off-line analysis of the collected data.

CHAPTER IV

METHODOLOGY

In this chapter, the main ideas are presented for transferring assembly skills to robots. Ideas will be drawn from previous work, but in many cases, we will take a look at the problem (and solutions) from a new perspective. Concepts may be united and extended in new ways. In doing so, it is important to keep the focus on the overall purpose and goals of the research, which are to make robots easier to use and re-program, thereby making them more functional. Consequently, we will begin by looking at user interface issues and present a model of the human-robot interface. We will then look at how assembly skills can be modeled and draw from that model the requirements for transferring skills to robots.

A. Anthropomorphic User Interface

To meet our stated goal for a natural interface for the human user, we will model the human-to-robot interface as a human-to-human interface. In this sense, the interface is anthropomorphic. The transfer of skills from people to robots (in a natural way) can be accomplished using the programming by demonstration method discussed in Chapter III. In this method, the human demonstrator acts like a teacher to the student robot. Therefore, a useful model for a human teacher to robot student interface would be a teacher-student model.

One such model was proposed by Collins and Stevens [11], as the result of an extensive study to understand how good teachers teach. Shown in Figure 1, the model is based on Collins and Stevens's theory of teaching, which uses an inquiry style. The inquiry teaching method is an interactive style, where the teacher engages the student in a dialogue and guides the student to discover new concepts. A key element of their theory is the control structure used by teachers in establishing the dialogue. To a large degree, this is determined by the teacher's model of the student's knowledge. The teacher must understand the model, know what is missing, and be able to supply the missing information. In this way, the questions asked and examples presented by the teacher are determined by what the teacher thinks the student already knows. The teacher updates his model continuously, as the teaching session progresses and the student's knowledge level develops. In this case, of course, the teacher must determine an estimate of the student's knowledge by the conversational dialogue between them.

The teacher-student model described above can be used as a basis for our human teacher-to-robot interface. The work done by Collins and Stevens emphasizes the importance of providing a representation of the student robot's knowledge. This representation will be in the form of a skill model, which will be presented in detail in the next section.

With this in mind, we present a model of the teacher-robot interface, as used in the demonstration teaching method. As shown in Figure 2, direct communication paths have been added between the robot's skill model and the teacher's copy of the robot's skill model. This can easily be done as a computer-to-computer transfer, and provides

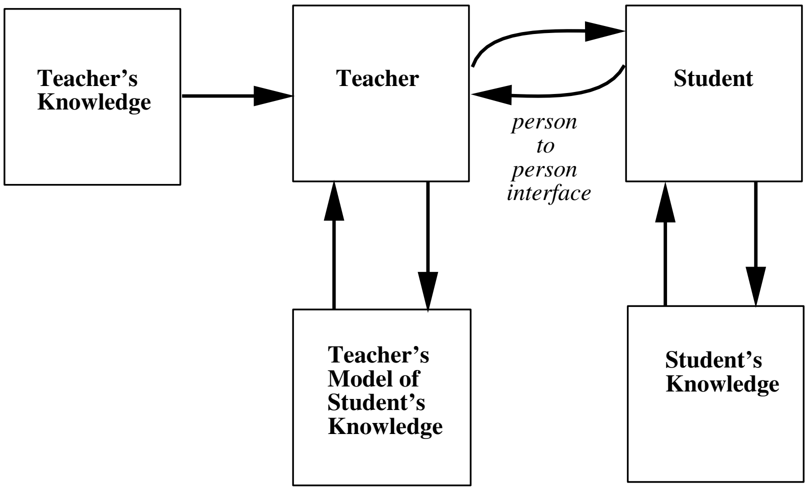


Figure 1. The Teacher-Student Interface in the Inquiry Teaching Method

a method of determining exactly what the robot is learning as the demonstration is performed.[1] The human teacher can also receive feedback from the robot student by using the normal human sensory system. In the case of teaching generalized assembly skills, a good (and probably necessary) sensory feedback is to be able to feel the forces felt by the robot (e.g., force reflection through a hand controller device).

The interfaces shown in Figure 2 (the person-to-robot interface and the computer-to-computer interface) are examples of providing feedback to the user. The advantages of feedback in a user interface have been observed by numerous experts (for example, see [18]. Indeed, Nielsen includes feedback as one of his 10 usability principles which should be followed by all user interface designers: "The system should always keep users informed about what is going on, through appropriate feedback within reasonable time." [51]. Also, Nakamura et al observed improved teaching of a robot

[1]The model has been defined using a generalized distributed systems approach, where the robot's skill model and the teacher's copy of the model may reside on different computers.

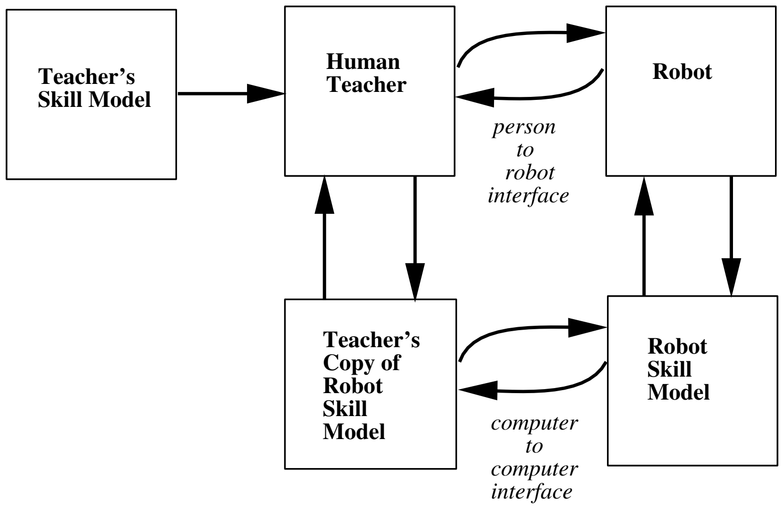


Figure 2. The Teacher-Robot Interface in the Demonstration Teaching Method

deburring skill, when the human teacher received feedback on the robot's status [50].

In summary, the student's skill model and the teacher's knowledge of the student's model are crucial elements in the teaching process. The teacher must be able to correct any misconceptions in the student's model, which implies an interactive, iterative teaching environment. We believe this is the necessary approach to achieve efficient, effective teaching of robot skills.

B. Skill Model

An important goal of this dissertation is to find a concrete model of a skill, represented in a mathematical form, which can be used to successfully achieve a robust, reusable strategy for assembly tasks. In this section, we present such a model, with the purpose of investigating its usefulness and applicability for modeling assembly skills.

The skill model used for the transfer of assembly skills to robots should meet the requirements of both the human teacher and the robot student. That is, it should provide an interface that both can easily understand. One such model is the hybrid control model proposed by Brockett [8], in which symbol processing interacts with signal processing. The hybrid control model considered here provides a mathematical framework for modeling motion control systems that are guided by symbolic representations as well as real-valued functions. As we will see, the symbolic representation will facilitate the interactive teaching environment discussed in the previous section.

The formal description of the hybrid control model will be based on Brockett's hybrid control models, as discussed in [8], with special application towards modeling force-based assembly skills. The model, shown in Figure 3, consists of four parts: (1) the robot, (2) the robot controller, (3) the state classifier, and (4) the supervisory controller. The interaction between the robot controller and the robot is based on functions of time. Symbolic processing is done at the supervisory controller level; both its inputs and outputs are in the form of symbols. The input to the supervisory controller is generated by the state classifier, which converts a time-varying signal into a symbolic representation of a qualitative state. The rate at which this occurs is dependent on an event trigger, which indicates that the system has changed to a new qualitative state.

The model can be described by a set of equations. Let x be the position of the robot end effector, and y be the reading from the robot's force sensor, and u be the force control command. Then x, y, and u are vector-valued functions of time, which take on values in the subsets of Cartesian spaces X, Y, and U, respectively.

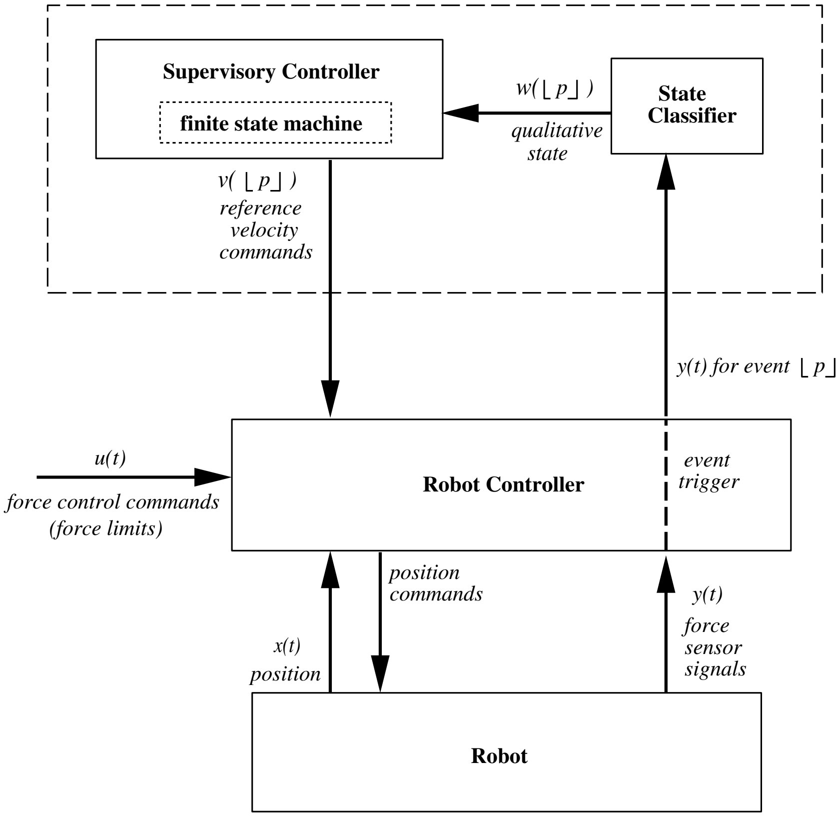


Figure 3. The Hybrid Control Model of an Assembly Skill

Following Brockett's convention, we will let p be a monotone increasing triggering signal, a function of time, and write $\lfloor p \rfloor$ to denote the largest integer less than or equal to p. Then $\lfloor p \rfloor$ may be viewed as an event index. Let $v(\lfloor p \rfloor)$ be the symbolic controller command (representing the reference velocity command), and $w(\lfloor p \rfloor)$ be the symbolic input to the supervisory controller (representing the qualitative state in the assembly process). The symbolic variables, v, and w take on values in the sets V and W, respectively, where V is the set of reference velocity commands and W is the set of force-based qualitative states. Let δ be the sampling time period for the robot controller and k be the integer sampling increment. The following equations describe the model:

$$
\begin{aligned}
x(k\delta+\delta) &= a(v(\lfloor p \rfloor),\ x(k\delta),\ y(k\delta),\ u(k\delta)) \\
y(k\delta) &= c_e(x(k\delta),\ u(k\delta),\ v(\lfloor p \rfloor)) \\
p(k\delta+\delta) &= r(y(k\delta)) \\
w(\lfloor p \rfloor) &= h(y(k\delta))
\end{aligned}
$$

As shown in the equations, the position x is dependent on the symbolic controller command and the values for the position, force sensor reading, and force control command at the last sampling period. The force sensor reading y is dependent on the position, the force control command, and the symbolic controller command. The subscript e on c emphasizes that the force sensor readings will also depend on the robot's environment. The rate equation r which describes the increase in the triggering signal p acts as an event trigger and indicates when the system has changed to a new qualitative state. The rate equation is constrained so that no more than one event may occur during the sampling period δ. The qualitative state w is dependent on the force reading.

Brockett's equations provide a mathematical framework for hybrid control models; however, as given above, it is still incomplete. Missing pieces include the implementation of the event trigger, as well as the method used by the state classifier to convert the time-varying force sensor signals into qualitative states. These will be discussed in subsequent sections of this chapter. Another missing function is the mapping of the qualitative states, w, into symbolic controller commands (specifically, reference velocity commands, v). A finite state machine is used for this purpose and will be described in more detail later in the section.

The robot controller regulates the robot using two control loops. At the lowest level, there is a position control loop. An outer force control loop is used to implement hybrid position/force control, using force signals for feedback. Ideally, the force control commands should be dependent on the qualitative state of the assembly process, meaning that the force control commands would change whenever the qualitative state changes. However, to simplify this initial work, the force control commands are modeled as pre-established force and moment limits for each axis of the force sensor. If a force along any axis exceeds the threshold value, then the robot controller attempts to control the force to a set value. The force and moment limits remain constant for an entire assembly task.

The rationale for using force sensor signals is based on the human-performed assembly operations. Our goal is to learn low-level sensorimotor skills that can be used to achieve desired assembly tasks. Assembly operations involve the manipulation of one or more movable objects, which make contact with the environment. As observed in Chapter 3, when people perform assembly tasks, they use vision until the grasped object contacts the environment. From that point forward, the haptic sense dominates. In this research, we are focusing on what happens after contact is made. Following the human analogy, force sensor signals are used as the robot's haptic sense, and thus, they are used as the robot's time-based output signals as applied to the hybrid control model.

The force sensor signals from the robot are mapped into discrete robot symbols, termed single-ended contact formations (SECF), which describe qualitatively how a grasped object is touching its environment. The SECF is used as the qualitative state that drives the assembly skill. The mapping is performed by the state classifier every time the event trigger detects a change in SECF state. The characterization and classification of SECF's will be described in more detail in subsequent sections of this chapter.

The supervisory controller uses a finite state machine (FSM), represented as a directed graph, to drive the assembly skill. The states stored in the FSM are the SECF's which describe the qualitative condition in the assembly process. The FSM stores the sequence of SECF's which are used in the assembly skill. The skill model also includes the controller output function, which generates reference velocity commands based on the current SECF and the desired sequence of SECF's. The velocity commands describe how to transition from the current SECF to the next desired SECF. Note that the reference velocity commands are relative commands, which instruct the robot to move in a specified direction. They are not based on absolute position. In this way, the assembly skill model is independent of the robot's absolute position.

Because we are not making any assumptions about the position and orientation of the robot's environment,[2] we cannot guarantee that the desired goal condition will be reached. This is similar to other sensor-based strategies used for robotic control in unstructured environments (e.g., [9]). However, we can increase the probability for success, in spite of position uncertainties, by incorporating more than one SECF sequence into the FSM. A skill may, in fact, contain several SECF sequences that will result in a successful task completion. Multiple sequences can be combined into the FSM and used to provide robustness in the skill execution. The combination of sequences can be used to achieve a sequence that was not included in the demonstration process but, in fact, still provides successful completion.

As modeled here, the learning of an assembly skill involves the learning of three functions: (1) the mapping of force sensor signals to SECF's (2) the sequence (or sequences) of SECF's and (3) the transition velocity commands which move the robot from the current SECF to the next desired SECF. It is important to reiterate that the SECF's are not based on position but rather are based on force signals and provide a symbolic representation of how the grasped object is touching its environment. This

[2]Indeed, the assumption here is that we do *not* know the position and orientation of objects in the environment.

provides our desired event-based approach which enables us to overcome uncertainties in the environment, especially position uncertainties.

The long-term goal in skill acquisition would be to learn an entire skill with as few demonstrations as possible. However, as a first step towards investigating the applicability of the hybrid control model, an iterative approach has been used. Supervised learning is used to first learn the mapping of force sensor signals to SECF states. This is accomplished by having the human teacher demonstrate each SECF while force data is collected. The accumulated force data for all of the SECF's is used to train the state classifier. The human teacher then demonstrates a skill, and the classifier is used to extract the sequence of SECF's and transition velocities which comprise the rest of the skill.

One advantage of using this two-step approach is that we have a partial skill model when the task demonstration is performed, and this information can be used during the demonstration. During the demonstration, feedback can be provided to the human teacher by classifying and displaying the SECF sequence in real-time. The symbolic representation of the SECF is useful for providing meaningful and easily recognizable feedback to the human teacher. This additional feedback can be used to increase the efficiency of the teaching process, because the teacher knows what sequence is being learned as the demonstration is performed. If the sequence is not the desired one, then corrections can be made immediately.

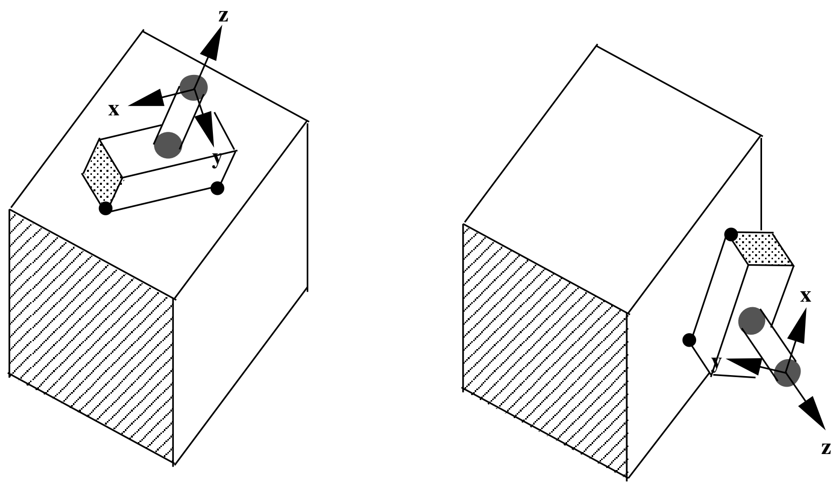


Figure 4. An Example of Two Contact Formations which Belong to the Same Class of Single-Ended Contact Formations

C. Force Sensory Patterns of Single-Ended Contact Formations

1. Characterization of the Single-Ended Contact Formation

The contact formation [16] provides a qualitative description of how 2 or more objects make contact with each other (e.g., edge 1 of the grasped object touches side B of the environment). As such, the possible contact formations depend on the relative size and shapes of the objects, reflecting the geometric relationship. In contrast, a single-ended contact formation (SECF) provides a one-sided description of how the grasped object touches its environment (e.g., edge 1 of the grasped object touches any side in the environment). It does not attempt to distinguish between similar contact points in the environment. Figure 4 shows an example of two contact formations, in which an edge of the movable object is touching different sides of a fixed object in the environment. Because both contact formations involve the same edge of the movable object, they provide examples of the same SECF.

Our goal is to recognize the contact state from force sensor data only, without using geometric knowledge or position information. It is not possible to determine the complete contact formation from force data alone. However, it may be possible to determine a one-sided description of the contact state, using force signals from the robot force sensor. From the human demonstrator's point of view, this is the haptic information needed. A human performing an assembly operation would use vision to position an object close to the desired point of contact and then use haptic senses to complete the task. In this situation, there is no need to determine the complete contact state from haptic senses alone. We wish to provide the same type of feedback to the robot. Thus, the SECF provides the necessary qualitative state required for the hybrid control skill model.

2. Sensory Patterns

To investigate the force sensory patterns, force sensor data was collected for a set of known SECF's. For the sample data included here, the test block shown in Figure 5 was used. This is a small, plastic block. In contrast to a finely machined block, it is not perfectly symmetric and has edges that are not necessarily straight nor sharp. To provide a controlled setting, the block was held in a stationary configuration by the robot arm while a flat plate (the environment) was moved manually. Signals were collected from the force sensor mounted on the robot arm.

The remaining figures in this section show the sample force and moment vectors collected for the test SECF's. In each case, the vectors shown represent the raw data

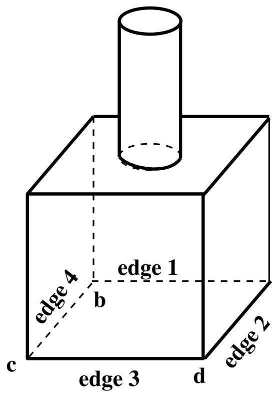


Figure 5. Test Block Showing 4 Edge and 4 Vertex SECF's

from the force sensor, as measured in the sensor frame. Figures 6 and 7 show the force and moment vectors which comprise the Vertex b and Vertex d data sets. Figures 8 and 9 show the corresponding sets for the Edge 2 and Edge 4 sets.

As shown in the figures, the vectors in each data set form a cone-shaped pattern. The shape of the cone is dependent on the geometric shape of the object and reflects the motion constraints existing between the grasped object and its environment. Other factors affecting the shape of the pattern include friction, uncertainty due to sensor noise, and errors that may occur in the data collection process. Figures 10, 11, 12, and 13 provide illustrations of the differences in the cone shapes, by showing projections of the force and moment vectors onto the X-Y plane.

The cone-shaped patterns shown in the figures are consistent with the theoretical discussions provided by Hirai and Asada in [25][3] and Farahat et al in [19], which are based on geometric relationships. However, in the theoretical, geometric approach, it is difficult to account for all of the real-world effects, such as friction, sensor noise,

[3]This discussion does not include the effects of friction.

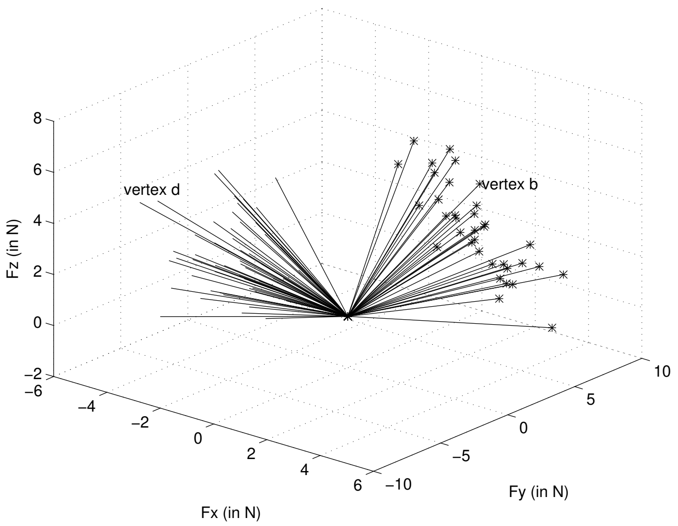


Figure 6. Force Vectors of 2 Vertex SECF's

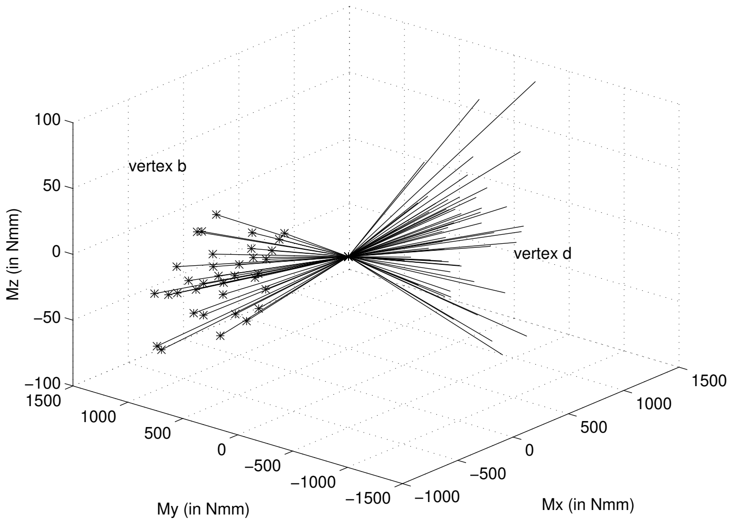


Figure 7. Moment Vectors of 2 Vertex SECF's

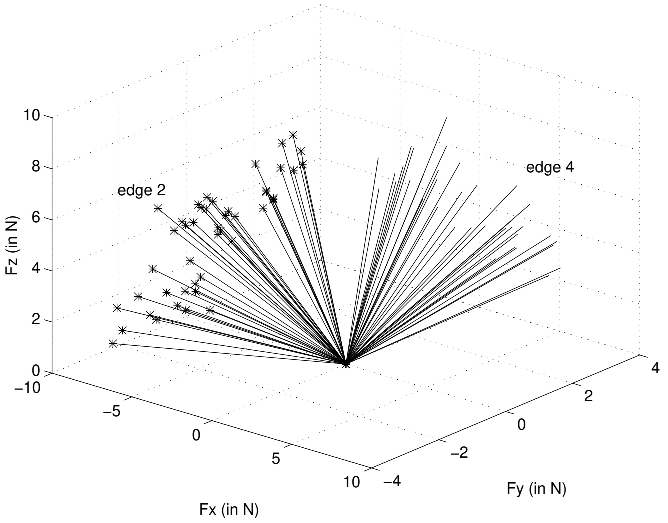


Figure 8. Force Vectors of 2 Edge SECF's

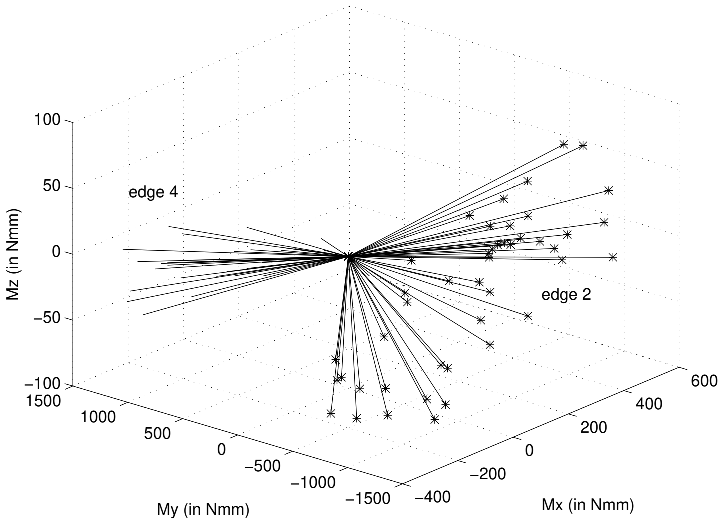


Figure 9. Moment Vectors of 2 Edge SECF's

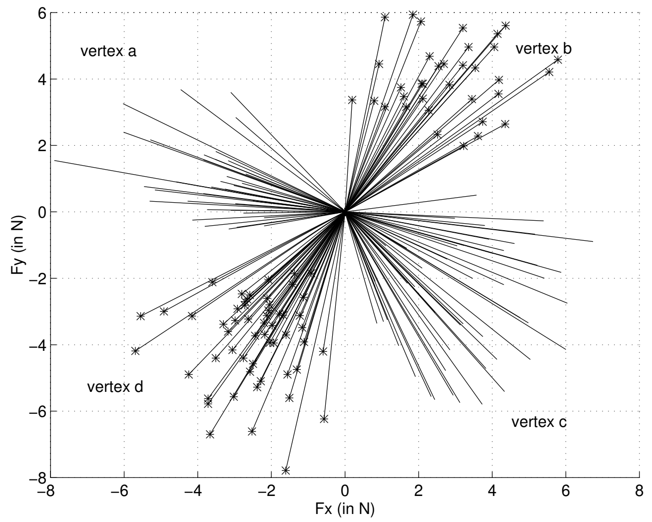


Figure 10. 2D Projection of the Force Vectors for 4 Vertex SECF's

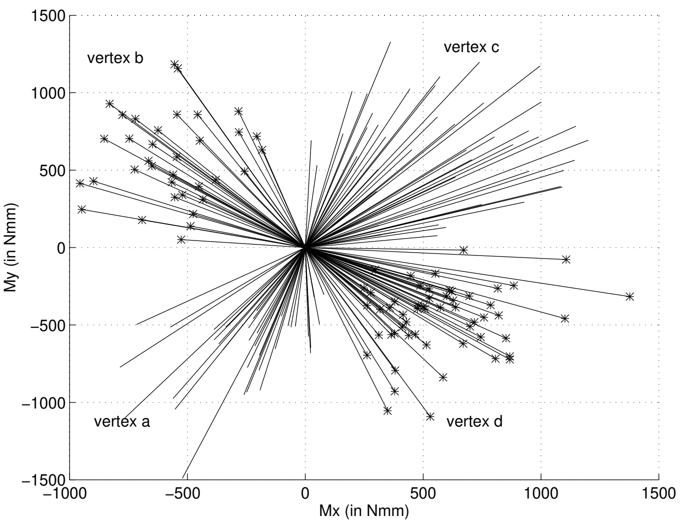


Figure 11. 2D Projection of the Moment Vectors for 4 Vertex SECF's

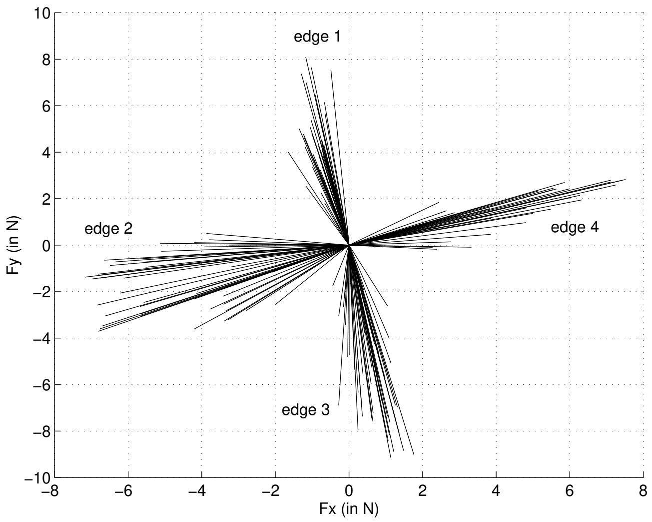


Figure 12. 2D Projection of the Force Vectors for 4 Edge SECF's

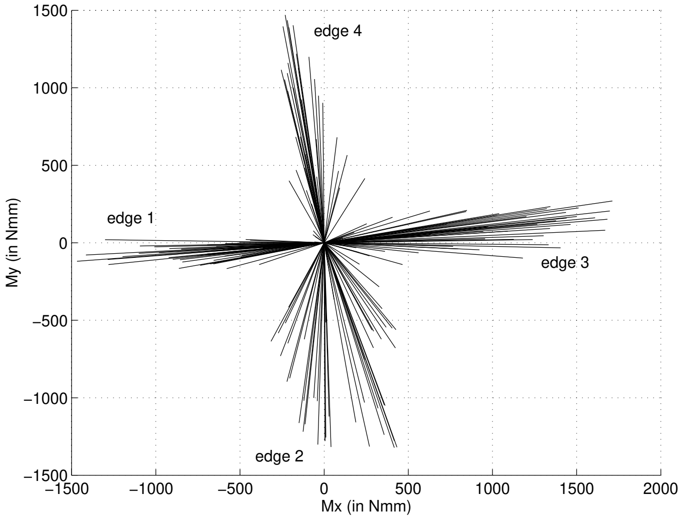


Figure 13. 2D Projection of the Moment Vectors for 4 Edge SECF's

and imperfection in the workpieces. By incorporating a sensor-based approach, we hope to account for these difficult-to-model effects.

D. Classification of Single-Ended Contact Formations

The cone-shaped sensory patterns shown in the previous section can be used to identify SECF's from force sensor signals. If the shape characteristic of each cone can be captured, then the direction of the force and moment vectors can be used to determine in which cone (and, therefore, which SECF) a particular signal resides.

To capture the direction, the data vectors are normalized, yielding a projection onto a unit sphere, as shown in Figure 14. Let f_x, f_y, and f_z be the force components of the sensor signal. Then, the normalized force components, f_{xn}, f_{yn}, and f_{zn}, are

$$f_{in} = \frac{f_i}{|f|} \qquad \text{for } i{=}x,y,z$$

where $$|f| = \sqrt{f_x^2 + f_y^2 + f_z^2}$$

Similarly, let m_x, m_y, and m_z be the moment components of the sensor signal. Then, the normalized moment components, m_{xn}, m_{yn}, and m_{zn}, are

$$m_{in} = \frac{m_i}{|m|} \qquad \text{for } i{=}x,y,z$$

where $$|m| = \sqrt{m_x^2 + m_y^2 + m_z^2}$$

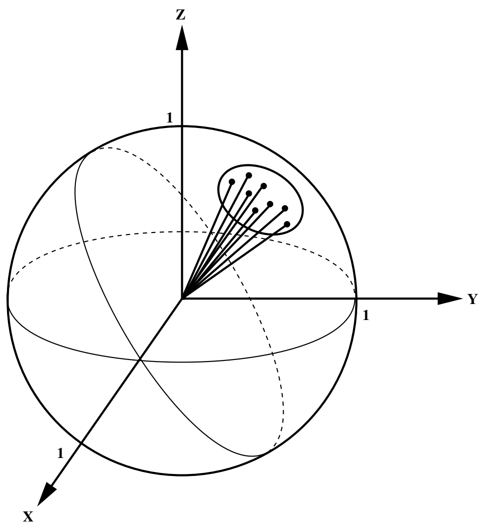


Figure 14. The Projection of a Set of Vectors onto a Unit Sphere

After the data vectors have been normalized, the resulting data sets look like clusters projected onto the unit sphere. The clusters have different shapes and sizes, reflecting the shape and size characteristics of the cone patterns.

In the transferring of assembly skills from human demonstrators to robots, we must also have a way of determining whether any contact is being made between the grasped object and its environment. For this purpose, an additional class, called C_0, is added for the condition where the grasped object is not contacting the environment. The force magnitude, $|f|$, is used to identify this class. For the time being, let us assume that we have a perfect force sensor. In that case, if the force magnitude is zero, then no contact is being made. If the force magnitude is nonzero, then contact has been established between the grasped object and its environment. This assumption is provided initially to better illustrate the logic used within the state classifier. In the next section, the assumption of a perfect sensor will be discarded, and we will present a precise definition of what is meant by zero and nonzero force magnitude.

With this in mind, a conceptual overview of an SECF classifier is proposed. As a high-level structure, the classifier may be viewed as one logical rule per class, which describes the mapping between sensory data and class. The rule for class C_0, the *no contact* class, uses the force magnitude as shown below:

if	*force magnitude*	is *zero*
then	*class*	is C_0

The rules for the remaining C_i classes have the following form:

if	*force magnitude*	is *nonzero*	and
	force	is in the F_i set	and
	moment	is in the M_i set	
then	*class*	is C_i	

where *force* and *moment* are the measured sensor vectors and F_i and M_i represent the set of possible force and moment vectors for SECF class C_i.

Expansion and implementation of the logical rules will be presented in the form of two classification methods. The first is based on fuzzy logic. The second uses a neural network. Both use the measured force sensor signals to identify the SECF classes.

1. The Fuzzy Classifier

In this section, we describe a state classifier which is based on fuzzy logic. The logical rules described in the previous section are expanded into fuzzy rules using the force magnitude and the 6 normalized sensor signals as input features. The normalized sensor signals are used to capture the direction of the force and moment vectors and to determine whether they lie within the appropriate projected cluster. Fuzzy sets are used to deal with the consequences of an imperfect force sensor (noise) and other uncertainties found in the real world.

The set of possible signals for each feature is represented as a fuzzy membership function. For a particular input signal, the degree of membership is calculated for each feature and used to determine the class to which the input signal belongs. The zero force magnitude membership function used for identifying class C_0 is represented by $\tilde{Z}ero$. The nonzero force magnitude is represented by the complement, $NOT\ \tilde{Z}ero$. The membership functions of the normalized force and moment components for each contact class C_i are represented by $\tilde{F}x_i$, $\tilde{F}y_i$, $\tilde{F}z_i$, $\tilde{M}x_i$, $\tilde{M}y_i$, and $\tilde{M}z_i$.

The rule for the no contact class C_0 is shown below:

if $|f|$ is $\tilde{Z}ero$

then $class$ is C_0

All of the remaining rules have the following form:

if $|f|$ is $NOT\ \tilde{Z}ero$ and

$$
\begin{array}{lll}
f_{xn} & \text{is } \tilde{F}x_i & \text{and} \\
f_{yn} & \text{is } \tilde{F}y_i & \text{and} \\
f_{zn} & \text{is } \tilde{F}z_i & \text{and} \\
m_{xn} & \text{is } \tilde{M}x_i & \text{and} \\
m_{yn} & \text{is } \tilde{M}y_i & \text{and} \\
m_{zn} & \text{is } \tilde{M}z_i & \\
\text{then } \textit{class} & \text{is } C_i &
\end{array}
$$

For each proposition in the antecedent (e.g., f_{xn} is $\tilde{F}x_i$), the grade of membership is calculated (e.g., the degree to which f_{xn} belongs in the Fx_i set), yielding a number in [0,1]. The membership grades for each proposition are combined using a fuzzy conjunction operator, and the result is interpreted as the possibility (i.e., a fuzzy number in [0,1]) of this sensor signal being in the specified class. In the classification process, possibility values are calculated for each class; the highest possibility number represents the identified SECF class. This has the effect of resolving classification ambiguities by selecting the class with the nearest prototype vector, which would be at the center of the combined fuzzy membership functions. In practice, a threshold is also used to determine whether any classification can be made. In the current work, the threshold is empirically set to 0.5. If no possibility value is above the threshold, then the data vector is said to be unclassified.

The membership functions are generated automatically using supervised learning. Training data is acquired by demonstrating each SECF class and collecting the force

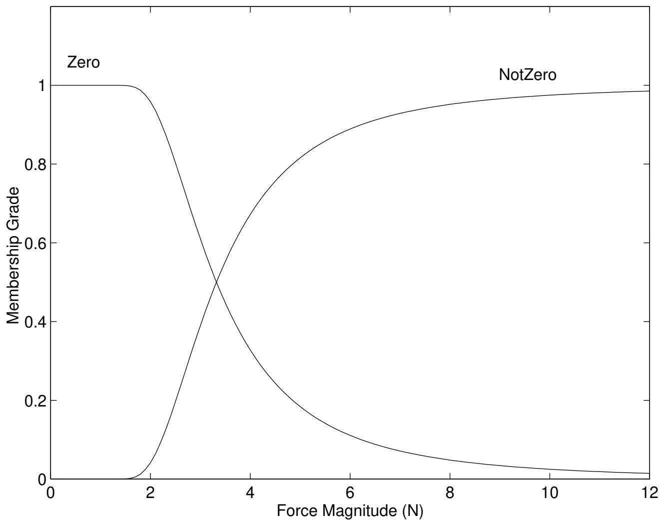


Figure 15. The Force Magnitude Membership Functions

and moment signals generated. For each contact class feature j, the mean, μ_j, and standard deviation, σ_j, are calculated. These are used as parameters to generate the membership functions, $\pi_j(r)$, as shown below:

$$\pi_j(r) = 1 - e^{-(3\sigma_j/|\mu_j - r|)^3} \qquad \text{for } j = Fx_i, Fy_i, Fz_i, Mx_i, My_i, Mz_i$$

Similarly, the membership functions for force magnitude, as shown in Figure 15, are given below:

$$\begin{aligned} \tilde{Z}ero(|f|) &= 1 - e^{-(3\sigma/|f|)^3} \\ NOT\ \tilde{Z}ero(|f|) &= 1 - \tilde{Z}ero(|f|) \end{aligned}$$

This membership function shape was chosen as a means of modeling the possible sensor distribution, as well as allowing for uncertainties in the sensor signals. Figure 16 shows the distributions for each sensor component with the corresponding membership function generated, for a sample SECF (edge2). [4] Two factors contribute to the range in possible force signals for an SECF. The primary factor is the way a workpiece can move and still maintain the desired contact formation. This is a function of the unconstrained degrees of freedom and the friction present. These movements will result in a change in the force components, but as long as the SECF is maintained, any of the force signals will still be valid for that class. A secondary factor contributing to the range is the sensor uncertainty (noise). The membership function shape is designed so that the flat portion of the curve will encompass almost the complete range of signals, which models the effect of the primary factor. As the curve drops off, this models the effect of the sensor uncertainty.

The fuzzy conjunction operator used to combine the membership functions is the Hamacher product [77]. The Hamacher product is defined for 2 fuzzy sets, as shown below:

$$H(\pi_A(r), \pi_B(r)) = \frac{\pi_A(r) * \pi_B(r)}{\pi_A(r) + \pi_B(r) - \pi_A(r) * \pi_B(r)}$$

It can be used in an iterative fashion, until the membership functions of all the features are combined. [5] As an alternative, the Hamacher product can be reduced to the following form:

$$H_n = \frac{1}{\sum_{i=1}^{n} \frac{1}{\pi_i} - (n-1)}$$

where n is the number of membership functions to be combined.

[4] The membership functions shown are scaled to match the histogram. The actual membership functions have a maximum value of 1.

[5] The order in which the features are combined is irrelevant, as the Hamacher product follows the fuzzy t-norm requirements for commutativity and associativity.

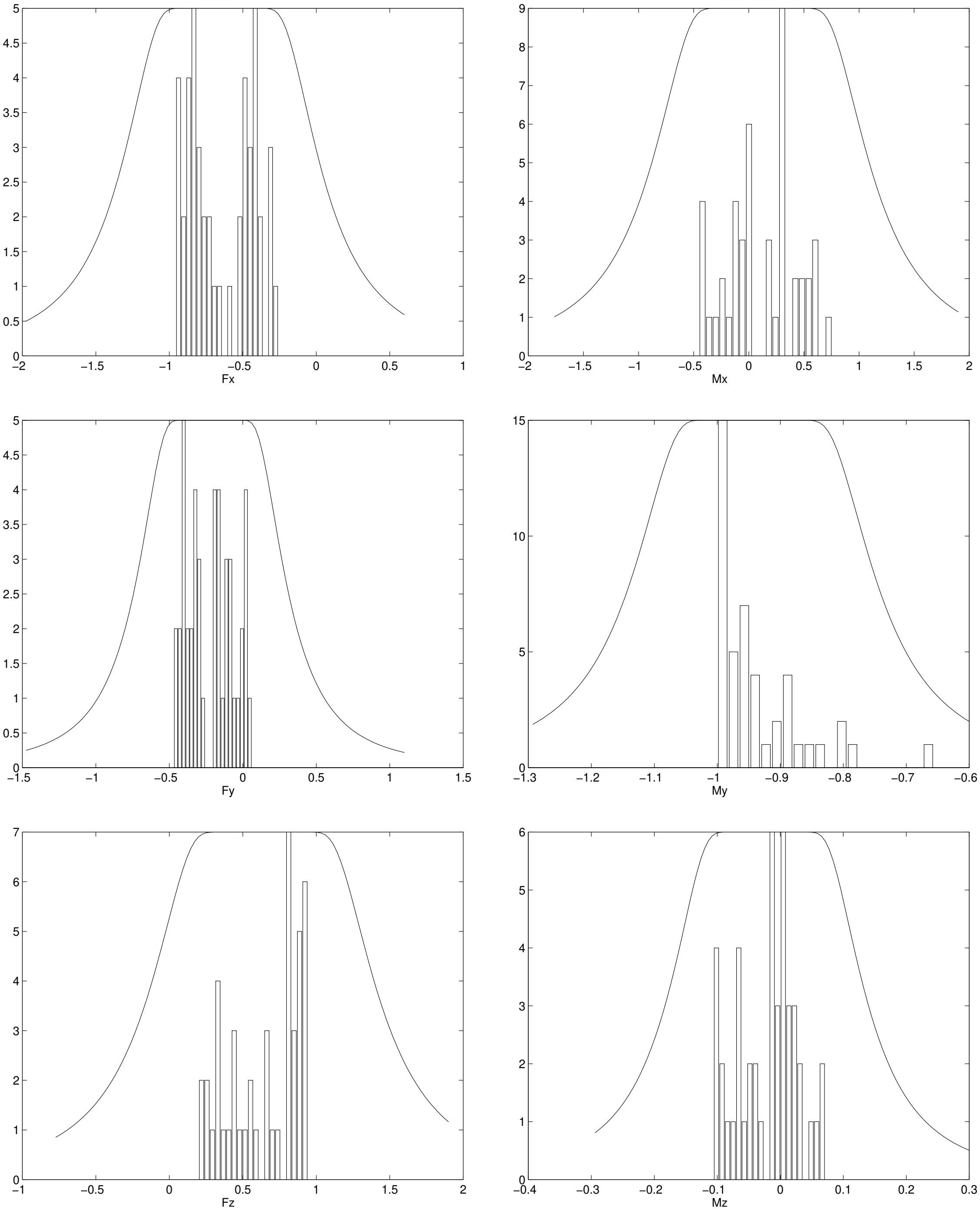


Figure 16. The Distribution of the Edge 2 Training Data Shown with the Generated Membership Functions (©1996 IEEE. Reprinted, with permission, from [65])

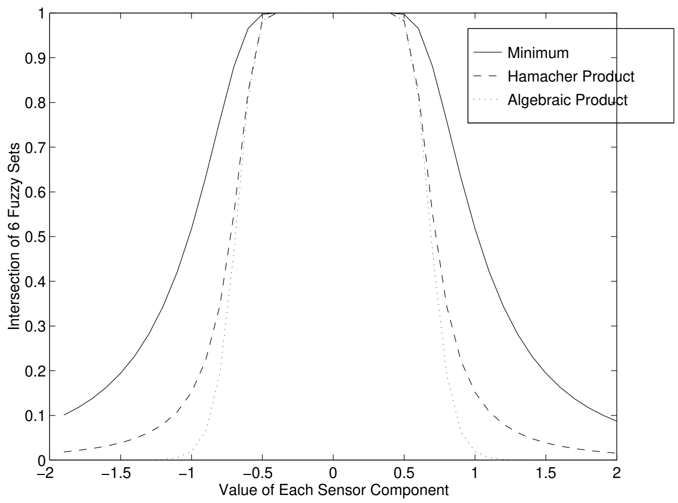


Figure 17. A Comparison of 3 Conjunction Operators, Showing the Result of Combining 6 Identical Membership Functions, with a Mean of 0.0 and a Standard Deviation of 0.3 (©1996 IEEE. Reprinted, with permission, from [65])

Other conjunction operators were investigated, including the minimum function and the algebraic product. The Hamacher product was ultimately chosen because it consistently yielded the highest success rate in classifying the test data sets. Figure 17 shows the comparison of the 3 intersection operators, by plotting the resulting combination of 6 identical membership functions. The Hamacher and algebraic products drop off much faster than the minimum function, thereby providing better discrimination between classes which are close together in feature space. The Hamacher product broadens the combined function slightly more than the algebraic product. Empirical results have shown it to be slightly better than the algebraic product.

The effects of the combination can be seen in Figures 18 and 19, where only 2 of the features are combined. Although this is a partial combination only, the distinct shapes are apparent, showing how an edge pattern is shaped differently compared to a vertex pattern. From Figures 18 and 19, you can also see the effects of imperfect

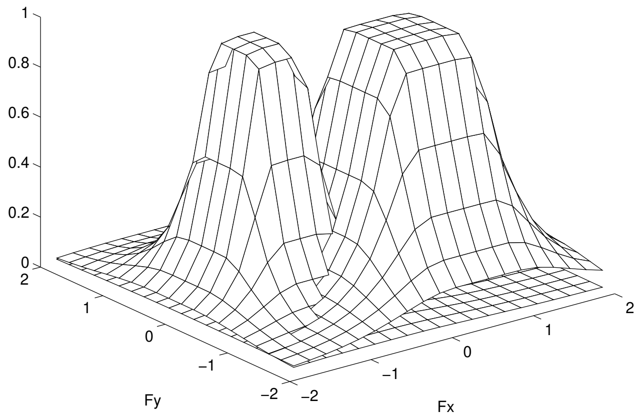


Figure 18. A Combination of Two Membership Functions for Vertex a and Vertex c (©1996 IEEE. Reprinted, with permission, from [65])

training data. Theoretically, the 2 vertex classes, and likewise the 2 edge classes, should have an identical shape.

Figures 18 and 19 also illustrate a limitation in this classification method. In Figure 19, the edge membership functions are well-defined because the edge features are aligned with one of the sensor axes. If the edges were aligned at an angle of 45 degrees to a sensor axis, then the shape of the combined membership functions would look similar to the vertex combinations shown in Figure 18. This has the effect of expanding the input feature space for that class. In this situation, the classifier may still be able to successfully identify the correct class. However, depending on the shape of the grasped object, it is also possible that the expanded feature space may now overlap the feature space of another class, making classification difficult. The use of the neural network, presented in the next section, is intended to overcome this limitation. The fuzzy classifier does provide an excellent method for identifying the

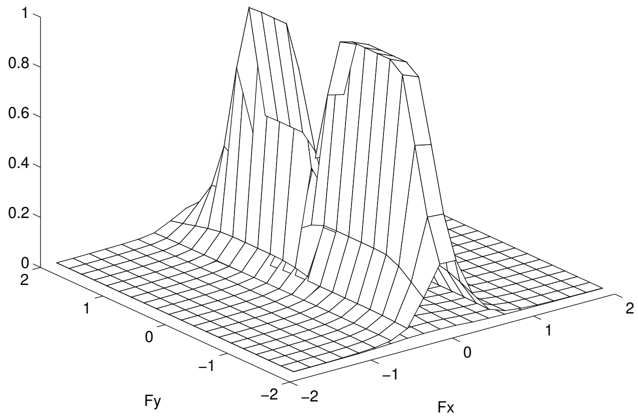


Figure 19. A Combination of Two Membership Functions for Edge 1 and Edge 3 (©1996 IEEE. Reprinted, with permission, from [65])

no contact class C_0, as the $\tilde{Z}ero$ force magnitude membership function aptly describes the no contact condition for a noisy force sensor.

The operation of the fuzzy classifier will be illustrated with an example. Using the parameters for the *edge2* class, we will calculate the possibility of an input signal belonging to the *edge2* cluster. The class parameters are given in Table I. Note that only the force magnitude parameters have units, as the other membership functions use normalized vectors. A sample force input signal is shown in Table II; both the raw input signal and the normalized components are given. The first 6 membership function grades are calculated and also shown in Table II. The membership function grade for $\tilde{Z}ero$ force magnitude is 0.065, and the grade for $NOT\ \tilde{Z}ero$ is 0.935. All 7 membership function grades are combined using the Hammacher function, and the final result is a possibility of 0.886 that this input vector belongs to the *edge 2* class. This process would be repeated for each of the classes. If the possibility values for the remaining classes are less than 0.886, then this input signal would be classified as being in the *edge 2* set.

TABLE I

Membership Function Parameters for the Edge 2 Class

Parameter	$\tilde{F}x$	$\tilde{F}y$	$\tilde{F}z$	$\tilde{M}x$	$\tilde{M}y$	$\tilde{M}z$	$\tilde{Z}ero$
μ	-0.640	-0.219	0.644	0.121	-0.936	-0.021	0.000 N
σ	0.214	0.155	0.239	0.318	0.073	0.048	0.979 N

TABLE II

Sample Input Vector (Forces in N; Moments in Nmm)

Vector	f_x	f_y	f_z	m_x	m_y	m_z	$\|f\|$
input vector	-3.96	0.68	6.01	-315.56	-624.57	-75.50	7.23
normalized vector	-0.55	0.09	0.83	-0.45	-0.89	-0.11	
membership grade	1.00	0.96	1.00	0.99	1.00	0.99	

2. The Neural Network Classifier

To overcome the potential limitations of the fuzzy classifier, a neural network classifier was designed. The neural network was used as a means of learning the force and moment cone-shaped patterns that may or may not align with the sensor coordinate frame axes.

The neural network classifier does not attempt to identify the no contact class C_0, and, as such, the force magnitude is not included as an input feature. In this case, a hierarchical approach is used. The fuzzy rule for class C_0 is used to determine whether there is contact or not. If the force magnitude indicates contact, then the

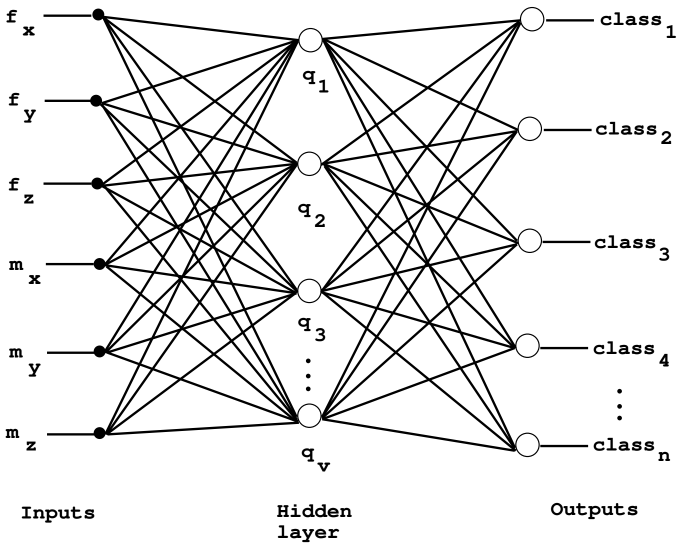


Figure 20. The Generalized Architecture for n Classes and v Vertices (©1997 IEEE. Reprinted, with permission, from [63])

neural network classifier is run to determine which SECF class corresponds to the sensor signals.

The network architecture is taken, in part, from a structure proposed by Asada for mapping force signals to velocity commands [2]. Asada included a hidden layer, which identified the contact formation, although he did not use this information in that form. The generalized neural network architecture is shown in Figure 20. The 6 input features are the normalized components of the force and moment vectors. The number of output nodes is the number of SECF classes to be identified (not including the no contact class C_0. The values at the output nodes may be interpreted as confidence factors for each class. The highest confidence factor represents the identified class. As with the fuzzy classifier, a threshold is used to determine whether the input vector is unclassified.

In keeping with Asada's proposed architecture, the number of hidden nodes is dependent on the geometric shape of the object. There is one hidden node for each vertex on the object that is part of an included SECF. That is, each vertex that can make contact with the environment, as part of the SECF set, has a corresponding hidden node in the network. This is done because surface and edge contacts are modeled as a finite set of point contacts. For example, an edge contact can be modeled as 2 point contacts, 1 at each end of the edge (i.e., the end vertices). Likewise, a surface contact can be modeled as 4 point contacts at the corner vertices. The connections between the 6 input signals and the hidden nodes correspond physically to the forces exerted on the object's vertex points. The output of the hidden layer indicates at which vertex points contact has been established. The output of the final layer then maps this information to the corresponding SECF class. In this way, the network structure has physical meaning. For an example, refer to the pentagon-shaped block shown in Figure 21, which contains 11 SECF's (5 sides, 5 edges, and the bottom surface). Within those 11 SECF's, there are 10 vertices that can make contact with the environment; therefore, the corresponding network would have 10 hidden nodes.

As with the fuzzy classifier, training data is acquired by demonstrating each SECF class and collecting the force and moment signals generated. The training of the neural network is accomplished by using backpropagation with a constant learning rate (lr) [20]. The bipolar sigmoid is used as the activation function for both the hidden and output nodes. Training is continued until the total squared error (tse) establishes an increasing instead of decreasing pattern. At that point, the training

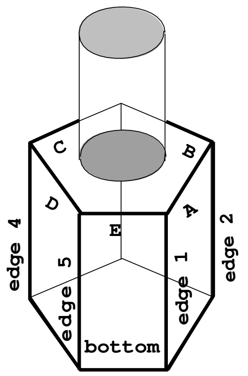


Figure 21. Pentagon Block with 11 Contact Formations: 5 Sides, 5 Edges Plus the Bottom Surface (Using 10 Vertices)

is stopped. The final weights and biases used for classification are those with the smallest tse encountered throughout the training process.

We will illustrate the performance of the neural net classifier with 2 data sets.[6] The *pentagon* set contains 11 classes and was made using the plastic, pentagon-shaped block shown in Figure 21. The set contains 6 surface and 5 edge classes. The *peg(11)* set contains 11 SECF classes made using a 4-sided aluminum peg. The 11 classes include the bottom and 4 side surfaces and the 4 bottom edges of the peg. Two additional classes are created by jamming the peg against opposing inside surfaces of a hole.

To test the neural network classifier, each data set was separated into two groups; one group was used for training and the second group was used for testing. The neural network classifier was trained using the backpropagation procedure described above. With the *pentagon* set, the trained neural net classifier achieved a success rate of 98.5%, which is quite close to the success rate of the fuzzy classifier at 98.8%. With

[6]The complete set of data is given in Chapter VI.

the *peg(11)* data set, the trained neural net classifier achieved a success rate of only 79%. Compare this to the fuzzy classifier, where a 94% success rate was achieved.

To better understand the results, an interactive visualization program was written to analyze the data sets and observe whether the input vectors overlapped from one SECF class to another. For the *pentagon* set, where no overlap was observed, both the fuzzy and neural net classifiers performed quite well. However, in the *peg(11)* set, there was significant overlap or very close sets between some pairs of SECF classes. The situation is illustrated in Figures 22 and 23, which show the force and moment vector sets for the right side SECF and the right bottom edge SECF. As the figures illustrate, the data sets are actually overlapping. Similarly, there is overlap between other pairs of data sets (e.g., the left side and the left bottom edge). A pictorial representation of 2 overlapping clusters is shown in Figure 24. This may explain the rather poor performance of the neural net classifier when the classic backpropagation training is used. The fuzzy classifier is not as sensitive to overlapping input sets, because membership functions are based strictly on the means and standard deviations of the input vectors.

In an effort to improve the performance of the neural net classifier in the case of overlapping data sets, alternative training methods were investigated. The attempts included adjusting the learning rate, using an adaptive learning rate, and using the Nguyen-Widrow initialization of weights [20]. None of these methods yielded significantly-improved results.

As a result, a two-step training process was developed. First, the classifier was

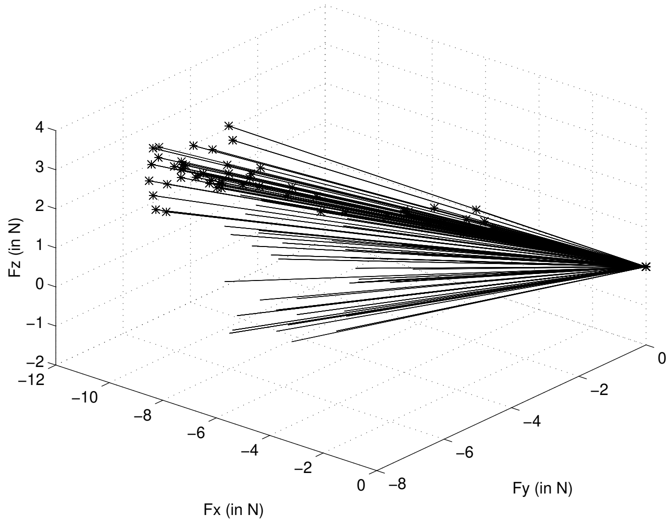


Figure 22. Force Vectors for Two Overlapping Data Sets

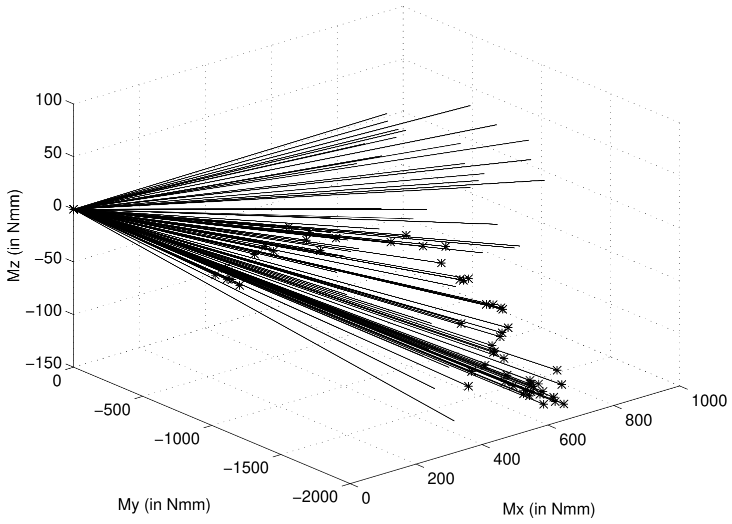


Figure 23. Moment Vectors for Two Overlapping Data Sets

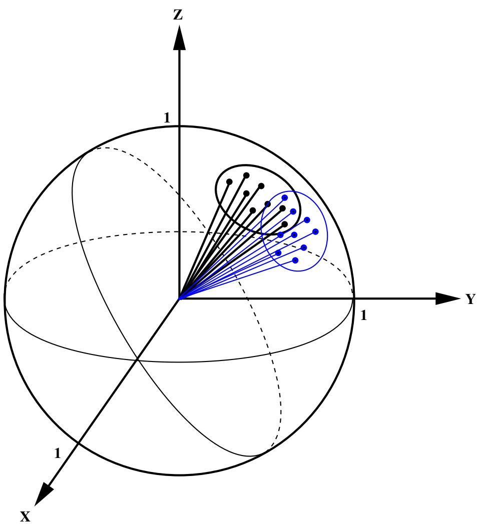


Figure 24. Projected Vectors Shown as Two Overlapping Clusters

trained using a small set of prototype vectors. For each SECF class, 10 prototype vectors were generated using the means and standard deviations of the 6 input features for that class. Each generated prototype vector was within a range of 1 standard deviation of the mean for that class. As illustrated in Figure 25, this resulted in clean data with a well-defined separation between the sets of input vectors. The weights and biases learned by the network using the prototype vectors were saved and then used as starting weights and biases for the second phase of training. In the second phase, the complete training set was used to train the neural net.

Using the modified training method, the performance of the neural net classifier was improved dramatically. In the case of the *peg(11)* set, the classifier achieved a success rate of 98%. Again, compare these results with the 94% success rate of the fuzzy classifier. For the *peg(11)* set, the peg was rotated about the z-axis by approximately 40 degrees, so that the geometric features of the peg were not aligned

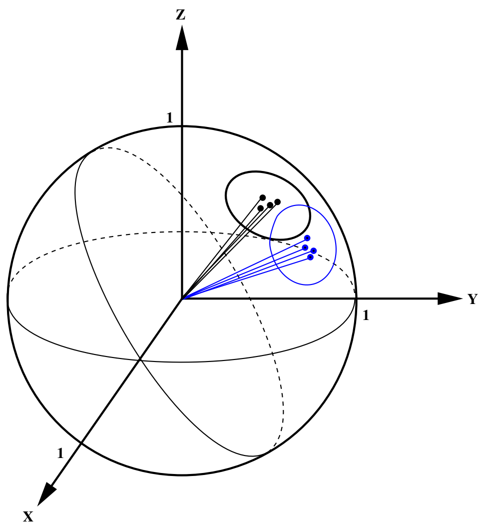


Figure 25. Generating a Small Set of Prototype Vectors for the First Phase of Training

with the sensor frame. As anticipated, the neural net classifier (with the modified training method) had a higher success rate than the fuzzy classifier for this data set. The results are summarized in Table III. More complete results are provided in Chapter VI, along with the results of additional data sets.

TABLE III

Preliminary Classifier Results

Classifier	Set	Good	Misses	Unclass.	Success Rate
Fuzzy	pentagon	1822	15	8	98.8
NN	pentagon	1818	11	16	98.5
Fuzzy	peg(11)	1333	82	6	93.8
NN	peg(11)	1119	192	110	78.7
Modified NN	peg(11)	1392	17	12	98.0

E. Learning a Sequence of SECF's

As mentioned previously, the learning of an assembly skill, as modeled here, involves the learning of 3 functions. In the last section, we discussed learning the mapping of force sensor signals to SECF states. In this section, we shall discuss the learning of the second function– the desired sequence of SECF states.

To learn a sequence of SECF's, we must be able to identify the SECF when the robot is moving and the grasped object makes contact with its environment. Ideally, we would like to use an efficient method that can be used both to extract a sequence of SECF's off-line from demonstration data and also to monitor the contact formation in real-time during both the demonstration and the execution of the skill. Looking at the hybrid control model of an assembly skill, as shown in Figure 3, an event trigger is used to determine when the system has changed to a new qualitative state and, therefore, when the state classifier should be run to identify the new SECF. The idea of an event trigger, as used here, is analogous to the force event defined by McCarragher [43], which is a gain or loss of contact resulting in a new contact formation. The force event is observed as a significant change in force. By monitoring the change in force (performed as a threshold check), we can decide when the classifier should be run to detect a change in SECF.

When the change in force is used to identify the actual contact formation (as McCarragher has done), the setting of the threshold value is critical. If the threshold is set too high, it is possible to miss an event. If the threshold is set too low, it is possible to incorrectly interpret dynamic effects as a change in contact. With our approach, the value of the threshold is not so critical. It is still possible to set the threshold too high and miss significant force events. However, setting the threshold too low will merely result in a loss of efficiency but will not affect the correctness of classification.

With this in mind, we use the concept of a force event as an event trigger for classifying time series data. A force event is defined as a significant change in any of the 3 force components. That is, a threshold check is performed for each of the 3 individual force components. If a force event is detected, then the force magnitude is calculated, and the force and moment vectors are normalized. The state classifier is then executed using these seven features as described in the previous section. This provides a fast, efficient means of monitoring SECF's.

The actual threshold value used in identifying the force event has been determined experimentally. A force event and SECF monitoring program was implemented as part of the demonstration system. As the human teacher demonstrated a skill, the SECF was classified on-line in real-time. Different threshold values were tested for use as an event trigger until a consistent and reliable value was found. For the class of experiments used in this work, the force event threshold was ultimately set to 0.03 N/msec. The threshold is dependent on the frequency at which the force sensor signals are sampled. The threshold is compared to the change in 2 consecutive force signal samples, so it is compared against the change in force over a given time period. If the sampling period is larger (i.e., a longer time between consecutive samples), then a unipolar rate in the change in force would be larger for a particular force event.

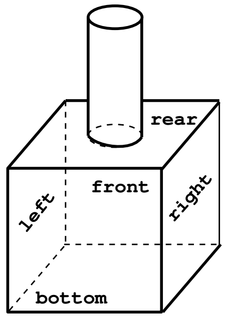


Figure 26. Square Block Workpiece

The learning procedure will be illustrated as off-line learning using profile data from an actual skill demonstration. The workpiece used is the plastic, square block shown in Figure 26. For this skill, 17 SECF classes (plus the *no contact* class) are used. The following SECF's comprise the set: 5 faces of the block (*front, right, rear, left,* and *bottom*), 4 combinations of a side face and the bottom face (*front bottom, right bottom, rear bottom,* and *left bottom*), 4 combinations of 2 side faces (*right front, right rear, left front,* and *left rear*), and finally, the 4 combinations of 3 faces that make up a corner (*right front bottom, right rear bottom, left front bottom,* and *left rear bottom*). The demonstrated skill uses the following sequence of SECF states: *no contact* to *right front* to *right* side to *right bottom.* For this illustration, the fuzzy classifier is used, as discussed in the previous section. After training, the classifier performs at a 97% success rate with static data.

The force and moment profiles for the skill demonstration are shown in Figures 27 and 28. The data shown here has been logged at a frequency of 40 Hz. Using the event trigger discussed above, along with the fuzzy SECF classifier, a sequence of SECF states was identified. The results are shown in Figure 29. For purposes of graphing

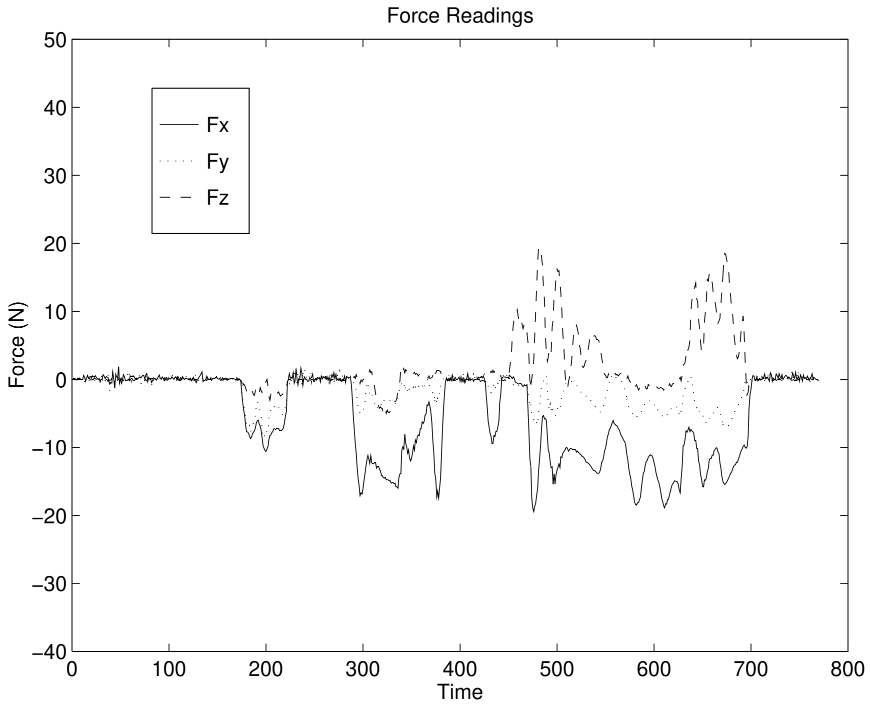


Figure 27. Force Profile of a Skill Demonstration

the results, an index is assigned to each SECF. The *no contact* class is shown with an index of 0. The unclassified cases are shown with an index of -1.

A filtering algorithm is applied to the classification results, yielding new results as shown in Figure 30. Transitory classes are filtered out, which has the effect of eliminating undesired human-generated actions. Human factors studies have shown that the bandwidth for human limb motion is at most 5 Hz for internally generated trajectories and around 10 Hz for reflex actions [17]. In this illustration, we have filtered out transitory states occurring faster than 10 Hz. The filtering algorithm also eliminates the unclassified cases. For the points of no classification, we assume that the SECF has not changed, and these points are graphed as the previously identified SECF.

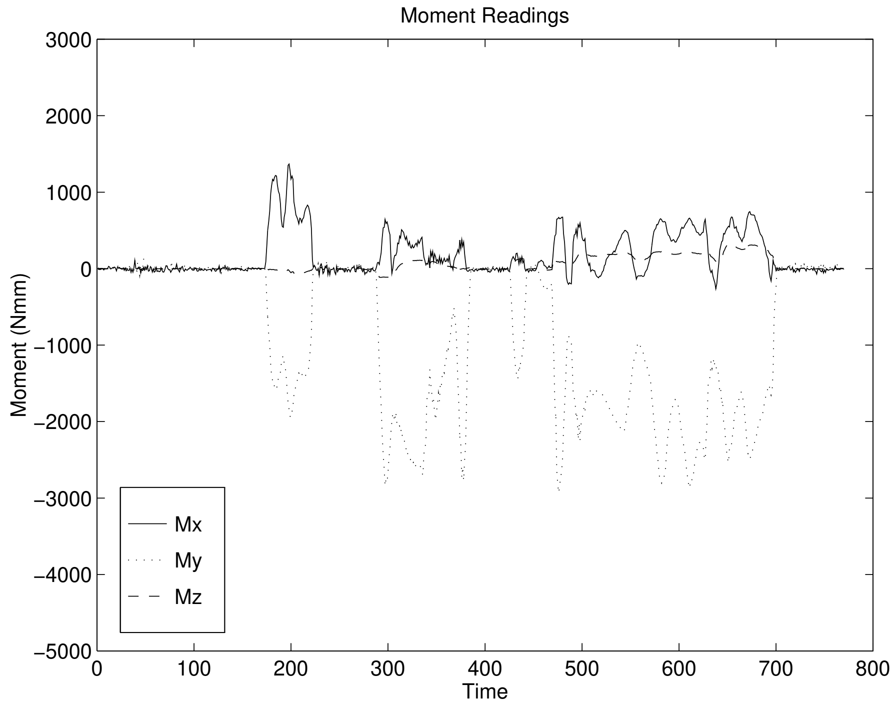


Figure 28. Moment Profile of a Skill Demonstration

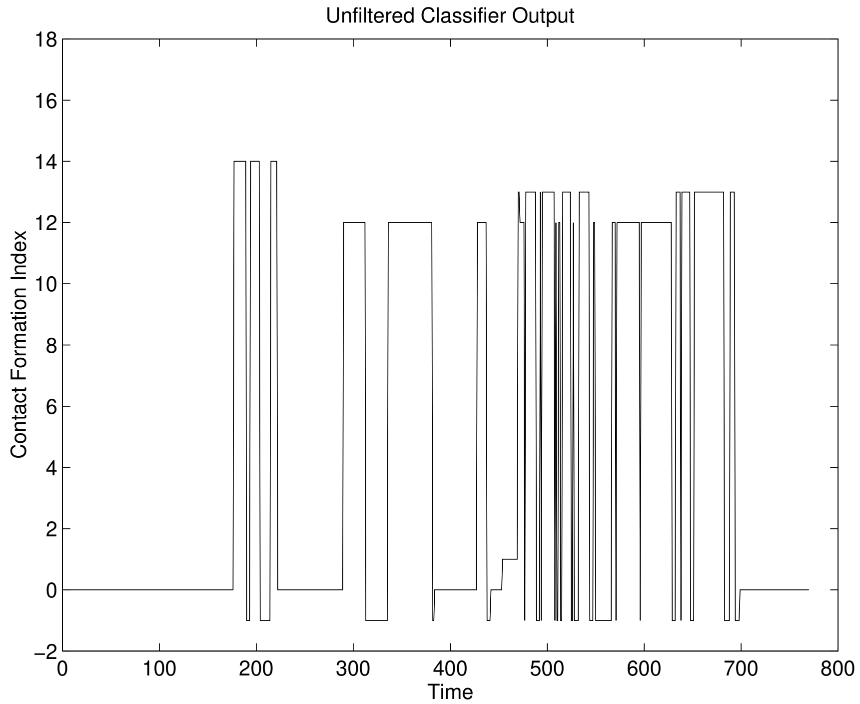


Figure 29. Unfiltered Classifier Output of a Skill Demonstration

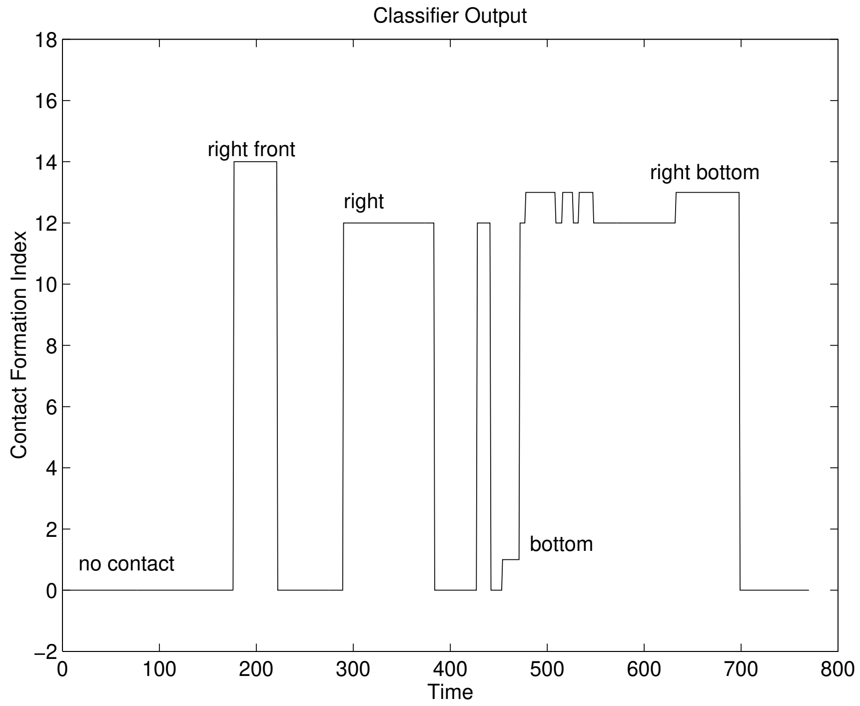


Figure 30. Filtered Classifier Output of a Skill Demonstration

The resulting sequence is an accurate identification of the demonstrated sequence; however, this is not exactly the sequence that was intended. As Figure 30 shows, the *bottom* SECF was encountered accidently, and there were several instances of oscillating between SECF's, such as the changes between *right bottom* and the *right* side. Also, there were periods of *no contact* that we might prefer to avoid in order to achieve a smoother skill execution. As a result, this is not yet the sequence that we want the robot to duplicate.

F. Refining the Sequence

The illustration presented in the previous section provides an example of an imperfect skill demonstration. If we could assume that every demonstrated skill sequence was perfect, then the method presented would be adequate. However, a stated theme of this entire work is the emphasis on making the teaching/programming process easy for the human operator, and it is not always easy for the operator to provide perfect demonstrations. Often, it may be easier for the operator to look at the results of an imperfect demonstration and modify those results directly.

This also relates to the earlier discussion of the teacher-student interface model and the skill model representing the student's knowledge. The sequence, as shown in Figure 30 is part of the robot student's skill model. In our teacher-student model, the human teacher should be able to analyze the student's skill model (i.e., knowledge) and determine whether further refinement (i.e., additional teaching) is necessary.

For this purpose, an interactive, iterative teaching environment is suggested, in which the operator can observe the learned SECF sequence and refine it if necessary. For example, an operator can easily look at the sequence shown in Figure 30 and determine which SECF steps are essential and which steps are unnecessary. This has been done, and Figure 31 identifies the selected sequence by showing the selected force events with vertical hash lines. The final sequence is shown in Figure 32.

G. Learning Transitions between SECF's

The final function to be learned, as part of the skill model, is the transition velocity information which will move the robot from the current SECF to the next desired SECF. Let us begin by looking at the demonstrated translation and rotation profiles that correspond to our sample illustration. Figures 33 and 34 show the corresponding

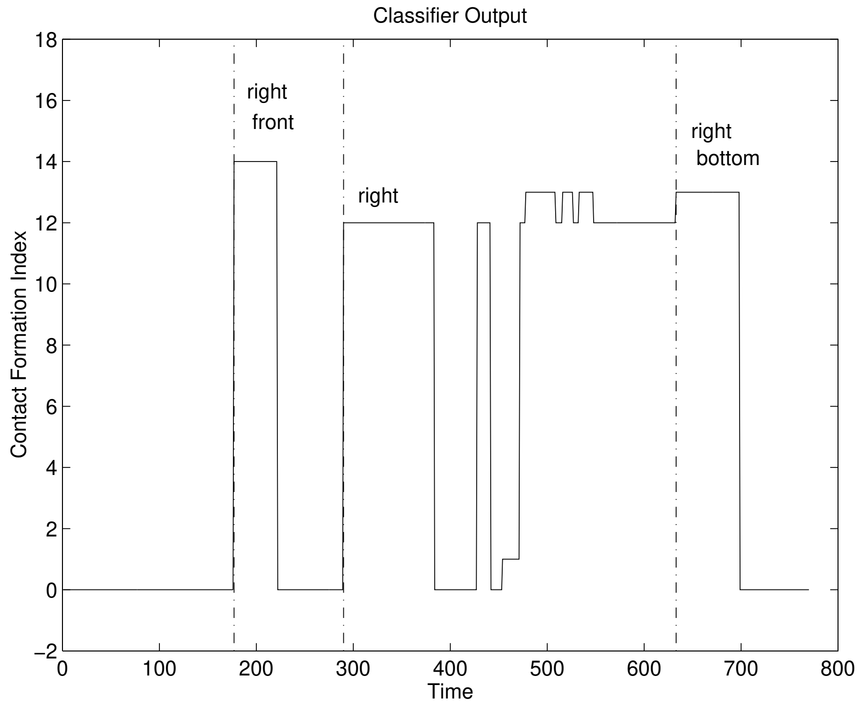


Figure 31. Filtered Classifier Output of a Skill Demonstration, with Selected Events

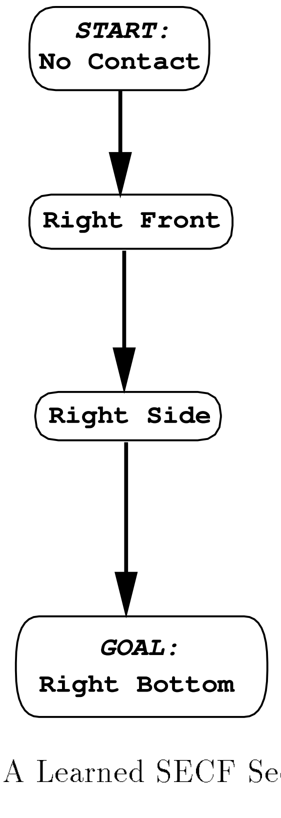


Figure 32. A Learned SECF Sequence

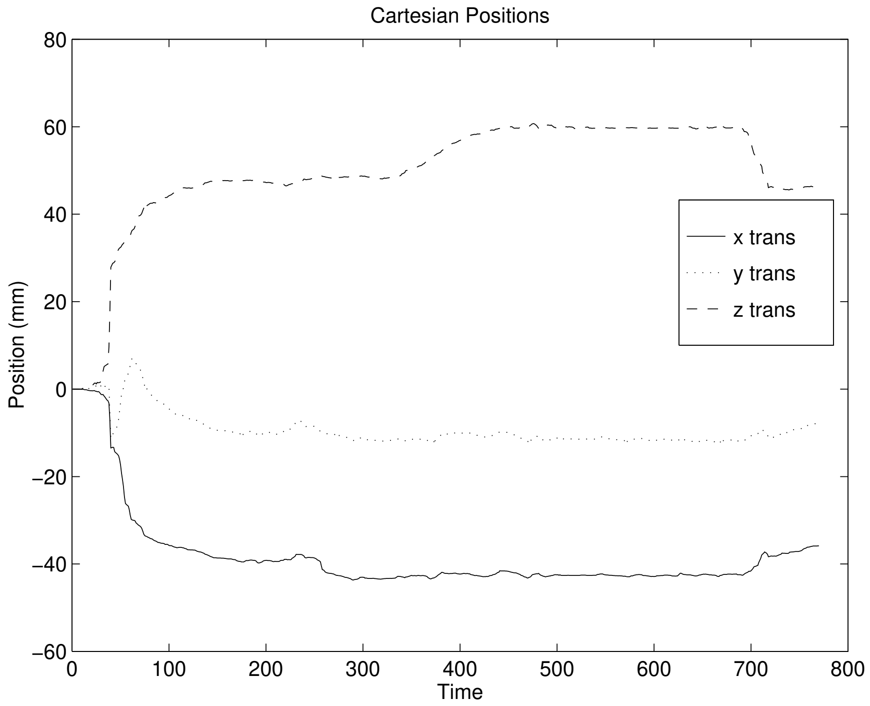


Figure 33. Translation Profile of a Skill Demonstration

profiles. From the graphed profiles, we can see that the motion generated by the human demonstrator is not necessarily smooth. In learning velocity commands that will generate a trajectory, it is not good enough to simply duplicate what the human demonstrator has done. We do not want to duplicate inconsistent or unintended actions, but rather we want to extract from the demonstrated profiles that action which is purposeful. That is, we want to extract actions that will drive the system to the next desired SECF state.

One approach is to approximate the trajectory as a series of linear segments. The selected force events, which define the change in SECF, can be used as reference points in segmenting the demonstrated skill. The result is a piecewise linear trajectory which drives the system towards the next desired state. Because force events are used

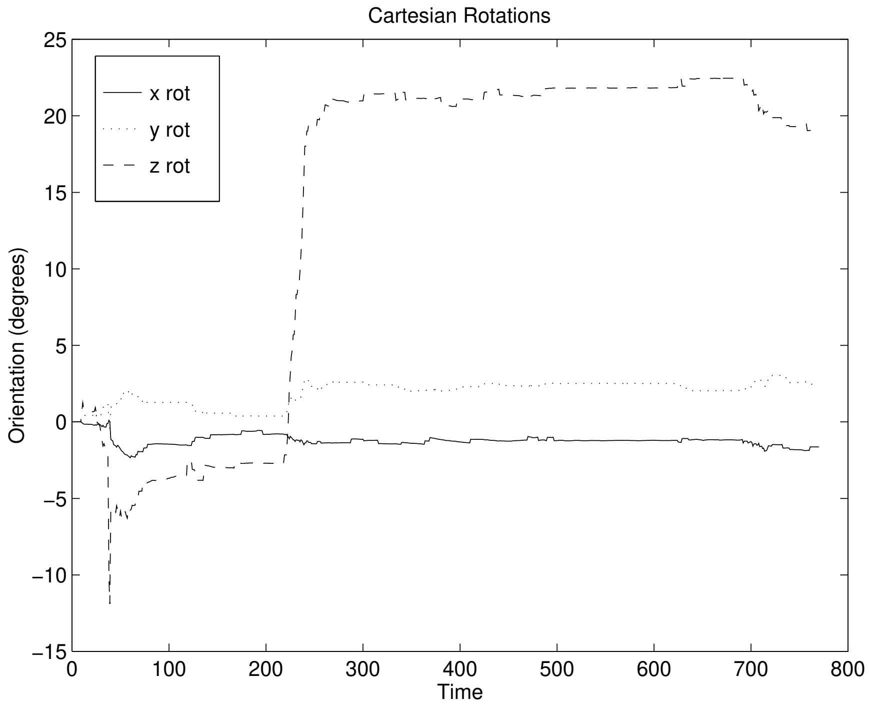


Figure 34. Rotation Profile of a Skill Demonstration

as reference points instead of position information, the trajectory generation is not dependent on the precise position.

The generation of the translational component of the reference velocity command is straightforward. Let t_p and t_{p+1} be the times at which force events p and p+*1* occur in the demonstrated position profile. Then the distances, d_x, d_y, and d_z, which describe the distance traveled in the x, y, and z directions, are calculated as follows:

$$d_x = x(t_{p+1}) - x(t_p)$$

$$d_y = y(t_{p+1}) - y(t_p)$$

$$d_z = z(t_{p+1}) - z(t_p)$$

These distances are then normalized as shown below:

$$d_{xn} = \frac{d_x}{d}$$

$$d_{yn} = \frac{d_y}{d}$$

$$d_{zn} = \frac{d_z}{d}$$

where

$$d = \sqrt{d_x^2 + d_y^2 + d_z^2}$$

The normalized distances, d_{xn}, d_{yn}, and d_{zn}, are used to define a relative direction. Assuming a constant speed, the normalized distances can be used to generate the translational component of the reference velocity command that drives the system from SECF state *w(p)* to state *w(p+1)*. The process is illustrated in Figure 35, where the linear segments are shown for the z translation component.

The generation of the rotational component of the reference velocity command is done in a similar manner. Let $\alpha(t)$, $\beta(t)$, and $\gamma(t)$ be the rotational profiles for the x, y, and z orientations, respectively. Then the rotational distances, r_x, r_y, and r_z are calculated as follows:

$$r_x = \alpha(t_{p+1}) - \alpha(t_p)$$

$$r_y = \beta(t_{p+1}) - \beta(t_p)$$

$$r_z = \gamma(t_{p+1}) - \gamma(t_p)$$

Again assuming a constant rotational speed, the rotational distances can be used to generate the rotational component of the reference velocity command that drives

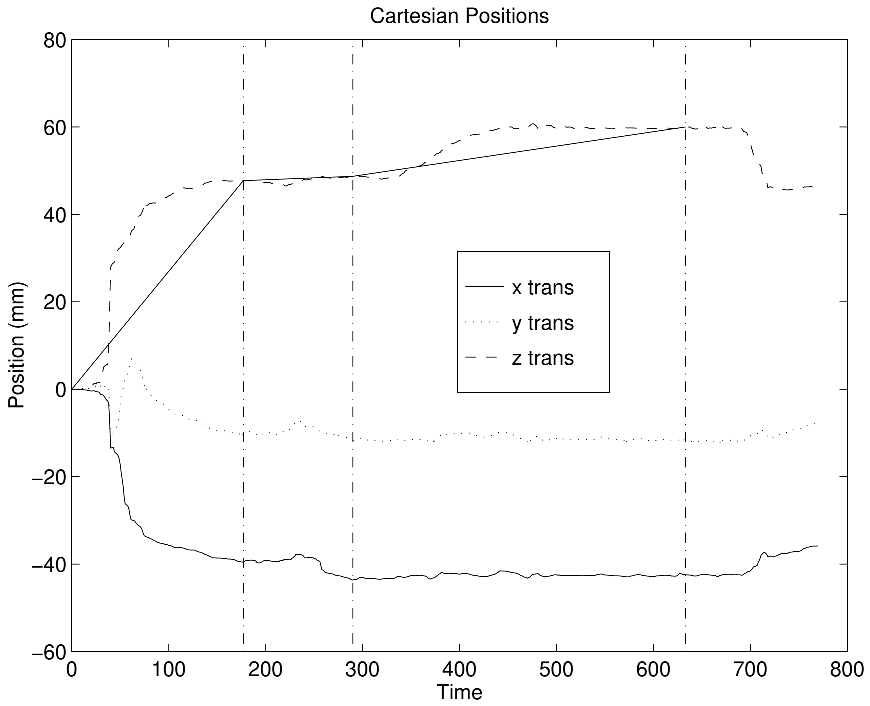


Figure 35. Approximating the Translation Trajectory as a Series of Linear Segments

the system from SECF state *w(p)* to state *w(p+1)*. Figure 36 provides an illustration for the z rotational component.

This strategy provides a relatively simple method for generating the reference velocity commands that is event-based rather than position-based. The limitation, however, is that no concept of speed is learned. As presented, the strategy assumes that a constant speed is used.

To test the effectiveness of the approach, the transition trajectory generation method was implemented as described, and experiments were conducted. These are described in detail in Chapter VI, and a complete set of test results is provided in Appendix A. In the experiments, the robot performed an assembly skill autonomously, using the learned model of the skill. The skill was completed successfully; however,

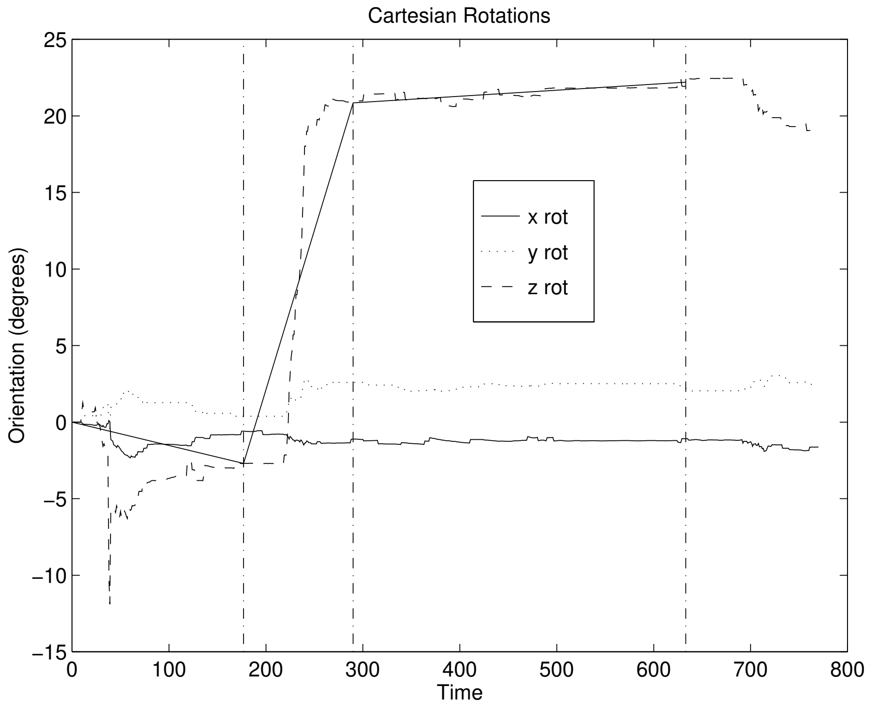


Figure 36. Approximating the Rotation Trajectory as a Series of Linear Segments

the result produced unstable force conditions, as can be seen in Figure 37.

In observing the skill execution, it became apparent that the constant translational speed was creating unstable conditions. For the set of skills investigated in this research, each demonstrated transition segment between SECF states was either mostly translational or mostly rotational. For the segments that were mostly translational, the constant speed worked fine. However, for the segments that should have been mostly rotational, the robot was still being driven at a constant translational speed. This caused the workpiece to be driven into the side of the environment fixture and produced oscillations between contact and no contact conditions.

Originally, we assumed that the force control algorithm, which runs at the same time as the supervisory controller, would eliminate this problem. Proportional force

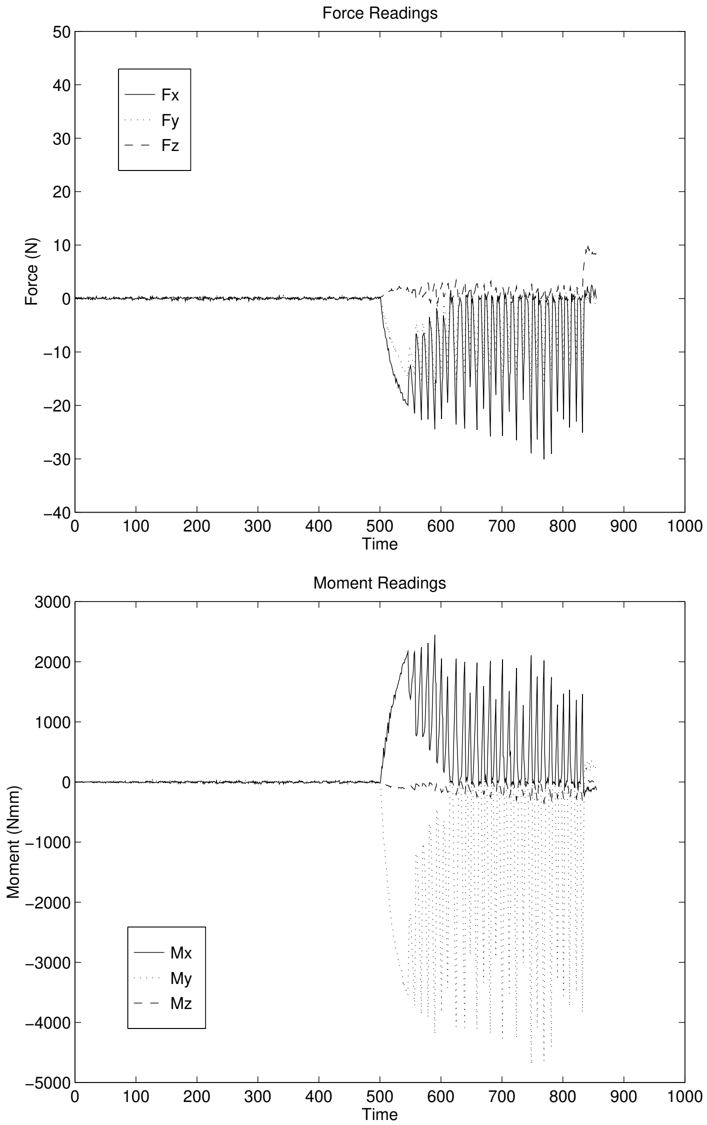


Figure 37. Force and Moment Profiles for Test Run I.b Showing Unstable Force Conditions

control is performed whenever the force and/or moment exceeds a threshold value. In this case, the force control algorithm attempts to hold the force at a constant value until contact is broken. It was assumed that the force control algorithm would prevent excessive translational movement during those segments which had small demonstrated translational distances. Unfortunately, the force control algorithm did not prove to be effective in this case. Additional experiments were conducted, where the force control parameters were adjusted, but the result was still the same unstable force conditions.

In the end, we concluded that the force control loop, which ran at a frequency of 100 Hz, was too slow to produce stable execution for the constant translational speed assigned to the velocity commands (5 mm/sec). The speed could have been reduced, but this would have forced the robot speed to be unacceptably slow. Another alternate was to increase the frequency of the force control loop. Unfortunately, the frequency could not easily be increased, because of limits in the timing resolution of the LynxOS operating system which is used for the robot controller.

Alternative strategies were investigated. The essence of the problem is that the transition segments that are primarily translational should produce mostly translational movement, and the segments that are primarily rotational should produce mostly rotational movement. This can be done by adjusting the ratio between the translational speed and the rotational speed. The rotational distances calculated (r_x, r_y, and r_z) can be used to determine a variable rotational speed. If each rotational distance is spread over a constant time period, then a larger rotational distance will produce a faster rotational speed. We also need a way to adjust the translational speed, so that it is decreased for the segments that are mostly rotational. With this in mind, the translational scale factor is introduced.

The translational scale factor, T_s, is calculated as follows:

$$T_s = min\left(1, \frac{d}{max(r_x, r_y, r_z)}\right)$$

T_s is artificially constrained to be less than or equal to 1, so that the translational speed is not increased beyond what is considered safe. The translational scale factor is multiplied times the translational component of the reference velocity command, and is used to slow the translational speed with respect to the rotational speed. The strategy has worked for the skills tried here, but more work needs to be done on addressing the problem in a general way.

CHAPTER V

TESTBED

Because of the decidedly experimental approach used in this work, it seems appropriate to discuss the equipment used. In this chapter, we will describe both the hardware and software that comprise the testbed equipment and explain the rationale behind its use. Except for small test fixtures, no hardware has been made special for this project. The equipment is off-the-shelf hardware that is (or was at one time) commercially available. The software is a combination of commercially available, public domain, general lab software, and software designed specifically for this research. The testbed will be described as 2 component subsystems: (1) the robot system, and (2) the haptic interface.

A. Robot System

The robot system used in this work was originally part of a testbed developed by USARC[1] for research in distributed telerobotics for space applications [64]. Although the capabilities of the distributed network are not fully utilized here, the robot system provided a reasonable equipment base for these experiments. The equipment is located in the Robotics Laboratory in the Department of Computer Science at Texas A&M University.

[1]Universities Space Automation and Robotics Consortium

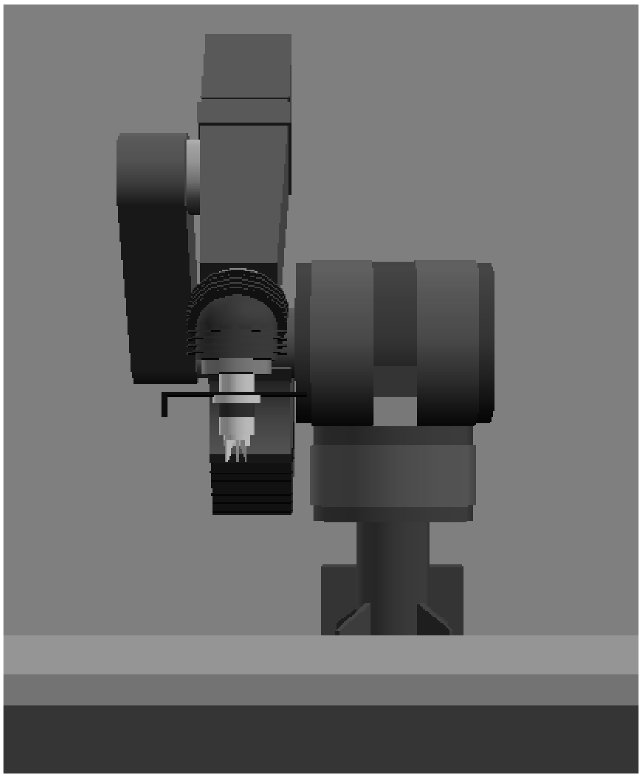


Figure 38. The MERLIN Robot

The robot arm is a 6 degree-of-freedom MERLIN manipulator built by the American Robot Corporation [1]. Kinematically, the MERLIN is similar to the PUMA robot found in many research labs. Stepper motors are used, however, to control the joints. The wrist is equipped with a 3-pronged gripper, which can be opened or closed by pneumatic control. Figure 38 shows a graphical representation of the MERLIN, as it is used for these experiments.

The American Robot Controller that comes with the MERLIN provides joint-level control of the robot. For advanced control functions (e.g., Cartesian control, force control), the MERLIN controller is interfaced to a 133 MHz pentium PC computer via a high-speed bus-to-bus interface. The pentium is running a real-time kernel, called

LynxOS [40], as well as a version of RCCL/RCI [39, 24], which provides a library of functions for robotics applications and control.

A JR3 DSP-based force sensor [31], model 67M25A-U560, is mounted on the robot wrist. Table IV shows the manufacturer's listed capacity and resolution for this model sensor. However, the listed resolution values did not seem realistic under normal operating conditions, given the variety of potential sources contributing to sensor noise.

To get an estimate of the sensor uncertainty under realistic operating conditions, a suite of experiments was performed using the same force sensor and robot [62]. The purpose of the tests was to investigate the effects of environmental factors (such as vibration of the robot and electromagnetic interference), to observe the force sensor signals over a range of the robot workspace, and to ultimately measure uncertainty values that were representative of typical operating conditions.

The results of the experiments showed a sensor distribution that approximates the Gaussian distribution. Generally, 99% of the sensor readings fell within a range of +/- 3 times the standard deviation. As a result, uncertainty values were calculated by multiplying the average standard deviation by 3 for each force and moment component. The average uncertainty values calculated for a 0.6 kg payload are given in Table IV. Note that the measured uncertainty values are significantly higher than the manufacturer's specified resolutions. The manufacturer's specifications probably do not include environmental effects such as vibration of the robot, whereas the measured uncertainty values do.

TABLE IV

Manufacturer's Specifications and Measured Uncertainty (Forces in N; Moments in Nmm)

	Fx	Fy	Fz	Mx	My	Mz
Specs: Capacity	100	100	200	6300	6300	6300
Specs: Resolution	0.0445	0.0445	0.0890	3.4	3.4	3.4
Measured Uncertainty	0.20	0.18	0.47	17.3	16.7	16.1

The software which performs the Cartesian control and the force control of the robot runs on the pentium computer and is written using the RCCL/RCI library. The force limit control described in Section IV.B (used for autonomous robot skill execution) is implemented using an RCI loop at a frequency of 100 Hz[2]. The supervisory control loop is implemented using a separate timing loop at the RCCL level and runs at 50 Hz.

B. Haptic Interface

For the demonstration of the assembly skills, we need a device that provides haptic feedback to the human teacher. For this purpose, a commercially available device called the PHANToM [59] is used. A 6 degree-of-freedom output is generated from the PHANToM, based on the position and orientation of the endpoint connected to a pen-like stylus. A 3 degree-of-freedom force input can also be provided to the PHANToM, which represents the 3 components of a point source of force delivered to the stylus tip. In this way, the PHANToM can be used as a force-reflecting hand

[2]We would prefer to run a force control loop at 600 Hz or more; however, the current version of the LynxOS operating system limits the timing resolution to 10 ms.

controller to drive the MERLIN robot telerobotically. The motion of the stylus is connected via software to the end effector of the robot. The operator holds the PHANToM stylus and moves it to direct the robot in the desired motion. The forces from the force sensor are used as input to the PHANToM so that the operator can feel the same forces sensed by the robot.

For a demonstration environment, it is important to provide a natural interface to the human teacher. As shown in Figure 39, the teacher sits in front of the robot so she can observe the demonstrated operation. The handle of the workpiece is held firmly by the robot's gripper. The teacher grabs hold of the PHANToM stylus, as shown in Figure 40, and starts the demonstration. The purpose of the haptic interface is to allow the teacher to move the grasped workpiece via the PHANToM in such a way as to achieve the same motion and feel as if she was holding the handle of the workpiece directly. Motions of the stylus tip correspond directly to that of the robot end effector. As the teacher moves the stylus to the left, the robot end effector moves to the left. As she moves the stylus to the right, the robot also moves to the right.

The control program used for performing skill demonstrations is written using the RCCL library. To achieve adequate haptic feedback, it is suggested (by the PHANToM manufacturer) that the force feedback loop run at a frequency of at least 800 Hz. Our experience confirms this. Unfortunately, because of the limitation in the timing resolution of the LynxOS operating system, we could not run a controlled loop faster than 100 Hz. Eventually, the problem was resolved by simply letting the loop run as fast as possible. For the experiments discussed here, the PHANToM force

Figure 39. The Testbed Layout, Showing the PHANToM in the Foreground and the MERLIN Robot in the Background

Figure 40. The PHANToM, as Used by the Human Teacher

feedback loop was running at approximately 4000 Hz. For purposes of data collection, the position and force information is sampled at a frequency of 40 Hz.

The forces sensed by the force sensor are pre-processed before they are sent to the PHANToM for haptic feedback. First, offsets are subtracted to eliminate gravitational forces. Forces are effectively nulled at the force sensor, before the demonstration process begins, so that the operator feels zero force when no contact has been established. Second, small forces are ignored. Any force component less than 0.45 N is ignored. Finally, the resulting force signals are run through a third order Butterworth filter and then sent to the PHANToM. The JR3 force sensor also has the capability of filtering force signals. However, adding the filter decreases the sensor bandwidth. It was decided to let the force sensor run at its maximum rate of 8 kHz and perform the filtering in the RCCL control program. This resulted in a very adequate haptic feedback, although no exhaustive study was done to determine just how important the filtering was.

We also experimented with running the robot force control algorithm[3] at the same time as the demonstration control program. This was viewed as a reasonable strategy for increasing the safety during the demonstration process by limiting the forces that could be exerted by the human-generated commands. However, the presence of the computer-generated force control and the implicit human-generated force control caused instabilities in the system such as bouncing. It also resulted in a poor haptic feedback system, creating the feel that the two command sources were fighting each other. This provided a good illustration of the potential problems in combining commands from more than one source and showed that a more sophisticated algorithm would be needed. It is possible that the problem could have been resolved by simply running the robot force control loop at a higher frequency. This was not an option, unfortunately. In the end, it was decided to eliminate the force limit control algorithm during the demonstration process.

The PHANToM proved to be a reasonable haptic display device for the purpose of teaching assembly skills to robots. The PHANToM is easy to operate and provides adequate fidelity in force feedback. The accuracy in position commands is quite good,

[3]Note that the force feedback loop used for the PHANToM as part of the haptic interface is a different loop from the force control loop used to achieve compliant motion for the robot. Unfortunately, the strategy of running a control loop as fast as possible could not be used to increase the frequency of the robot force control loop, due to the interaction between the RCI functions and the LynxOS kernel. As a result, the robot force control loop only runs at 100 Hz.

due to the kinematic links. More resolution could have been used in the three rotational degrees of freedom, but it was adequate for these experiments. In some ways, the limited force feedback provided by the PHANToM (as a point source of force only) may have facilitated the skill teaching. Physiological studies have suggested that the force sensing experienced by humans results from the interaction of neural receptors in the muscles, joints, and skin [46]. Current robotic systems do not have sensors which can duplicate the full sensation of human force sensing, so by comparison, they are haptically handicapped. In terms of haptic feedback, the use of the PHANToM puts the human teacher and the robot on more equal terms, and makes the teacher perform the demonstration with haptic feedback similar to that of the robot.

CHAPTER VI

RESULTS

In this chapter, we present the results of experiments designed to test the methodology. The equipment used for the experiments is described in Chapter V. We present the experimental results in 3 parts. First, the training and performance of the state classifier is discussed for both the fuzzy logic-based classifier and the neural network classifier. This is equivalent to learning the mapping from sensor signals to SECF states. Experimental results are given for 6 different data sets. Next, using one of the 6 data sets, we present results on the rest of the skill acquisition – the learning of a sequence of SECF states and the transitions which drive the robot system from one SECF to the next desired SECF. We will illustrate how two learned sequences can be merged into one FSM, representing multiple successful skill strategies. Finally, we present the results of the autonomous execution of the learned skill as performed by the robot. Test runs were performed under different starting conditions; that is, the position and the orientation of the environment were changed between test runs. The experimental results show that the robot was able to successfully complete the learned skill in spite of the changes in the environment.

A. Classifier Training and Performance

Both the fuzzy classifier and the neural network classifier were trained by using supervised learning. The training data used was in the form of input vectors (force and moment vector components from the force sensor) with the corresponding SECF. Training data was generated by demonstrating a known SECF and monitoring the signals from a force sensor. The object was held in a fixed grasp by a robot arm. A JR3 force sensor mounted on the arm provided the force signals. For each set collected, the data was separated into 2 groups, one used for training the classifiers and the second used for testing.

The demonstration of a SECF was done in one of two ways: (1) the robot arm was held in a fixed position and a human assistant manipulated the environment so that contact was made, or (2) the robot arm was driven by a teleoperator so that the grasped object made contact with the environment. In the second case, the PHANToM haptic device was used as a handcontroller so that haptic feedback was provided to the operator. In both demonstration modes, it was quite possible to collect data from an unintended SECF, thus generating a kind of noise. This occurred because the data collection process was done manually, and it was possible for the operator to accidently hit an undesired SECF. Although every effort was made to collect clean data, undoubtedly some of this noise was included in the training sets, so that some input vectors may have been assigned to an incorrect SECF.

1. Data Sets

Six sets of training data were used in the experiments, representing different shapes of objects and different combinations of SECF classes. Different combinations of classes were chosen to test the effectiveness and determine the potential limitations of the two classification methods. The training sets are characterized below:

- The *pentagon* set contains 11 classes and was made using the plastic, pentagon-shaped block shown in Figure 21. The set contains 6 surface and 5 edge SECF's.

- The *plug* set was made using a slightly flexible 3-prong electrical plug. The 7 classes of SECF's included the 7 combinations of single, double, and triple prongs, making contact against a flat surface.

- The *peg(11)* set was made using a 4-sided aluminum peg. The 11 classes included the bottom and 4 side surfaces and the 4 bottom edges. Two additional SECF's were created by jamming the peg against 2 opposing inside surfaces of a hole. The peg was oriented in the gripper so that its geometric features did not align with the sensor frame axes. The peg was rotated approximately 40 degrees about the z-axis.

- The *peg(9)* set was also made with the aluminum peg. The 9 classes included the bottom surface, 4 side surfaces, and 4 double-sided combinations (front and left, front and right, rear and left, rear and right). Note that, although the same peg was used for the *peg(11)* set, this data set includes a different group of SECF's, most notably, the 4 combinations of two side surfaces. The double side combinations were included to test for classification in instances where there is more than one constraint.

- The *square(17)* set was made using the small, plastic, square block shown in Figure 26. The 17 classes included the bottom and 4 side surfaces, 8 double-sided combinations, and the 4 bottom corner surface combinations.[1]

- The *square(8)* set was also made using the square block shown in Figure 26. This set was designed to test classification in the situation where 2 or more SECF's could produce the same force vector. The 8 classes include the 4 bottom edges and the 4 combinations of a side surface with the bottom surface (*front bottom*, *right bottom*, *rear bottom*, and *left bottom*).

2. Classifier Results

The classifiers have been trained as discussed in Chapter IV. Training the fuzzy classifier involves the generation of the membership functions for each feature in each class. The membership functions are parameterized based on the means and standard deviations of the input vectors in the training set. As a result, membership function generation involves simply calculating the means and standard deviations for each feature. The neural network classifier is trained by using backpropagation with a constant learning rate (lr). Training continues until the total squared error (tse) establishes an increasing instead of decreasing pattern. The training is stopped at

[1]This is the same SECF set described in Section IV.E, and it is used again in Section VI.B.

that point, and the final weights and biases used for classification are those with the smallest tse encountered throughout the training process.

Results of the fuzzy classifier are shown in Table V. The classifier performance for the *pentagon* and *square(17)* sets is quite good. For the *peg(11)* and *peg(9)* sets, the peg was rotated about the z-axis by approximately 40 degrees, so that the geometric features of the peg were not aligned with the sensor frame. This may account for part of the decreased performance in these sets.

TABLE V

Fuzzy Classifier Results

Set	Classes	good	misses	unclass.	success rate
pentagon	11	1822	15	8	98.8
plug	7	168	5	6	93.9
peg(11)	11	1333	82	6	93.8
peg(9)	9	3354	204	26	93.6
square(17)	17	8233	52	166	97.4
square(8)	8	1657	299	28	83.5

To further analyze the data sets, an interactive visualization program was developed so that the sets of input vectors could easily be observed. Force vectors were displayed in one window; moment vectors were displayed in a second window. A handcontroller device was used to rotate and translate the vector sets so they could be observed from different viewpoints. When using the visualization program for the *plug* set, we observed that a few vectors looked misplaced, probably as a result of the training-generated noise discussed earlier. An example is illustrated in Figures 41 and 42. As shown in the graphs, it appears that at least one input vector was incorrectly assigned to the *left & center* SECF class, most likely as a result of the data collection procedure. For a relatively small number of training vectors, these misplaced vectors could skew the calculation of the means and standard deviations used to generate the membership functions.

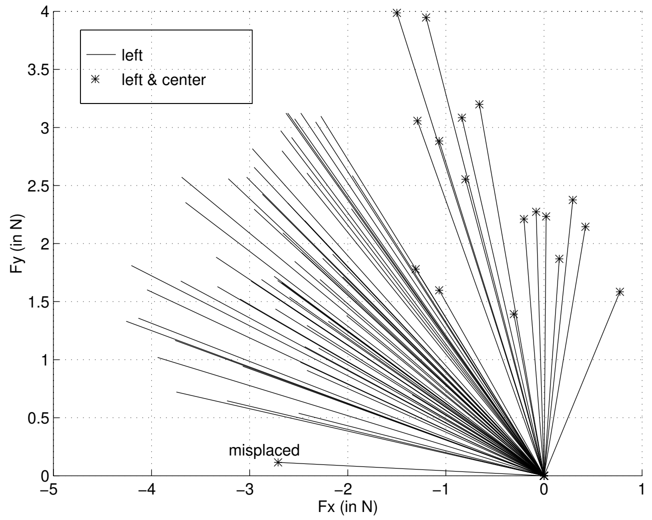


Figure 41. Misplaced Force Vectors from the *plug* Set

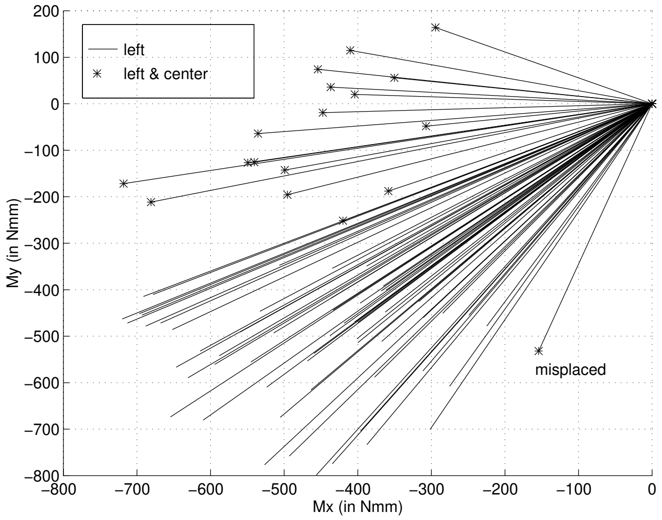


Figure 42. Misplaced Moment Vectors from the *plug* Set

The visualization program was also used to examine the data sets and observe whether the input vectors overlapped from one SECF class to another. Because only three dimensions could be displayed at one time, the force vectors were displayed in one window and the moment vectors were displayed in a second window. As a result, the visualization program made it easy to see whether there was overlap in the force vectors or in the moment vectors. However, the visualization program did not display whether there was overlap among all 6 dimensions of data.

The visualization program showed no overlap in the *pentagon* set and almost no overlap in the *plug* set, except for the misplaced vectors discussed. However, in the remaining four data sets, there was overlap in the input vectors. An example can be seen in Figures 22 and 23. The *peg(11)*, *square(17)*, and the *square(8)* sets contained significant overlap or very close sets between some pairs of SECF classes. The *square(8)* set showed particularly strong overlap in the force vectors and in the moment vectors, and the result of the fuzzy classifier reflected this overlap.[2] The peg(9) set contained some overlap between input vector sets but not as significant as the other 3 sets. For the two peg sets, the decreased performance in the fuzzy classifier may have resulted from the combination of the overlapping data sets and the geometric features being misaligned with the sensor frame.

In addition to the fuzzy classifier, the 6 data sets were also used to train the neural network classifier. Results of the neural net classifier are shown in Table VI. It is here that the overlap in the training data sets may have dramatically affected the

[2]The classification results for the *square(8)* set look especially poor at a success rate of only 83.5%. However, two thirds of the missed classifications occurred in one of the eight classes.

training and the performance of the classifier. For the pentagon set, where no overlap was observed, the neural net classifier performed almost as well as the fuzzy classifier. In the plug set, the performance was better than that of the fuzzy classifier. In the *peg(11)*, *peg(9)*, and *square(17)* sets, where there was overlap, the performance of the neural net classifier was worse. For the *peg(11)* and the *square(17)* sets where the overlap was quite significant, the performance was especially bad. As expected, the neural net classifier does not perform well in these cases. However, in the *square(8)* set, the neural net classifier performed remarkably well (99%). Although the visualization program showed distinctive overlap in the force vectors and in the moment vectors, there was apparently significant separation when all 6 dimensions were considered together. That is, even though two SECF's may have produced similar normalized force vectors, the normalized moment vectors were different.

TABLE VI

Neural Net Classifier Results

Set	hidden	tse	epochs	lr	good	misses	unclass.	%success
pentagon	10	9.5	766	0.1	1818	11	16	98.5
plug	3	12.0	13000	0.2	177	1	1	98.9
peg(11)	8	103	600	0.1	1119	192	110	78.7
peg(9)	8	152	65300	0.1	3277	53	254	91.4
square(17)	8	212	200	0.1	6250	833	1368	74.0
square(8)	8	39.6	13000	0.1	1965	16	3	99.0

We can see some significant differences in the training and the performance of the fuzzy classifier compared to that of the neural net classifier. In situations where two different SECF's produce the same set of force vectors and the same set of moment vectors, the generated membership functions will be quite similar for the two SECF classes. Thus, the fuzzy classifier will not perform well in this situation. However, if the same force vector produces different moment vectors for two different SECF's, then the neural net classifier can distinguish between the two classes and will perform well. In a different situation, where the force vector sets (for two different SECF's) and the moment vector sets overlap slightly, the fuzzy classifier may perform much better than the neural net classifier, because of the training. The training of the fuzzy classifier is not as sensitive to slightly overlapping input sets, because membership functions are based strictly on the calculated means and standard deviations of the input vectors in the training set.

After examining the classifier results, we concluded that the neural net classifier may provide better performance if the training method was modified. Because of the poor performance for the *peg(11)*, *peg(9)*, and *square(17)* sets, the classifier was re-trained for these sets using the two-step procedure described in Section IV.D.2. The classifier was first trained using a small set of prototype vectors. The weights and biases learned as a result of the first step were then used as starting weights and biases for the second step. In the second step, the neural net was trained with the complete training set.

The results of the two-step training process are shown in Table VII. The modified training process yielded dramatically improved classification results. The success rates for all of the classifiers and sets are summarized in Figure 43.

To further examine the performance of the SECF classifiers, a timing test was also

TABLE VII

Modified NN Classifier Results

Set	hidden	tse	epochs	lr	good	misses	unclass.	%success
peg(11)	8	87	20100	0.025	1392	17	12	98.0
peg(9)	8	147	5900	0.050	3388	95	101	94.5
square(17)	8	203	1200	0.050	8263	91	97	97.8

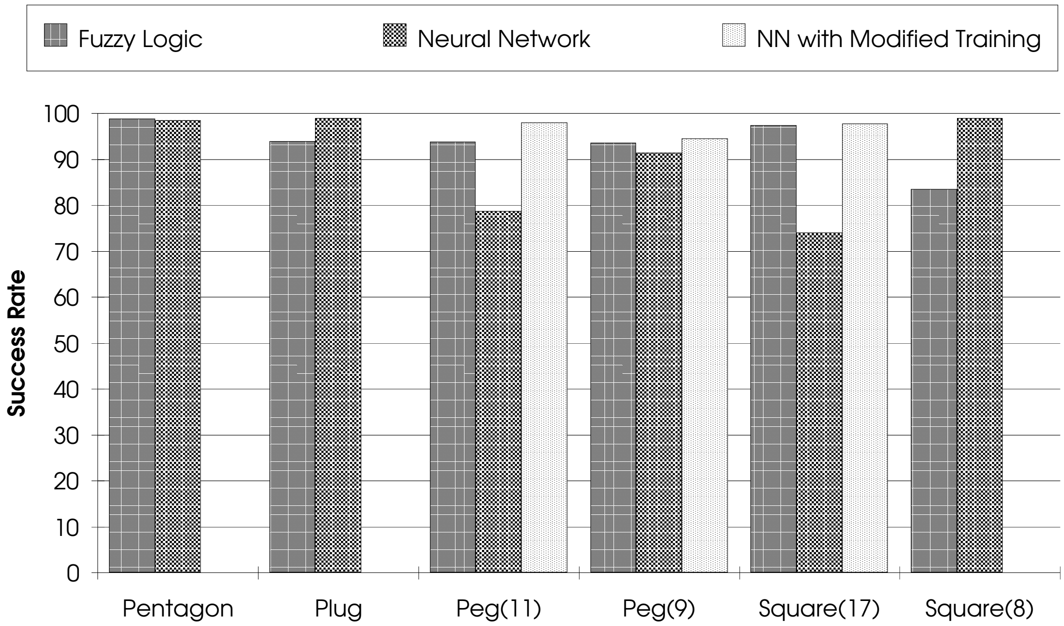


Figure 43. Classifier Performance Comparison

run. We were especially interested in the timing of the fuzzy classifier, because this classifier was used in the remaining experiments. The timing test was run on the 133 MHz pentium PC which functions as the robot controller. For the *pentagon* set, with 11 classes, the fuzzy classifier took 0.539 sec to classify all 1845 data points, which is an average of 0.29 msec per point. Thus, the algorithm is more than fast enough for real-time force-based systems.

One of the more interesting results of the classifier tests is the use of the non-rigid part (the *plug*) and the success in identifying SECF classes. Even with a small training set, the results were surprisingly good, especially for the neural net classifier. Flexible parts such as the *plug* are difficult to model; consequently, the sensor-based approach used here may be especially useful.

B. Learning the Sequence of SECF's and the Transition Trajectories

In this section, we will use the results of the classifier training for one of the 6 data sets presented above, and show the experimental results of learning a sequence of SECF states and the transition trajectories that drive the system from one SECF to the next desired SECF. In the experiments, two different SECF sequences have been demonstrated. The two sequences have the same goal SECF state, but they provide different strategies (i.e., different sequences) that can be used to achieve the same goal. The results of the two learned sequences are merged into one FSM that can be used to achieve robust execution of the skill.

The data set used for these experiments is the *square(17)* set described in the previous section and also in Section IV.E. For convenience in graphing the results, each class in the set is assigned an index as well as a symbolic representation (in this case, a name). The names were designed to be meaningful to a human teacher as she observes the demonstration process, because they are displayed as feedback during the actual demonstration.

The complete set of classes is given in Table VIII. For the *square(17)* set, some of the classes were trained with a single surface of the workpiece making contact with a flat surface in the environment (e.g., *right* and *left* sides). Also, some of the classes were trained with two surfaces of the workpiece making contact with two corresponding surfaces of the environment (e.g., *right bottom* and *left bottom*). In addition, some classes were trained with the three corner surfaces of the workpiece making contact simultaneously with a concave corner of the environment (e.g., *right front bottom* and *left front bottom*).

The *square(17)* set was trained before we decided on which SECF sequence to use for the experiments. In the design of the experiments, we discovered that the existing set did not include all of the SECF's that we wanted to use for the skill. To avoid having to train the classifier for additional SECF's, we decided to use the *square(17)* set as originally trained but extend the interpretation for some of the SECF classes. For instance, we might want to construct a skill sequence which uses the right front edge of the workpiece instead of the right front surfaces, as shown in Figure 44. The force vectors for these two SECF's should be similar, because the force sensor measures the resultant force vector of all forces acting on the workpiece. As a

TABLE VIII

SECF Indices for the *square(17)* Data Set

Index	SECF
-1	unclassified
0	no contact
1	bottom
2	front
3	front bottom
4	left
5	left bottom
6	left front
7	left front bottom
8	left rear
9	left rear bottom
10	rear
11	rear bottom
12	right
13	right bottom
14	right front
15	right front bottom
16	right rear
17	right rear bottom

result, we might be able to interpret the *right front* SECF as the right front edge even though it was not actually trained this way. Similarly, we may be able interpret the *right front bottom* SECF as the right front edge and bottom surface, even though it was actually trained using 3 surfaces. In a sense, it provides an additional test of the training and classification method, to see whether we can achieve acceptable results even if the training condition is not exactly the same as the execution condition.

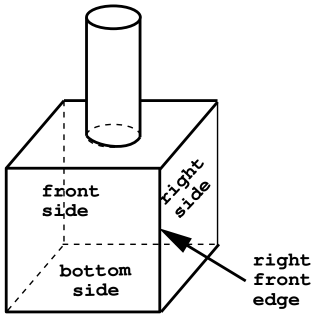


Figure 44. Workpiece Showing the Right Front Surfaces and the Right Front Edge

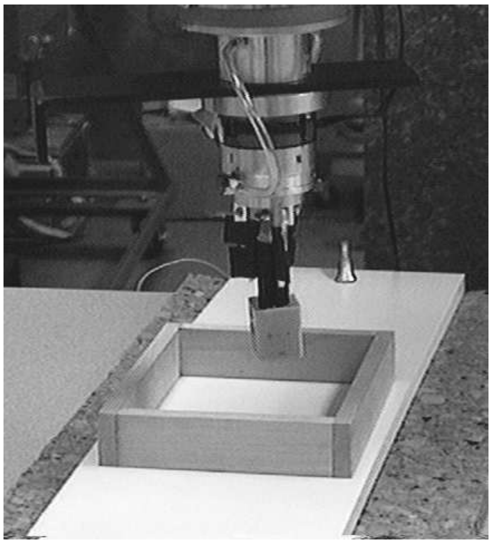
The demonstration of the skill was performed using the PHANToM haptic interface, as described in Chapter V. The human teacher was located in front of the robot, with full view of the operation. In the mind of the teacher, there was a specific intended SECF sequence that was part of the teacher's skill model, and this is the sequence that we would like the robot to learn.

The environment is an open, rectangular, wooden box which is attached to a formica-covered wooden board. The open box was chosen because it provides a variety of constraint surfaces, so that single, double, and triple surface combination SECF's can be achieved. As a matter of convenience for observing the learned skill, the height of the box matches the height of the square workpiece, so an observer can easily tell if the workpiece is touching the bottom. The wooden box is made of soft, bare wood. Although it has been sanded, it still has a relatively high coefficient of friction. This was intended, as we specifically wanted to include the effects of friction in the experiments. The attached board can be clamped to the worktable in a variety of positions and orientations so that different starting conditions can be generated. A foam pad was used beneath the board for safety reasons.

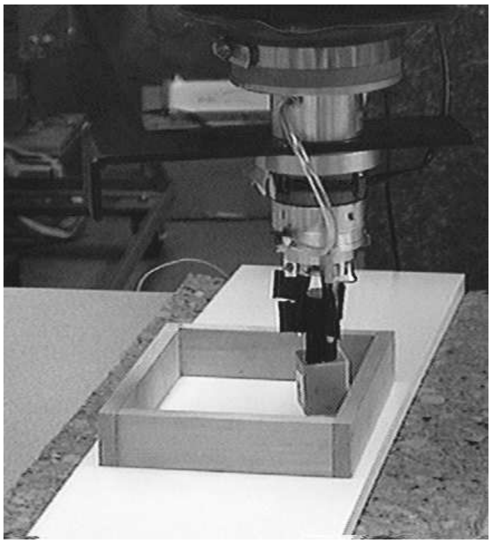
Snapshots of the first sequence, which we will call *Sequence A*, are shown in Figure 45. Starting from a *no contact* state, the intended sequence is *right front*, *right* side, and *right bottom.* The sequence was demonstrated 3 times, and the final demonstration was used to learn the SECF sequence and transition trajectories. The demonstration profiles are given in Chapter IV, in Figures 27- 31 and 33- 36, along with a description of the learning process used. Graphs of the first and second demonstration trials are included in Appendix A.

The fuzzy logic-based classifier is used to identify SECF states from the force and moment profiles, as discussed in Chapter IV. The output of the classifier is filtered and presented to the human teacher for refinement. The teacher selects the desired events which define the intended SECF sequence, and these events are used to extract linear segments which comprise the transition trajectories.

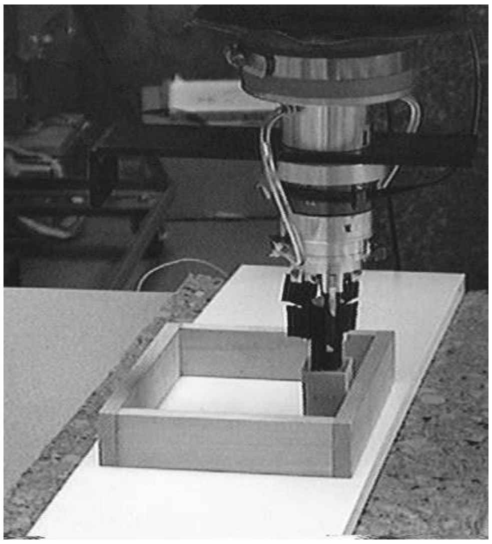
Snapshots of the second sequence, which we will call *Sequence B* are shown in Figure 46. In this case, the intended sequence of SECF steps is *no contact*, *bottom*, *right front bottom*, and *right bottom. Sequence B* is designed to provide a successful skill strategy for the cases where the position of the environment fixture is shifted so



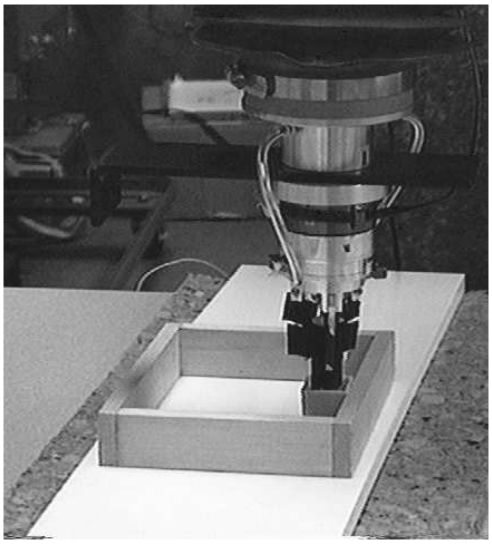
1: Start Position



2: Right Front



3: Right Side



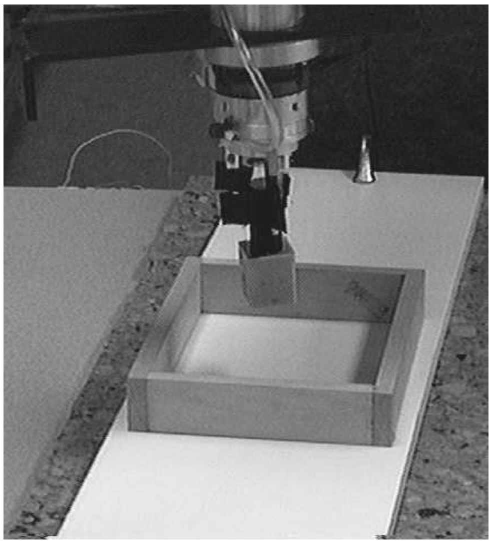
4: Goal Position is Right Bottom

Figure 45. The SECF Steps for Sequence A

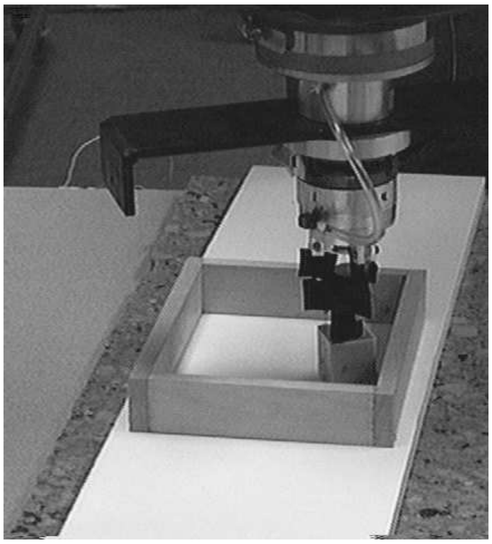
that the workpiece contacts the bottom surface first, instead of the right front edge.

The sequence was again demonstrated three times and the third demonstration was used for skill acquisition. The demonstration profiles are shown in Figures 47, 48, and 49. The profiles for the remaining *Sequence B* trials are included in Appendix A. The same learning process was used, as discussed in Chapter IV. In this case, there were also some unintended SECF's, which were eliminated by the human teacher as part of the interactive teaching environment. The teacher selected the desired events, and these are shown in the plots as vertical hash lines. As before, the selected events are used to provide time steps in extracting linear segments which define the transition trajectories.

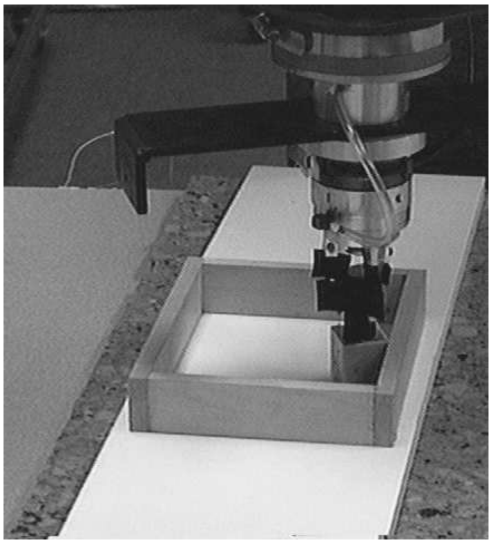
To provide a robust execution of the learned skill, in spite of position and orientation variations, *Sequence A* and *Sequence B* are merged into one FSM, as shown in Figure 50. The two sequences are combined so that *Sequence A* is the initial sequence. That is, until an SECF class is encountered, *Sequence A* is used, and the FSM attempts to drive the robot toward the *right front* SECF. However, if the *bottom* SECF is encountered, then control shifts to the equivalent of *Sequence B*. Likewise, if any SECF is encountered out of order (e.g., *right* is encountered before *right front*), the FSM is still able to drive the control through a successful skill sequence. To accommodate a greater variety of starting positions and orientations, additional sequence strategies could be demonstrated and merged into the FSM. Note that the selection of *Sequence A* as the initial sequence is arbitrary.



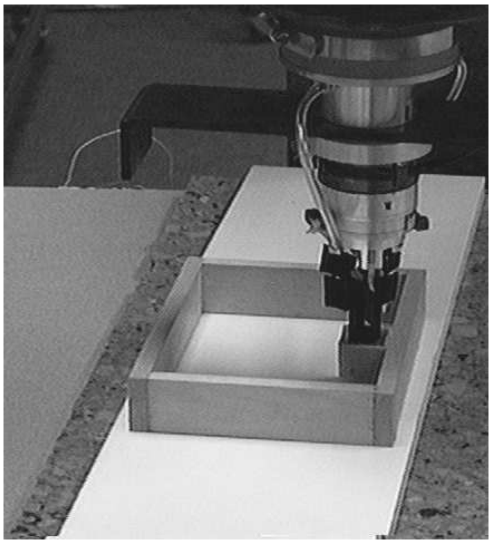
1: Start Position



2: Bottom Surface



3: Right Front Bottom



4: Goal Position is Right Bottom

Figure 46. The SECF Steps for Sequence B

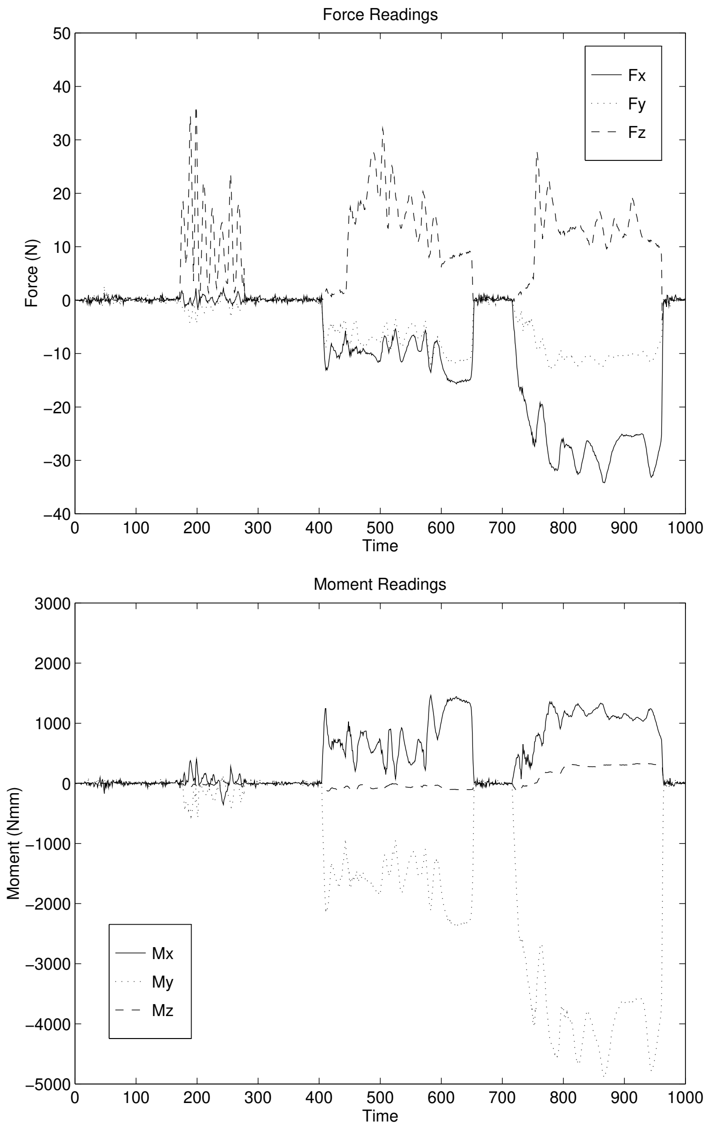


Figure 47. Force and Moment Profiles for Sequence B Demonstration

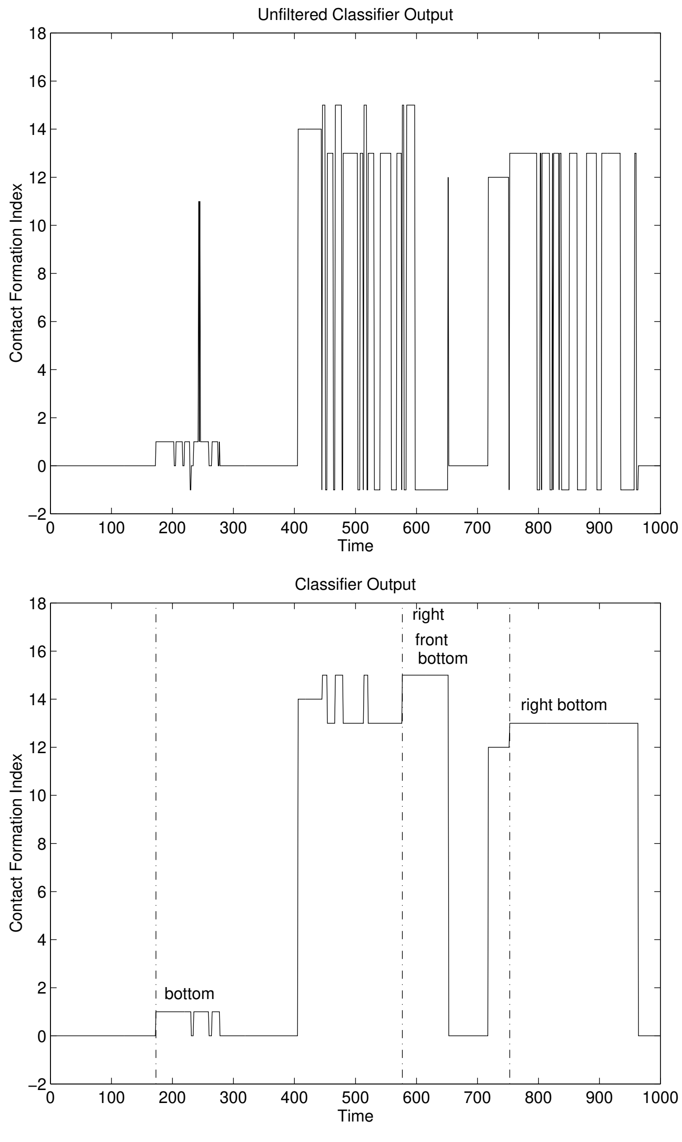


Figure 48. Classifier Output for Sequence B Demonstration

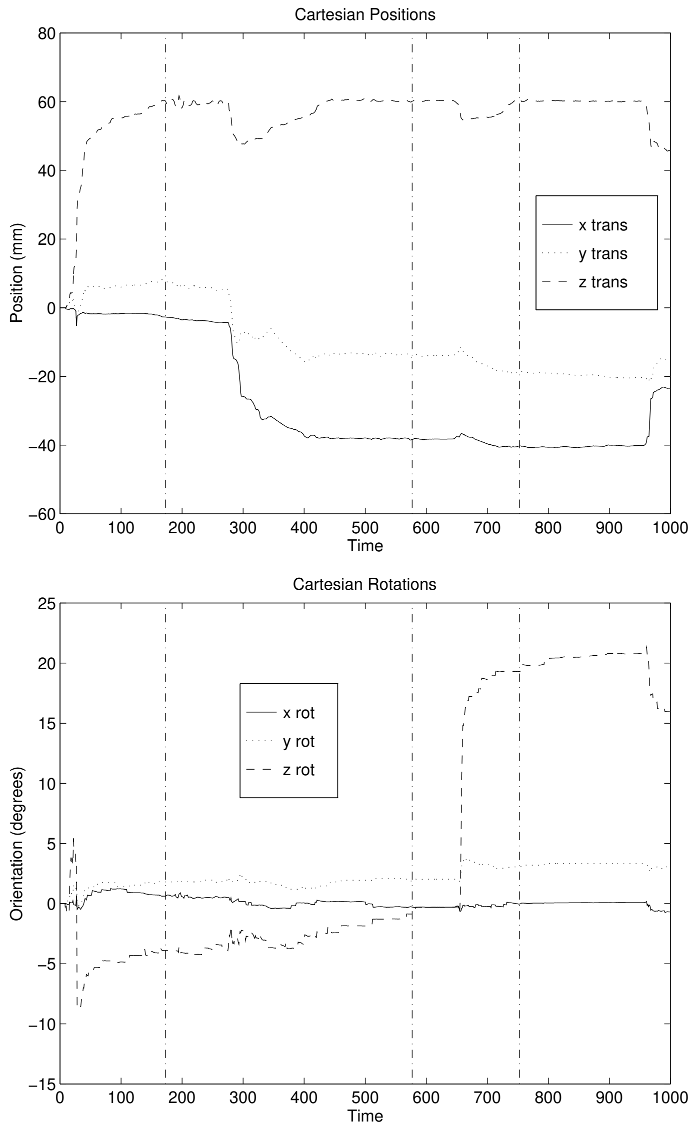


Figure 49. Translation and Rotation Profiles for Sequence B Demonstration

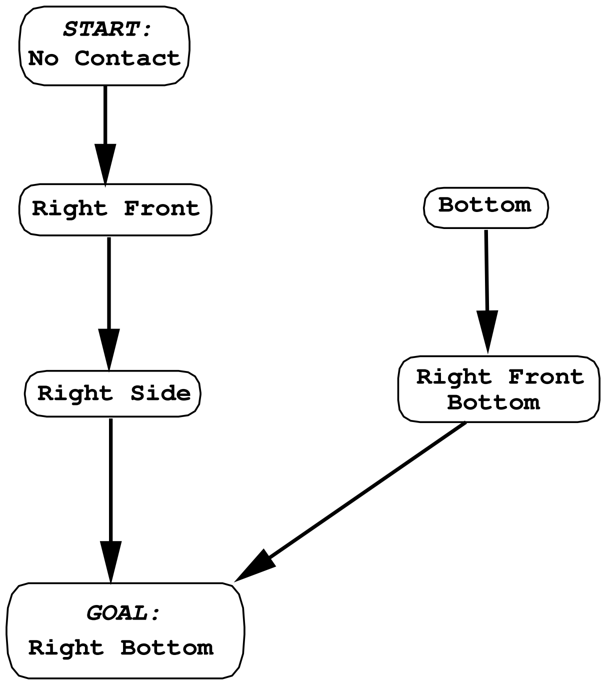


Figure 50. The Merged FSM with Sequence A and Sequence B

C. Execution of the Learned Skill

In this section, we present results of test runs performed where the robot autonomously executed the skill using the learned skill model. The same workpiece and environment fixture were used for both skill learning and for autonomous skill execution. The starting position of the robot, with the grasped workpiece, was the same in all tests; the workpiece was not contacting the environment. To test the execution of the skill in different starting conditions, the position and orientation of the environment fixture was varied between test runs.

A data logging function was added to the robot control program to monitor the performance. The logged data included the force sensor signals, the robot end effector position and orientation, and the classified SECF. All data was logged at a frequency of 50 Hz. At the end of the test run, if the goal SECF was reached, data was logged for another 20 sample periods (400 msec), and the program then halted. During the last 400 msec period, no additional reference velocity commands were issued. However, the force control algorithm was still in effect during this time, and new position commands may have been generated as a result of the force limits.

The tests were performed in two groups. In Test Suite I, attempts were made to execute *Sequence A*. In Test Suite II, the starting position and orientation of the environment fixture was varied in such a way as to induce a variety of sequences, including *Sequence B*. Representative runs from both groups will be discussed here. Graphs of the complete set of test runs can be found in Appendix A.

1. Test Suite I

There were 10 test runs made as part of Test Suite I. The first 5 runs were performed without the translational scale factor discussed in Section IV.G. Figures 51, 52, and 53 show the results of a typical test run. In all 5 test runs, the skill was successfully completed. However, as Figure 51 shows, the execution of the skill produced unstable force signals, as the SECF state oscillated between the *no contact* class and the *right front* or *right* classes. In addition, there were several unclassified conditions encountered. The unstable condition occurred during a transition period that should have been predominantly rotational movement. In the actual skill execution, the robot was being driven by a strong translational component. The unstable conditions also carried over to the last 400 msec period, where the FSM stopped and tried to hold the termination condition. At this point, unintended SECF's were encountered, probably as the result of a bouncing action. This can be observed in the classifier output plot of Figure 52, where the SECF oscillates between the *left bottom* and the *bottom* SECF's.

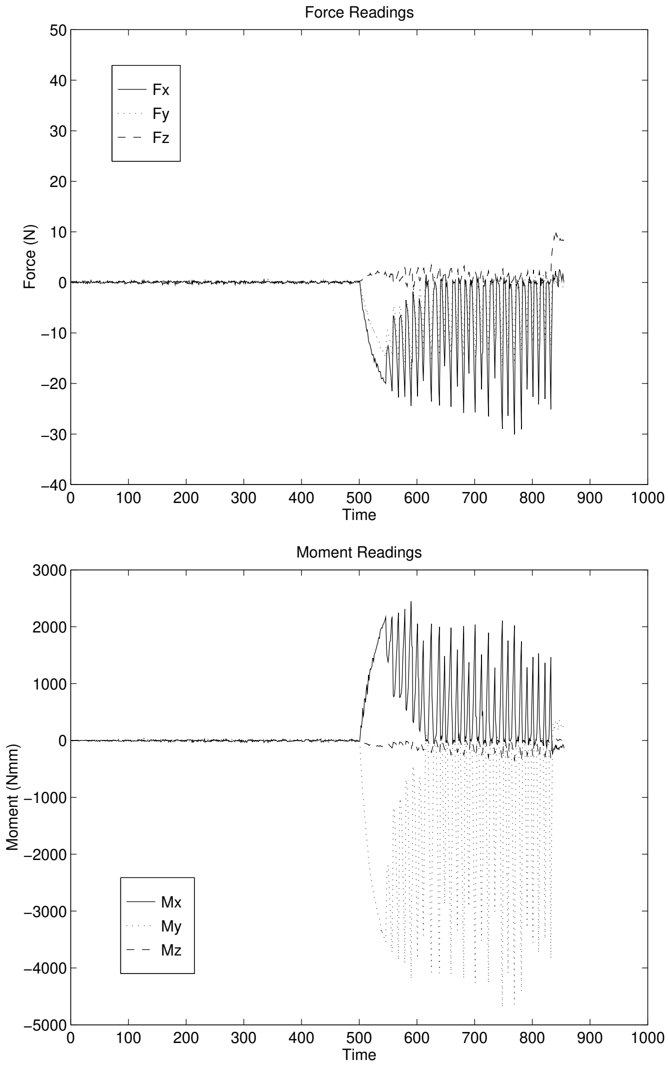


Figure 51. Test Run I.b: Without the Translational Scale Factor (Force and Moment Profiles)

The original intent was that this problem could be eliminated by adjusting the

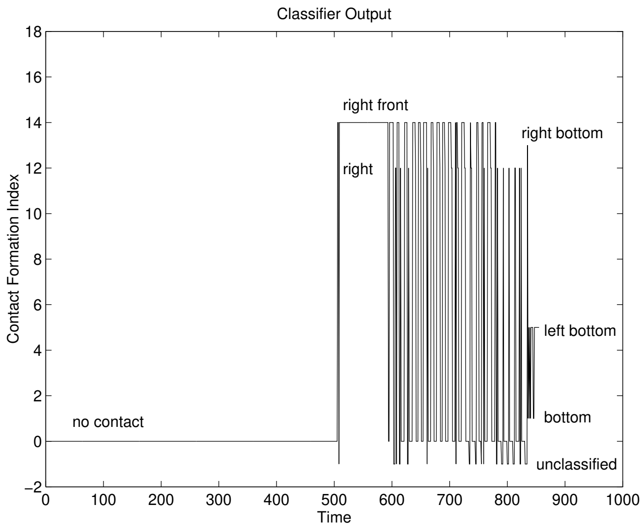


Figure 52. Test Run I.b: Without the Translational Scale Factor (Classifier Output)

parameters in the force control algorithm. In the first 5 test runs of Suite I, the parameters were varied, but the results were essentially the same. In the end, the conclusion was that, for a force control loop running at 100 Hz, the constant speed assigned to the velocity commands (only 5 mm/sec) was too large to produce stable execution. Two possible solutions, then, were to either increase the frequency of the force control loop or to decrease the speed of the robot. The first solution was not an option, because of the LynxOS timing resolution. The second solution was not desirable because it would have forced the robot speed to be unacceptably slow.

As an alternate solution, the translational scale factor was introduced. The last 5 runs were all performed with the translational scale factor. Figures 54, 55, and 56 and Figures 57, 58, and 59 show the results of two of the test runs. In both cases, the *no contact* condition was encountered, but the skill model was able to accommodate this unexpected condition and successfully complete the skill. Indeed, the skill was

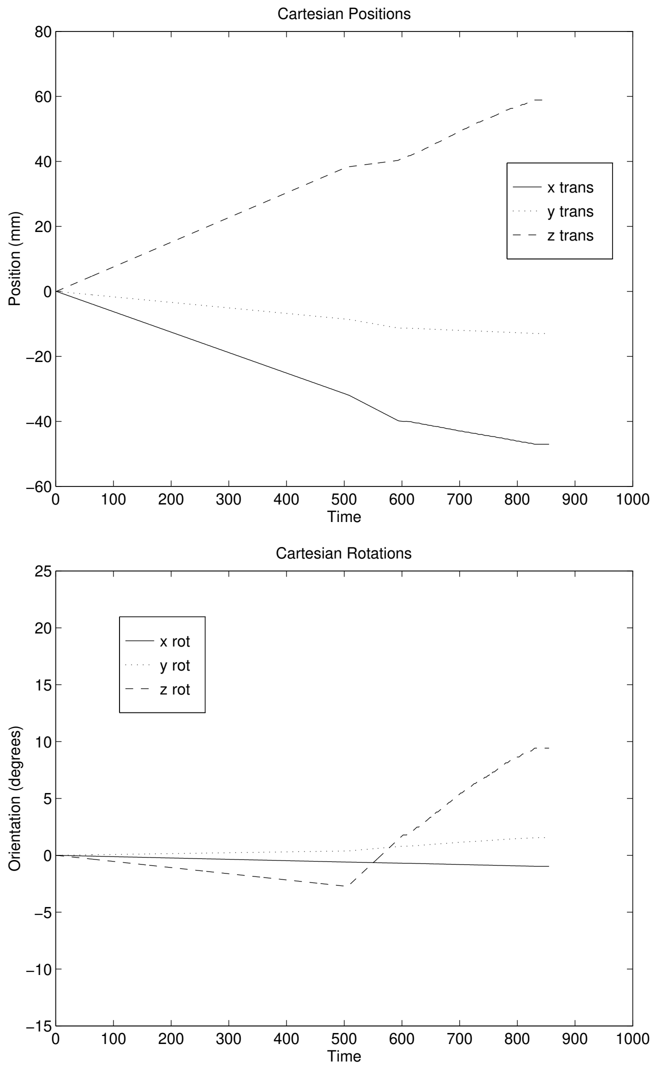


Figure 53. Test Run I.b: Without the Translational Scale Factor (Translation and Rotation Profiles)

successfully completed in all 5 test runs, even though the starting position and orientation of the environment fixture was changed. These effects can be seen in the test results. In Figure 59, the rotation profile shows a z rotation of less than 10 degrees compared to Figure 56 where the z rotation is greater than 20 degrees.

2. Test Suite II

In this test suite, we introduced more variations in the starting position and orientation of the environment fixture, in an attempt to produce additional sequences of SECF states. The first 4 test runs are all attempts to execute *Sequence B*. Figures 60, 61, and 62 show the results of a representative test run. From the SECF profile, it appears that the goal SECF was reached. However, during the execution, the robot program stopped before completing the last rotation from *right front bottom* to *right bottom*. The classifier incorrectly identified the *right front bottom* SECF as the goal *right bottom* SECF. The same thing occurred in all 4 of these first test runs.

It appears that the incorrect classification has resulted from training the classifier with the right front bottom surfaces and then trying to extend this class to include contact on the right front edge and bottom surface. The resultant force and moment vectors are outside of the trained class. This can be seen in the SECF profile as unclassified regions. During the skill execution, the workpiece also slides along the environment fixture as it rotates into the final SECF. This produces frictional forces which further drive the resultant force and moment vectors outside of the class. We might have predicted this problem by looking at the unfiltered classification output

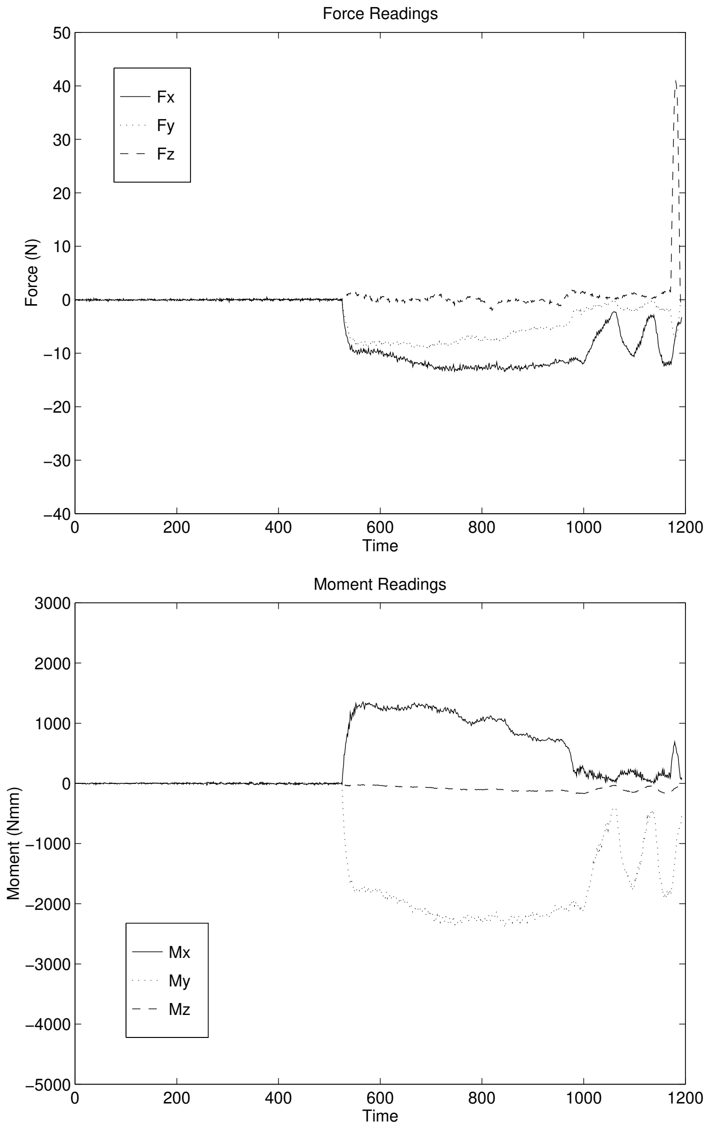


Figure 54. Test Run I.i: Successful Sequence A (Force and Moment Profiles)

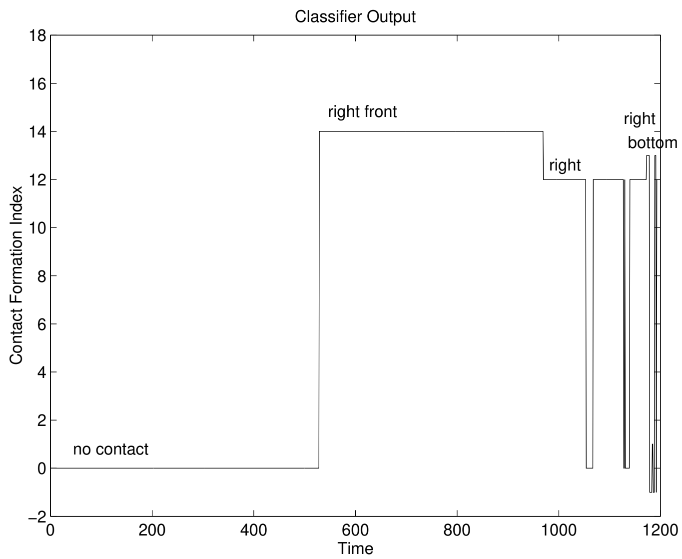


Figure 55. Test Run I.i: Successful Sequence A (Classifier Output)

for the *Sequence B* demonstration, as shown in Figure 48. Beginning at time step 600, there is an unclassified region which lasts for approximately 50 sample periods (1 second), which should have been classified as *right front bottom*.

The conclusion was that the classifier should be retrained with a more realistic training set for the *right front bottom* class, as it is being used in this skill sequence. This was done by collecting training data while the workpiece was sliding along the surface of the environment fixture, thus producing high friction forces. Five additional test runs were then performed. In all 5 cases, the skill was successfully completed, in spite of additional position and orientation variations being introduced. Figures 63, 64, and 65, Figures 66, 67, and 68, and Figures 69, 70, and 71 show the results from 3 of the test runs and provide a variety of skill sequences. Unexpected SECF's were encountered during the skill execution, but the skill model was able to accommodate the variation and still produce a successful skill completion.

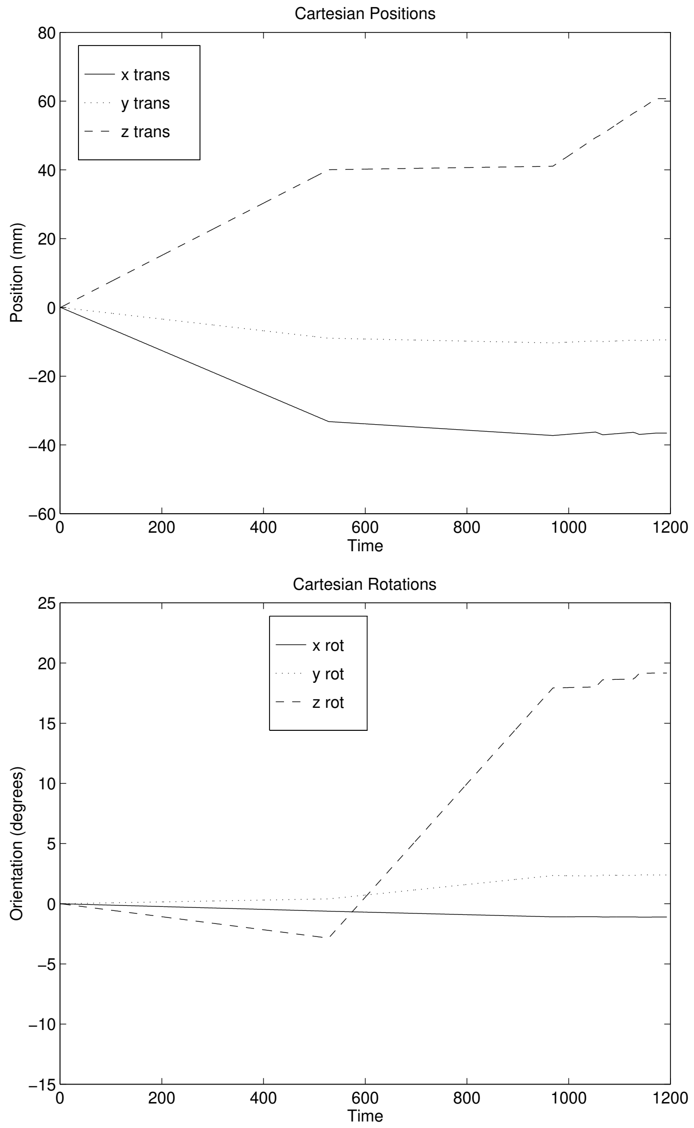


Figure 56. Test Run I.i: Successful Sequence A (Translation and Rotation Profiles)

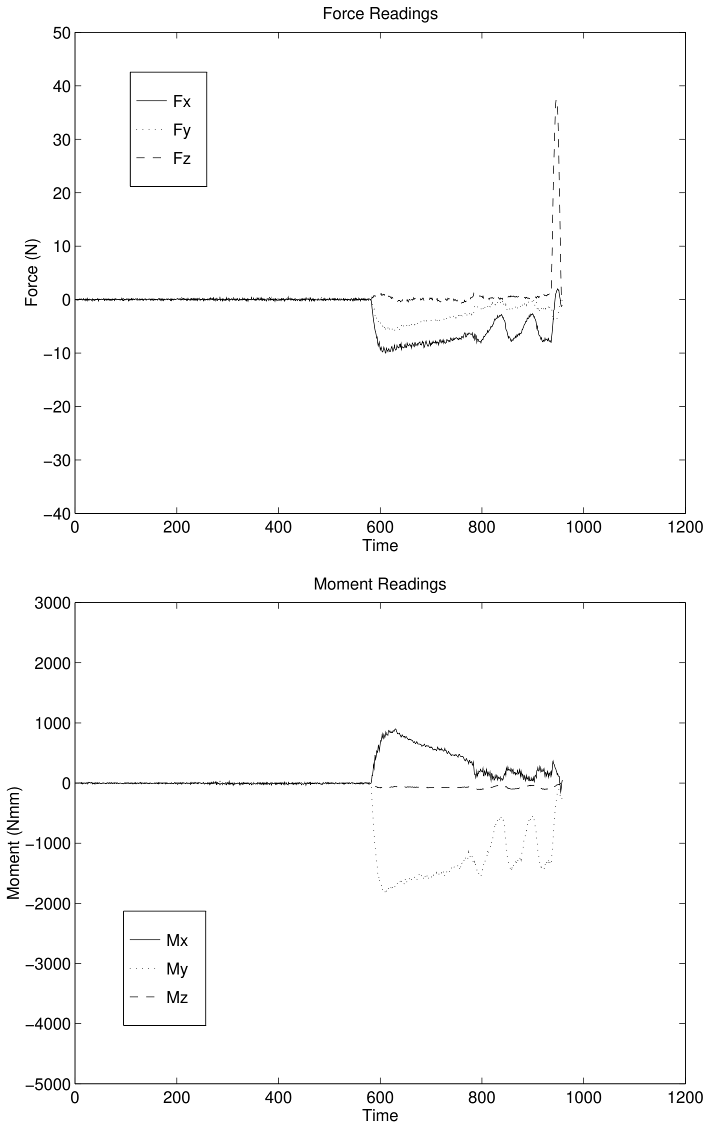


Figure 57. Test Run I.j: Successful Sequence A (Force and Moment Profiles)

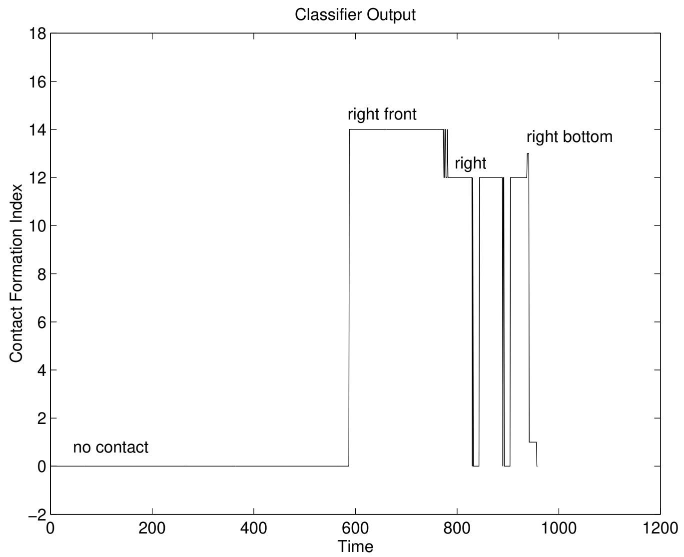


Figure 58. Test Run I.j: Successful Sequence A (Classifier Output)

D. Discussion

Although the experiments were generally successful, there are limitations that became apparent in the course of the testing. One of the biggest potential limitations results from trying to use signals from a robot wrist-mounted force sensor as the sole sensory medium for identifying the contact state. The force sensor can only measure resultant force and moment vectors which occur as a response to contact between the environment and the grasped workpiece. The same resultant vectors may occur with different contact formations, and there may be instances where the same resultant vectors occur for different single-ended contact formations. For the test cases included here, we achieved quite reasonable results for distinguishing between different SECF's, even when the vector sets overlapped for different classes. However, we did not prove that we could distinguish between SECF's in all cases. There may be some instances

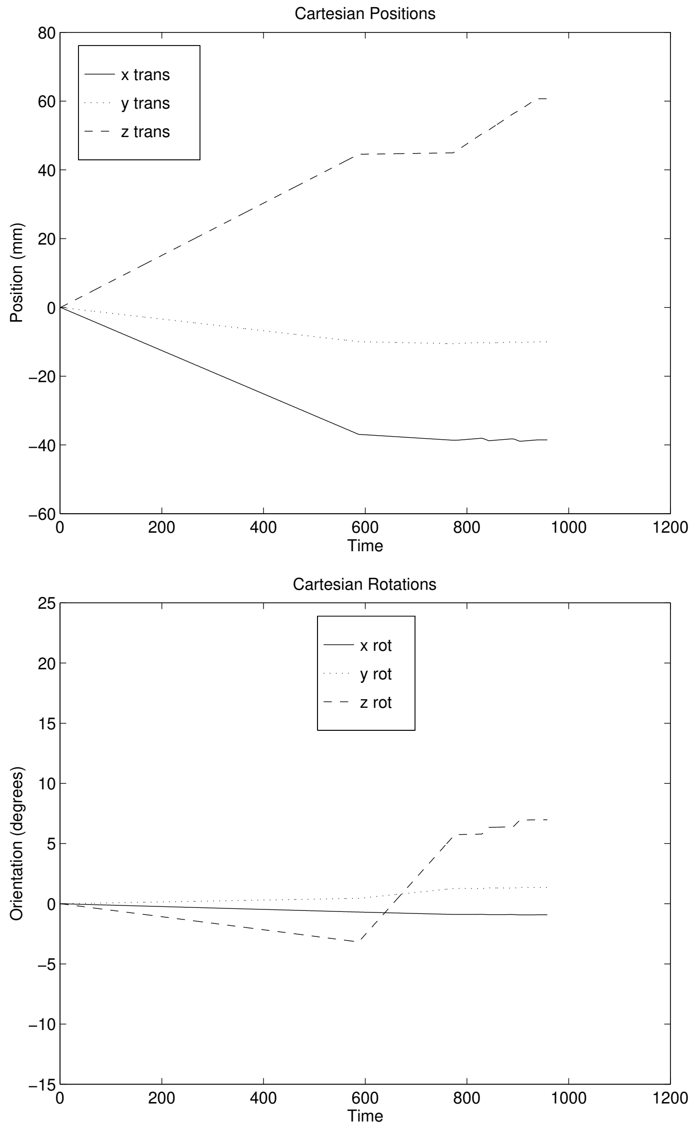


Figure 59. Test Run I.j: Successful Sequence A (Translation and Rotation Profiles)

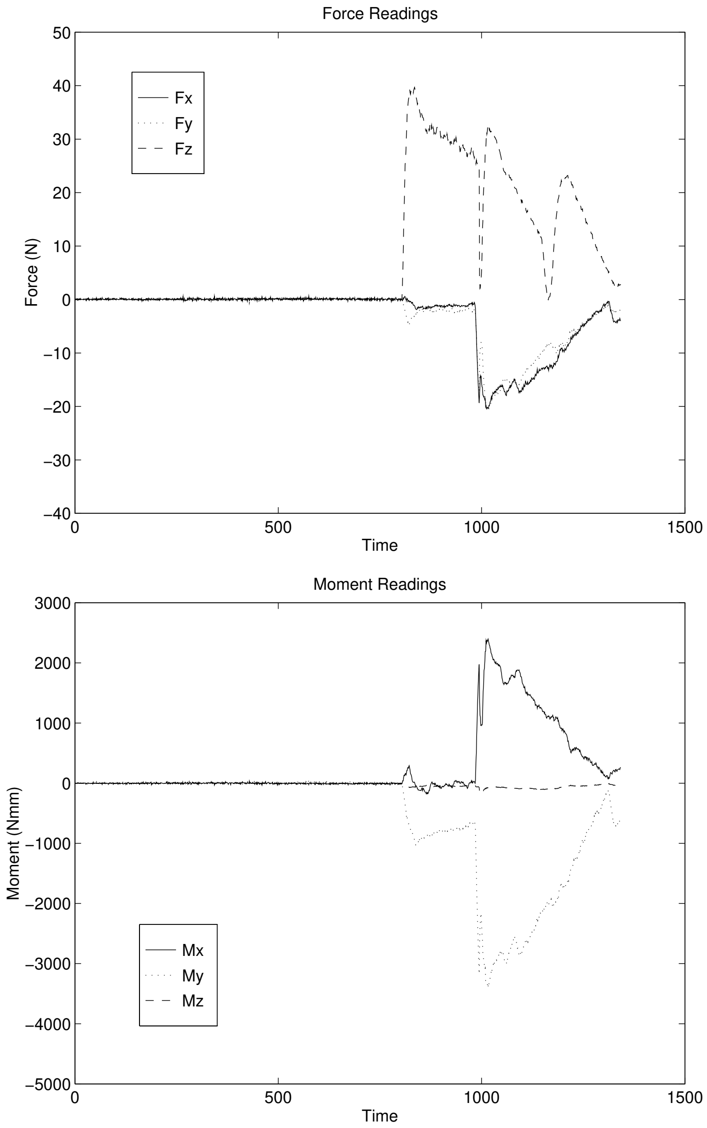


Figure 60. Test Run II.b: Incomplete Last Rotation (Force and Moment Profiles)

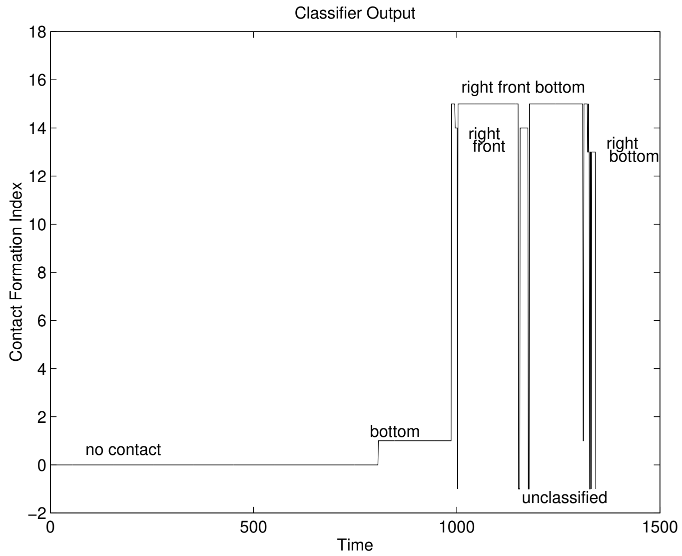


Figure 61. Test Run II.b: Incomplete Last Rotation (Classifier Output)

where additional sensory information will be required for reliable classification.

Another potential problem results from the use of supervised learning. The approach taken here, where the SECF classifier is trained before the skill demonstration, assumes that the operator knows which SECF classes to include in the set. While that may be trivial for simple skills, it may not be so obvious for complex skills.

Another factor that continues to complicate the issue is the presence of friction. If significant friction will be present in the skill execution environment, then it must be included when collecting the training data for the classifier. In some cases, this means that training data should be collected while the workpiece is sliding along a surface in the environment, so that force vectors are on the boundary of the friction cone. This was done in the retraining of the classifier so we could achieve an accurate representation of the *right front bottom* class.

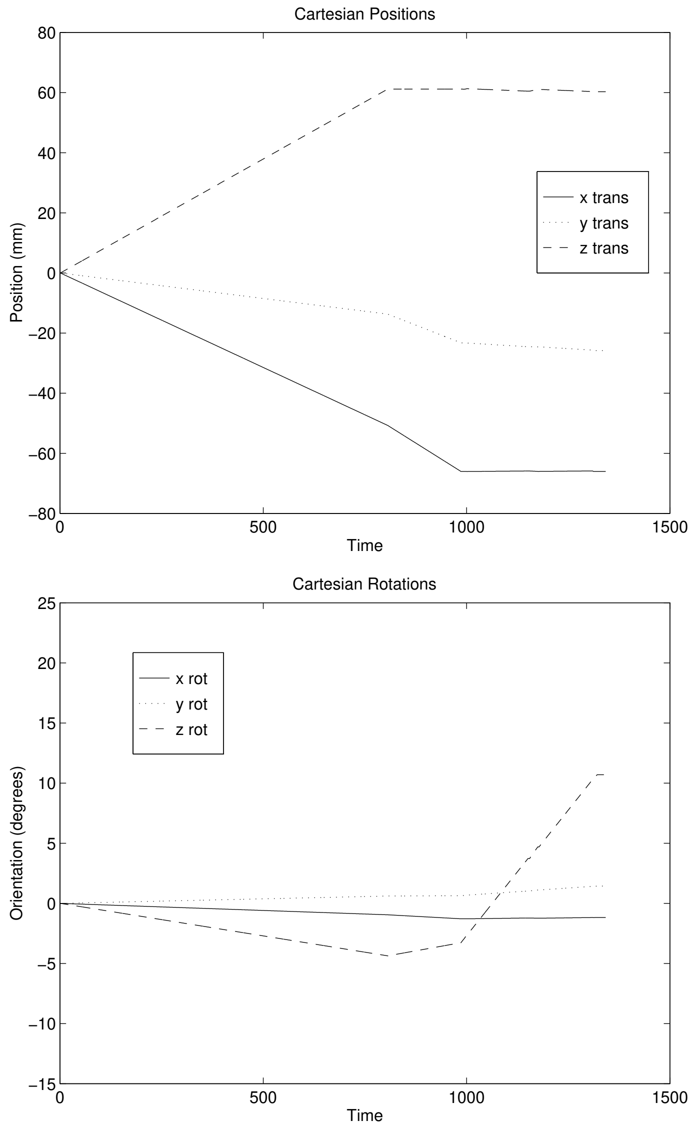


Figure 62. Test Run II.b: Incomplete Last Rotation (Translation and Rotation Profiles)

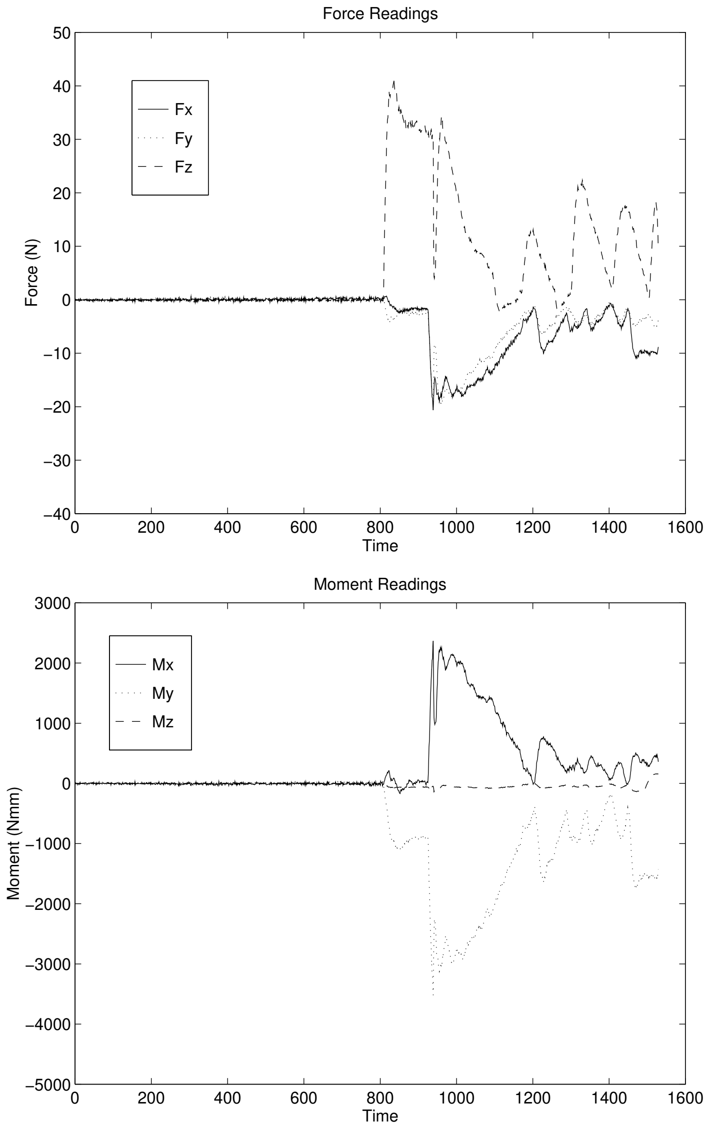


Figure 63. Test Run II.e: Successful Sequence B (Force and Moment Profiles)

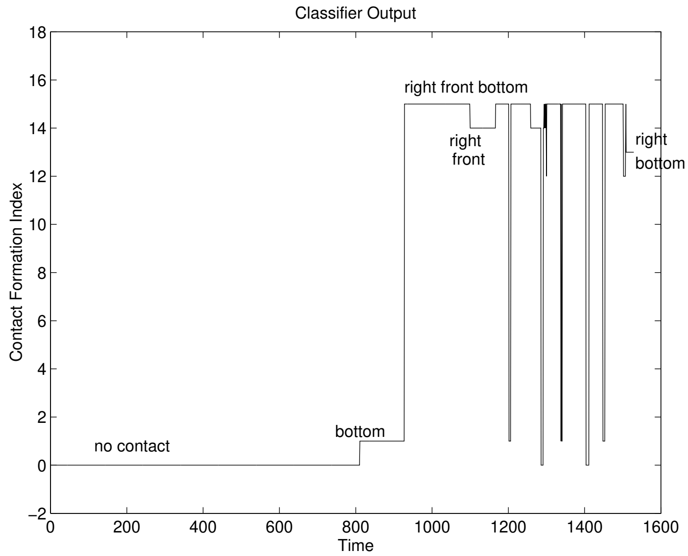


Figure 64. Test Run II.e: Successful Sequence B (Classifier Output)

In general, the collection of good training data sets was more difficult than expected. The significant point is that the training data for the classifier should be collected in a condition as close as possible to the condition that will be encountered in the execution environment. This will ensure that the classifier will be trained to handle the same real-world, difficult-to-model effects that will be encountered during skill execution.

While it is possible to collect separate training data (for a known SECF) in this manner, it may be easier to collect training data while actually performing an entire skill sequence. Although this has not yet been attempted, it may be possible to use an automatic clustering algorithm to segment an entire profile of data into individual force-based clusters. The clusters of vectors would then be used to train the classifier. This approach would ensure that actual conditions, such as friction, were included in

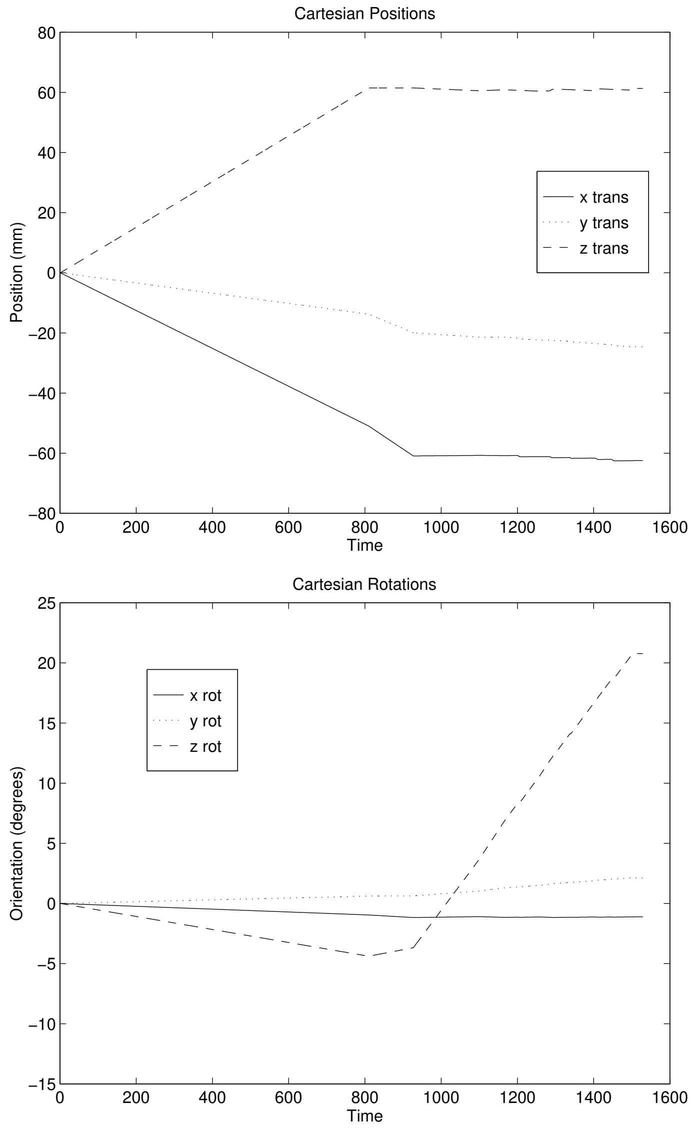


Figure 65. Test Run II.e: Successful Sequence B (Translation and Rotation Profiles)

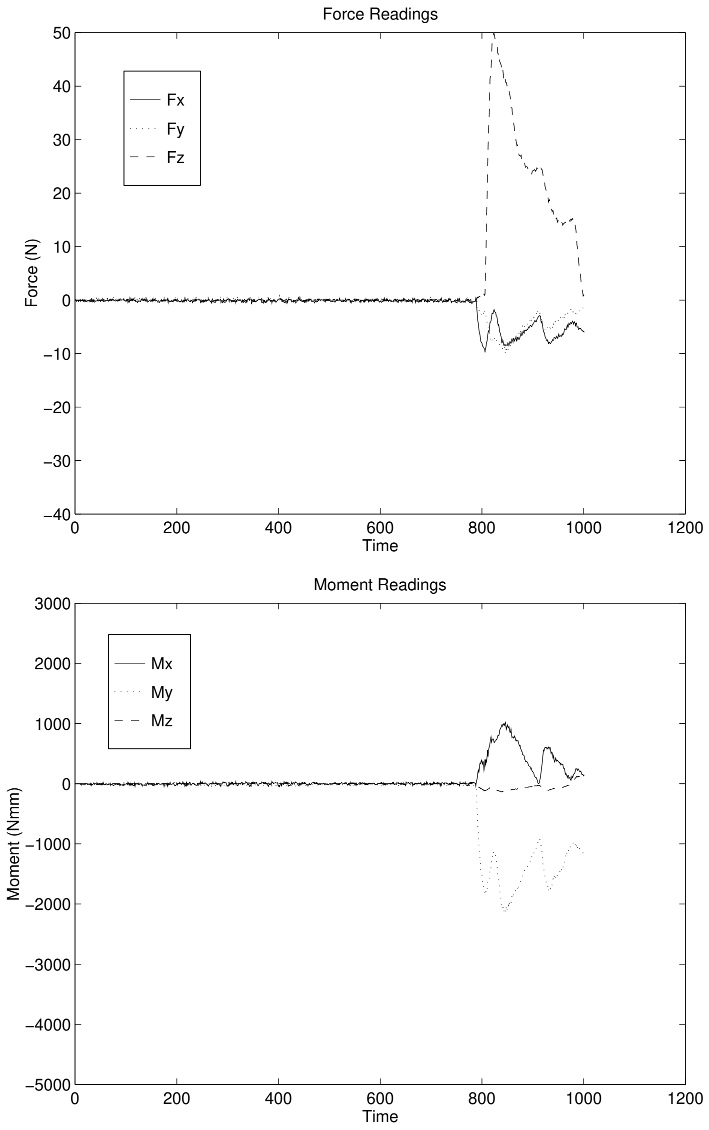


Figure 66. Test Run II.h: A Different Successful Sequence (Force and Moment Profiles)

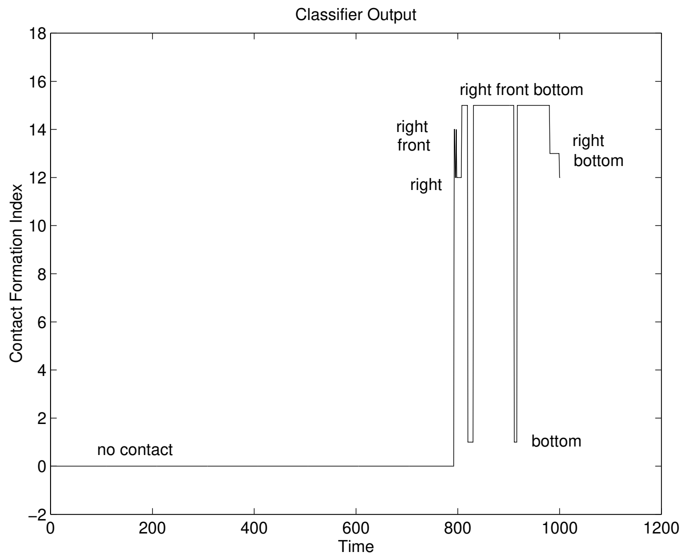


Figure 67. Test Run II.h: A Different Successful Sequence (Classifier Output)

the training data. It would also eliminate the problem of determining which SECF's to include in the class set. The tradeoff is that the SECF classes would not be classified on-line during the actual demonstration, so this feedback would not be provided to the operator.

Another limitation showed up in the implementation used for the robot force control. First, the force control loop should have been much faster, at least 600-800 Hz. This would have supported a faster speed for the robot as well as more complex skills, and would have avoided the rather large force spikes that occurred sometimes, especially in the z-direction. A faster force control loop may also have eliminated the need for the translational scale factor. Another problem was that the goal condition was not always held at the end of the skill, even for the brief 400 msec period; a faster force control loop may have alleviated this problem as well.

In addition, there are limitations to using a force limit control algorithm with con-

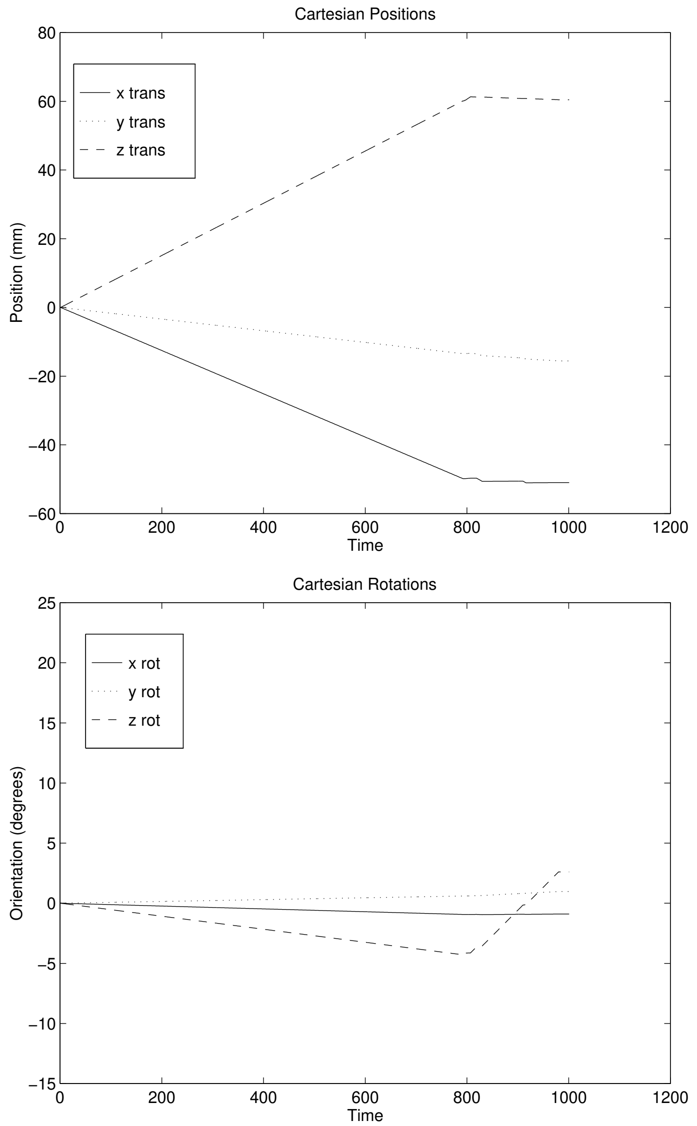


Figure 68. Test Run II.h: A Different Successful Sequence (Translation and Rotation Profiles)

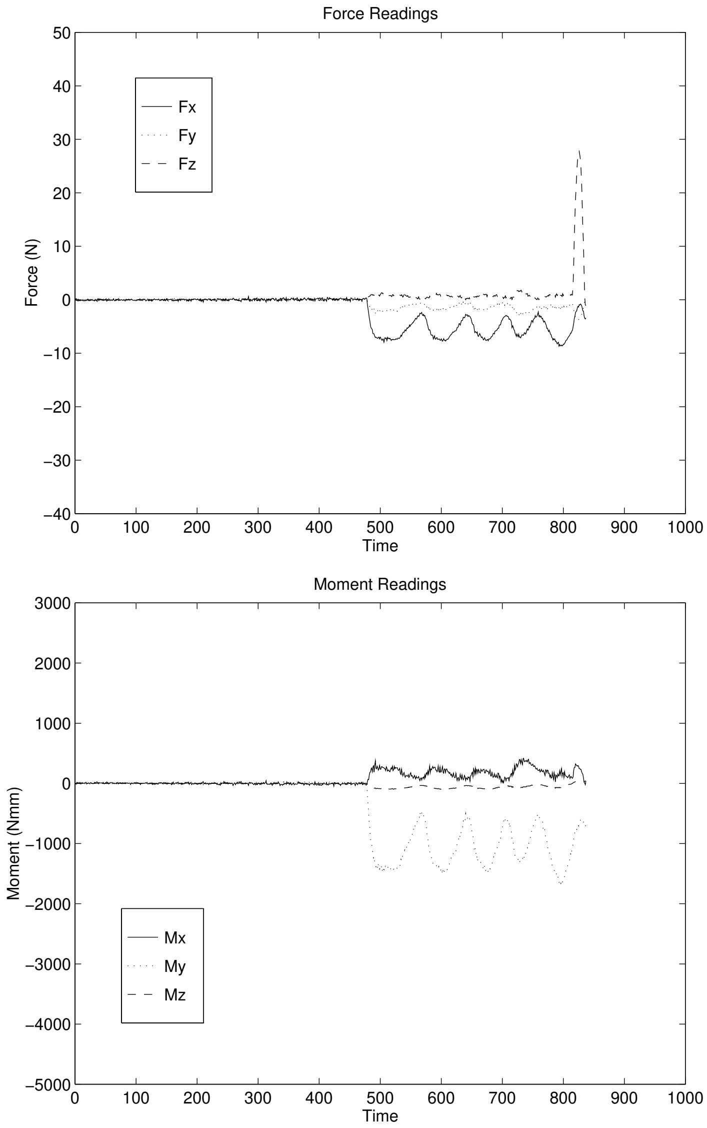


Figure 69. Test Run II.i: Another Successful Sequence (Force and Moment Profiles)

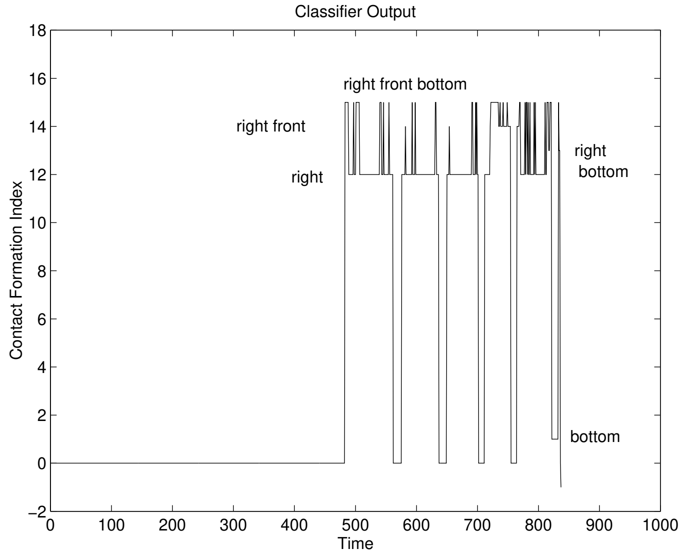


Figure 70. Test Run II.i: Another Successful Sequence (Classifier Output)

stant parameters throughout the skill sequence. For instance, this simple algorithm does not try to maintain a constant force if the workpiece breaks contact with the environment. Certainly, a desirable extension to this work would include changes in the force control parameters for each transition between SECF steps. This may be especially useful for more complicated skills, where it may be necessary to maintain contact during the transition period. For example, a hybrid force/position control algorithm could be used where the direction of force control (and perhaps magnitude of force) changes at each transition step.

Perhaps the biggest surprise in the results presented here is that the skill acquisition method works remarkably well, in spite of the limitations and problems. Although the experiments involved a relatively simple skill, we feel the results validate the assembly skill model. The event-based approach proved to be a good idea and useful in accommodating variations in the environment. The interactive teaching

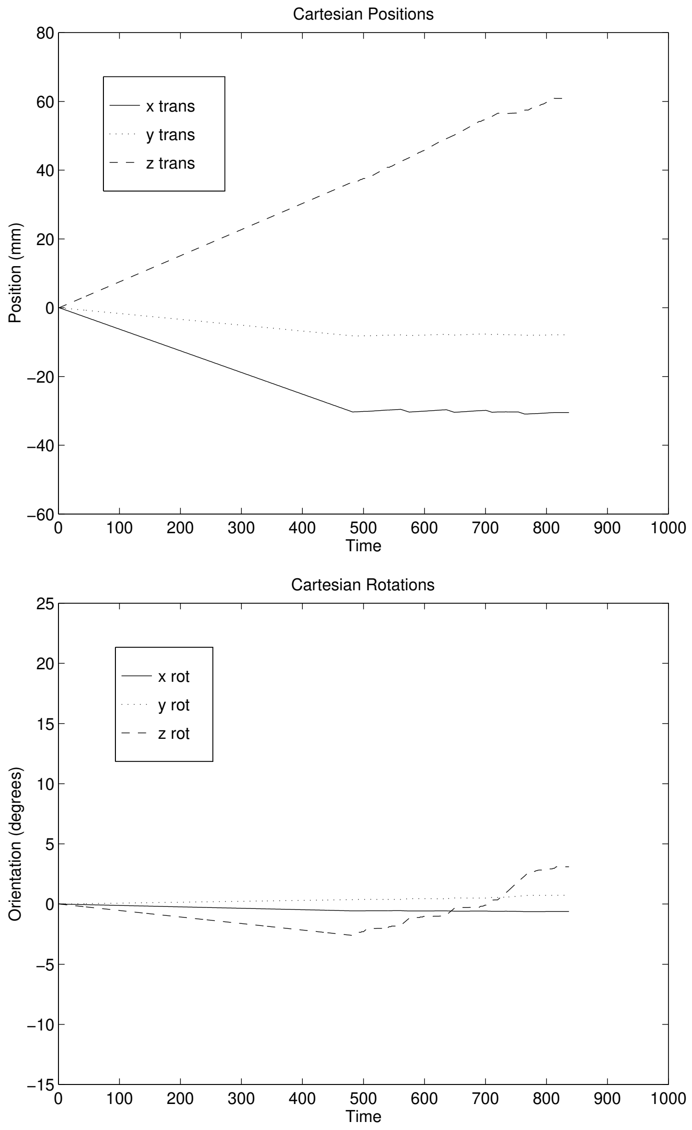


Figure 71. Test Run II.i: Another Successful Sequence (Translation and Rotation Profiles)

environment also proved to be a good idea, especially in letting the teacher select the events that would be used in skill acquisition. Our experience is that this made the teaching process a great deal easier because the skill demonstration did not have to be done perfectly. In the end though, the key element is the successful classification and use of the SECF classes, because this is what defines the events and makes the rest of the skill acquisition possible.

CHAPTER VII

CONCLUSION

Robots have been used successfully in structured settings where the environment is well-controlled and the tasks are repetitive. This research has been inspired by the vision of intelligent robot tools and assistants working successfully in unstructured environments and not necessarily just in repetitive tasks. To support such a vision, the robot may need task refinement, in the form of reprogramming, possibly done at the work site with instruction provided by a human user rather than a specialized engineer. The current difficulty in programming robots has remained one of the key barriers which prevents the implementation of our vision. This is especially true for operations which involve the robot making contact with its environment when the environment is not completely structured.

In the research described here we have tried to address this problem by investigating methods of transferring assembly skills to robots by observing a human-performed demonstration of the skill. Specifically, the focus here has been on learning low-level, force-based assembly skills (rather than position-based skills). The ability to quickly transfer force-based skills to robots will dramatically increase the functional capabilities of our envisioned intelligent robot tools and assistants.

The goal has been to provide a method of programming which can be done quickly and easily and which does not require the talents of a trained engineer. To accomplish such a goal, we need an interface which facilitates the demonstration process for the operator and makes the skill transfer easy. Thus, we have tried to maintain focus on the human operator. In an attempt to make the skill transfer easier for the operator, we have designed the human-to robot interaction to be similar to a human-to-human interaction and have developed a teacher-student model for robot programming. The human-robot relationship is modeled as a teacher-student relationship where the human operator plays the role of the teacher and the robot takes the role of student. From this, we have produced an interactive teaching environment for programming robots.

To overcome uncertainties in position and orientation, we have presented a model for assembly skills which uses an event-based approach. The hybrid control model of an assembly skill uses discrete, force-based qualitative states, termed single-ended contact formations (SECF), which describe how contact is being made with the environment. A change in SECF constitutes an event and triggers a new control command to the robot. A supervisory controller is used to guide the system from one SECF to the next desired SECF. Thus, the skill execution is not dependent on absolute position but rather responds to changes in the force-based qualitative state. Using the model, we have shown how skill acquisition involves the learning of three functions: (1) the mapping of force sensor signals to qualitative states, (2) the desired sequence of qualitative states, and (3) the transition trajectories which drive the system from one qualitative state to the next.

The key to the event-based approach using the hybrid control model is the identification of the force-based qualitative state. Because of the need to execute the learned skill in real-time, the computational load must be kept low. Consequently, in this research, we have explored sensor-based methods for identifying the SECF and have avoided the use of geometric models. The objective was to explore the possibilities and the limitations of using a pure, sensor-based approach, in an effort to gain insight on how to combine force information with geometric information or other sensing modalities.

Using a sensor-based approach, we have searched for distinguishing patterns in the force signals that could be used for identifying the SECF. A pattern was found by projecting the force vectors and the moment vectors onto unit spheres. We have shown how the projections of the vector sets can be viewed as clusters. Identifying the SECF for a force signal is then equivalent to determining in which cluster the projected vectors fall.

In this work, two SECF classifiers have been presented which are based on the idea of using clusters to represent the force-based patterns. One of the classifiers is based on fuzzy logic, and the other classifier uses a neural network. To train the classifiers, we have collected data for a set of known SECF's and then used supervised learning. In the case of the fuzzy logic-based classifier, the training involved the automatic generation of the appropriate membership functions. For the neural network classifier, a two-phase backpropagation technique was introduced as an alternative to the classic backpropagation method.

To test the performance of the SECF classifiers, experiments were designed for

6 different data sets. The data sets represented different shapes of workpieces and different combinations of SECF's, which were designed to test how well the classifiers performed under different situations. There were some cases where the fuzzy classifier performed better and other cases where the neural network classifier performed better. The different situations were addressed in detail in the results discussion. In all 6 data sets, we were able to achieve a successful classification rate of at least 94% with one of the classification methods. In most cases, the success rate was 98% or better.

Using the trained classifier for one of the data sets, additional experiments were designed to test the complete skill acquisition technique. A skill demonstration was performed by a human teacher, and the demonstration profiles were used to extract information for the skill model. In the experimental results, we showed how the classifier could be used to extract a sequence of SECF's from the force profile. An interactive teaching environment was also presented, so that, if the teacher did not perform the demonstration perfectly, the demonstration profiles could be modified and used for refinement of the SECF sequence. Finally, we showed how the transition trajectories can be generated from the refined sequence, thus completing the skill acquisition. To increase the ability to handle position and orientation uncertainties, a second sequence was also demonstrated and learned, and the two skill sequences and transitions were merged into one finite state machine which could be used to control the assembly operation.

In the experiments, the learned skill model was used to autonomously execute the skill. By using different starting positions and orientations of the environment fixture, we demonstrated how the skill model can accommodate changes in the environment and still successfully complete the learned skill.

A. Contributions

This research represents several contributions to the study of robotics. These contributions are briefly discussed below.

The creation of a natural human-to-robot interface for force-based skill transfer

One characteristic that distinguishes this work from other similar research is our emphasis on the human user and our persistence in creating an interface that will be easy and natural to use. The human-robot interface uses an anthropomorphic model which is an extension of a teacher-student model developed for human-human interaction. This emphasis has led us to the development of an interactive teaching environment. One of the noticeable benefits is that skill acquisition can take place even if the skill demonstration was performed in a less than perfect manner.

The characterization and learning of the force-based qualitative state

A significant effort was made to analyze the force signals resulting from the various single-ended contact formations that comprise our force-based qualitative states. The analysis showed that the force signals could be processed using pattern recognition techniques. The result is an effective, efficient method of identifying the single-ended contact formation from force sensor signals only.

The understanding and modeling of assembly skills

In this research, we investigated the use of a hybrid control model to model assembly skills. The model presented here differs from other research in that it incorporates both processing of symbols (using the force-based qualitative state) and processing of time-based functions (used for robot force control). Our research validated the skill model, illustrating how the model can be used both to learn assembly skills from human demonstration and also how to perform smooth execution of the learned skills.

Increased robustness of skill execution in semi-structured environments

Finally, we showed how the event-based approach used in the skill model could accommodate position and orientation uncertainties in the environment. In the experiments presented here, multiple test runs were performed with different starting conditions. The skill model was able to accommodate these differences so that the skill could still be executed successfully.

B. Future Work

Although the research objectives have been successfully met, we feel that this is still at the beginning stages of a research area. There is much more work that can be done. The first, most obvious step would be to try the skill acquisition method and assembly skill model using more complex tasks. We would also want to generate a bigger finite state machine, with more states, that could accommodate even greater diversity in position and orientation.

As mentioned in the earlier discussion section, the implementation of the robot force control loop should be improved so that the loop runs at a frequency of at least 600-800 Hz. This may, in fact, be necessary to smoothly perform complex assembly skills. In addition, now that the basic model has been validated, an extension of this work would be to learn force control parameters for each trajectory segment. This feature may also facilitate the learning and execution of more complex assembly skills.

We would also like to try using unsupervised learning for skill acquisition. Instead of manually demonstrating each SECF and collecting force data while in a known SECF, we would like to extract all of the skill information from a demonstration of the complete task. Using the force signals for an entire task demonstration, the training data could be generated automatically by using a clustering algorithm. Once the training data is generated, the training of the classifiers could be done as described here. This suggestion is not only a reasonable extension but also follows the basic underlying theme of trying to make the teaching process as easy as possible for the human user.

REFERENCES

[1] American Robot Corporation, *MERLIN System Operator's Guide*, American Robot Corporation, Bridgeville, PA, 1985.

[2] H. Asada, "Teaching and learning of compliance using neural nets: Representation and generalization of nonlinear compliance," In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pp. 1237–1244, Cinncinnati, OH, May 1990.

[3] H. Asada and Y. Asari, "The direct teaching of tool manipulation skills via the impedance identification of human motions," In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pp. 1269–1274, Philadelphia, PA, April 1988.

[4] H. Asada and H. Izumi, "Automatic program generation from teaching data for the hybrid control of robots," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, pp. 166–173, April 1989.

[5] H. Asada and B.-H. Yang, "Skill acquisition from human experts through pattern processing of teaching data," In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 1302–1307, Scottsdale, AZ, May 1989.

[6] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, NY, 1981.

[7] J. Bezdek and S. Pal, *Fuzzy Models for Pattern Recognition*, IEEE Press, New York, NY, 1992.

[8] R. Brockett, "Hybrid models for motion control systems," In *Essays on Control: Perspectives in the Theory and Its Applications*, H. Trentelman and J. Willems, editors, chapter 2, pp. 29–53, Birkhauser, Boston, MA, 1993.

[9] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14 – 23, March 1986.

[10] Y.-J. Cho, K. Tanie, P. Akella, and T. Kotoku, "Command sequence replanning using discrete-event-based task model in tele-robotic part mating," In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, volume 1, pp. 647–653, Nagoya, Japan, May 1995.

[11] A. Collins and A. Stevens, "Goals and strategies of inquiry teachers," In *Advances in Instructional Psychology*, R. Glaser, editor, chapter 2, pp. 65–119, Lawrence Erlbaum Associates, Hillsdale, NJ, 1982.

[12] T. Dean, K. Basye, and L. Kaelbling, "Uncertainty in graph-based map learning," In *Robot Learning*, J. Connell and S. Mahadevan, editors, chapter 7, pp. 171–192, Kluwer Academic Publishers, Boston, MA, 1993.

[13] N. Delson and H. West, "Robot programming by human demonstration," In *Japan-USA Symposium on Flexible Automation*, volume 2, pp. 1387–1397, San Francisco, July 1992, ASME.

[14] N. Delson and H. West, "Robot programming by human demonstration: Subtask compliance controller identification," In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 33–41, Yokohama, Japan, July 1993.

[15] N. Delson and H. West, "Robot programming by human demonstration: Adaptation and inconsistency in constrained motion," In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 1, pp. 30–36, Minneapolis, MN, April 1996.

[16] R. Desai and R. Volz, "Identification and verification of termination conditions in fine motion in presence of sensor errors and geometric uncertainties," In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 800–807, Scottsdale, AZ, May 1989.

[17] N. Durlach and A. Mavor, editors, *Virtual Reality: Scientific and Technological Challenges*, National Academy Press, Washington, DC, 1995.

[18] R. E. Eberts, *User Interface Design*, Prentice Hall, Englewood Cliffs, NJ, 1994.

[19] A. Farahat, B. Graves, and J. Trinkle, "Identifying contact formations in the presence of uncertainty," In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pp. 59–64, Pittsburgh, PA, August 1995.

[20] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and*

Applications, Prentice Hall, Englewood Cliffs, NJ, 1994.

[21] J. Funda, T. S. Lindsay, and R. P. Paul, "Teleprogramming: Toward delay-invariant remote manipulation," *Presence*, vol. 1, no. 1, pp. 29–44, Winter 1992.

[22] B. Graves, "A generalized teleautonomous architecture using situation-based action selection,", Ph.D. dissertation, Texas A&M University, College Station, TX, 1995.

[23] K. Hara and R. Yokogawa, "Recognition of state in peg-in-hole by fuzzy schema," *Journal of Advanced Automation Technology*, vol. 4, no. 3, pp. 134–139, 1992.

[24] V. Hayward and R. Paul, "Robot manipulator control under UNIX: RCCL, a robot control C library," *International Journal of Robotics Research*, vol. 5, no. 4, pp. 94–111, 1986.

[25] S. Hirai and H. Asada, "A model-based approach to the recognition of assembly process states using the theory of polyhedral convex cones," In *Proceedings of the 1990 Japan USA Symposium on Flexible Automation*, pp. 809–816, Kyoto, Japan, 1990.

[26] S. Hirai, H. Noguchi, and K. Iwata, "Transplantations of human skillful motion to manipulators in insertion of deformable tubes," In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, volume 2, pp. 1900–1905, Nagoya, Japan, May 1995.

[27] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-based space robotics

– ROTEX and its telerobotic features," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 649–663, October 1993.

[28] N. Hogan, "Impedance control: An approach to manipulation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, pp. 1–24, March 1985.

[29] G. Hovland and B. McCarragher, "Frequency-domain force measurements for discrete event contact recognition," In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 2, pp. 1166–1171, Minneapolis, MN, April 1996.

[30] G. Hovland, P. Sikka, and B. McCarragher, "Skill acquisition from human demonstration using a hidden markov model," In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 3, pp. 2706–2711, Minneapolis, MN, April 1996.

[31] JR3, Inc., *JR3 DSP-Based Force Sensor Receivers: Software and Installation Manual*, JR3, Inc., Woodland, CA, 1993.

[32] M. Kaiser and R. Dillmann, "Building elementary robot skills from human demonstration," In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 3, pp. 2700–2705, Minneapolis, MN, April 1996.

[33] S. B. Kang and K. Ikeuchi, "Determination of motion breakpoints in a task sequence from human hand motion," In *Proceedings of the 1994 IEEE Inter-*

national Conference on Robotics and Automation, volume 1, pp. 551–556, San Diego, CA, May 1994.

[34] K. Kitagaki, T. Ogasawara, and T. Suehiro, "Methods to detect contact state by force sensing in an edge mating task," In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, volume 2, pp. 701–706, Atlanta, GA, May 1993.

[35] R. Koeppe, A. Breidenbach, and G. Hirzinger, "Skill representation and acquisition of compliant motions using a teach device," In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pp. 897–904, Osaka, Japan, November 1996.

[36] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[37] K. Kosuge, T. Fukuda, and H. Asada, "Acquisition of human skills for robotic systems," In *Proceedings of the 1991 IEEE International Symposium on Intelligent Control*, pp. 469–474, Arlington, VA, August 1991.

[38] S. Liu and H. Asada, "Transferring manipulative skills to robots: Representation and acquisition of tool manipulative skills using a process dynamics model," In *Modeling and Control of Compliant and Rigid Motion Systems*, volume 31, pp. 121–129, Atlanta, GA, December 1991, ASME.

[39] J. Lloyd, *RCCL/RCI Reference Manuals, Release 4.2*, McGill Research Centre

for Intelligent Machines, McGill University, Montreal, Quebec, Canada, 1992.

[40] Lynx Real-Time Systems, Inc., *LynxOS 2.4 Reference Manual*, Lynx Real-Time Systems, Inc., San Jose, CA, 1996.

[41] M. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 6, pp. 418–432, June 1981.

[42] B. J. McCarragher, "Force sensing from human demonstration: Stiffness, impedance and kinesthetic sensibility," In *Proceedings of the 1994 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pp. 1226–1233, Munich, Germany, September 1994.

[43] B. J. McCarragher, "Force sensing from human demonstration using a hybrid dynamical model and qualitative reasoning," In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 1, pp. 557–563, San Diego, CA, May 1994.

[44] B. J. McCarragher and H. Asada, "Qualitative template matching using dynamic process models for state transition recognition of robotic assembly," In *Modeling and Control of Compliant and Rigid Motion Systems*, volume 31, pp. 155–163, Atlanta, GA, December 1991, ASME.

[45] B. J. McCarragher and H. Asada, "A discrete event controller using petri nets applied to assembly," In *Proceedings of the 1992 IEEE/RSJ International Con-*

ference on Intelligent Robots and Systems, volume 3, pp. 2087–2094, Raleigh, NC, July 1992.

[46] D. McCloskey, "Kinesthetic sensibility," *Physiological Reviews*, vol. 58, no. 4, , October 1978.

[47] J. Melsa and D. Schultz, *Linear Control Systems*, McGraw-Hill Book Company, New York, NY, 1969.

[48] N. Mimura and Y. Funahashi, "Parameter identification of contact conditions by active force sensing," In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 3, pp. 2645–2650, San Diego, CA, 1994.

[49] A. Nakamura, T. Ogasawara, T. Suehiro, and H. Tsukune, "Skill-based backprojection for fine motion planning," In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pp. 526–533, Osaka, Japan, November 1996.

[50] Y. Nakamura, M. Mizukawa, and K. Kanayama, "Multimedia communication pendant for sensor-based robotic task teaching by sharing information: Concepts and evaluation," In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pp. 1417–1424, Osaka, Japan, November 1996.

[51] J. Nielsen, *Usability Engineering*, AP Professional, Cambridge, MA, 1993.

[52] H. Ogata and T. Takahashi, "Robotic assembly operation teaching in a virtual

environment," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 391–399, June 1994.

[53] K. Ogata, *System Dynamics*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1978.

[54] H. Onda, H. Hirukawa, and K. Takase, "Assembly motion teaching system using position/force simulator extracting a sequence of contact state transition," In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pp. 9–16, Pittsburgh, PA, August 1995.

[55] G. Paul and K. Ikeuchi, "Modelling planar assembly paths from observation," In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pp. 520–525, Osaka, Japan, November 1996.

[56] M. Raibert and J. Craig, "Hybrid position/force control of manipulators," *Transactions of ASME, Journal of Dynamic Systems, Measurement and Control*, vol. 102, pp. 126–133, June 1981.

[57] J. Salisbury, "Active stiffness control of a manipulator in cartesian coordinates," In *Proceedings of the 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, volume 1, pp. 95–100, Albuquerque, NM, December 1980.

[58] T. Sato and S. Hirai, "Language-aided robotic teleoperation system (LARTS) for advanced teleoperation," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 5, pp. 476–481, October 1987.

[59] SensAble Devices, Inc., *PHANToM Haptic Interface: Installation and Technical Manual*, SensAble Devices, Inc., Cambridge, MA, 1994.

[60] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*, MIT Press, Cambridge, MA, 1992.

[61] P. Sikka and B. McCarragher, "Monitoring contact using clustering and discriminant functions," In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 2, pp. 1351–1356, Minneapolis, MN, April 1996.

[62] M. Skubic, "Force sensor characterization," Technical Report 96-010, Dept. of Computer Science, Texas A&M University, College Station, TX, April 1996.

[63] M. Skubic, S. Castrianni, and R. Volz, "Identifying contact formations from force signals: A comparison of fuzzy and neural network classifiers," In *Proceedings of the 1997 International Conference on Neural Networks*, volume 3, pp. 1623–1628, Houston, TX, June 1997.

[64] M. Skubic, G. V. Kondraske, J. D. Wise, G. J. Khoury, R. A. Volz, and S. Askew, "A telerobotics construction set with integrated performance analysis," In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pp. 20–26, Pittsburgh, PA, August 1995.

[65] M. Skubic and R. Volz, "Identifying contact formations from sensory patterns and its applicability to robot programming by demonstration," In *Proceedings of*

the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 2, pp. 458–464, Osaka, Japan, November 1996.

[66] M. Skubic and R. Volz, "Learning force sensory patterns and skills from human demonstration," In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, volume 1, pp. 284–290, Albuquerque, NM, April 1997.

[67] S. Stansfield, "Reasoning about grasping," In *AAAI*, pp. 768–773, St. Paul, MN, August 1988.

[68] J. Stiver and P. Antsaklis, "Modeling and analysis of hybrid control systems," In *Proceedings of the 31st Conference on Decision and Control*, volume 4, pp. 3748–3751, Tucson, AZ, December 1992.

[69] P. D. Summers and D. D. Grossman, "XPROBE: An experimental system for programming robots by example," *The International Journal of Robotics Research*, vol. 3, no. 1, pp. 25–39, Spring 1984.

[70] S. Tso and K. Liu, "Automatic generation of robot program codes from perception of human demonstration," In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pp. 23–28, Pittsburgh, PA, August 1995.

[71] C. Tung and A. Kak, "Automatic learning of assembly tasks using a dataglove system," In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pp. 1–8, Pittsburgh, PA, August 1995.

[72] Q. Wang, J. D. Schutter, W. Witvrouw, and S. Graves, "Derivation of compliant motion programs based on human demonstration," In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 3, pp. 2616–2621, Minneapolis, MN, April 1996.

[73] *Webster's Seventh New Collegiate Dictionary*, G&C Merriam Company, Springfield, MA, 1971.

[74] P. Whalen, "Teaching accomodation task skills: from human demonstration to robot control via artificial neural networks,", Ph.D. dissertation, Air Force Institute of Technology, Wright-Patterson Air Force Base, 1995.

[75] D. E. Whitney, "Historical perspective and state of the art in robot force control," *The International Journal of Robotics Research*, vol. 6, no. 1, pp. 3–14, Spring 1987.

[76] Y.Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 799–822, December 1994.

[77] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications, Second Edition*, Kluwer Academic Publishers, Boston, MA, 1991.

APPENDIX A

COMPLETE TEST RESULTS

Skill Demonstrations

This section contains profile graphs of the skill demonstrations, as summarized in Table IX.

TABLE IX

Skill Demonstration Summary

Sequence	Trial	Status
Sequence A	Trial 1	Successful
Sequence A	Trial 2	Unsuccessful
Sequence A	Trial 3	Successful
Sequence B	Trial 1	Unsuccessful
Sequence B	Trial 2	Successful
Sequence B	Trial 3	Successful

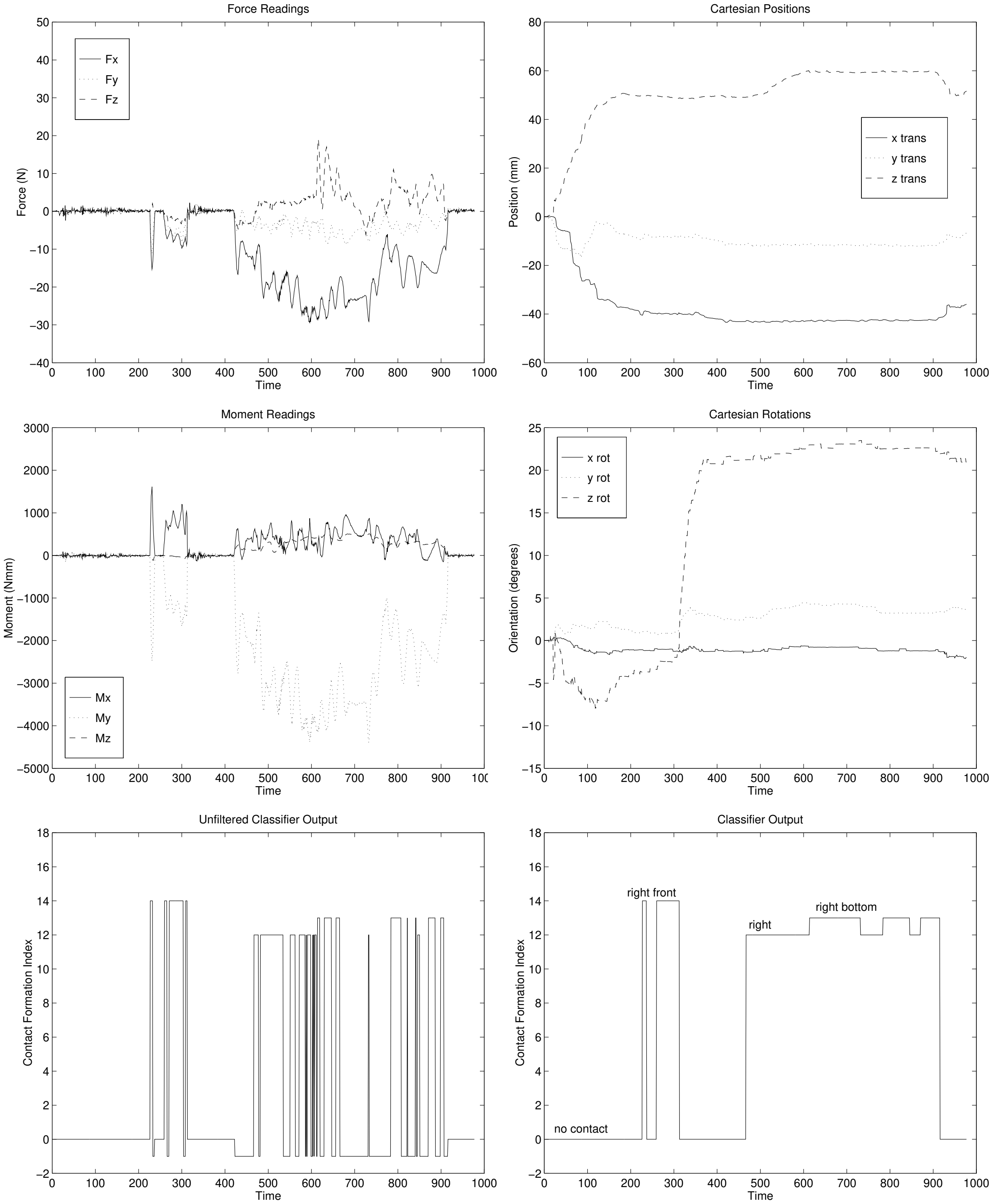


Figure 72. Sequence A Demonstration - Trial 1

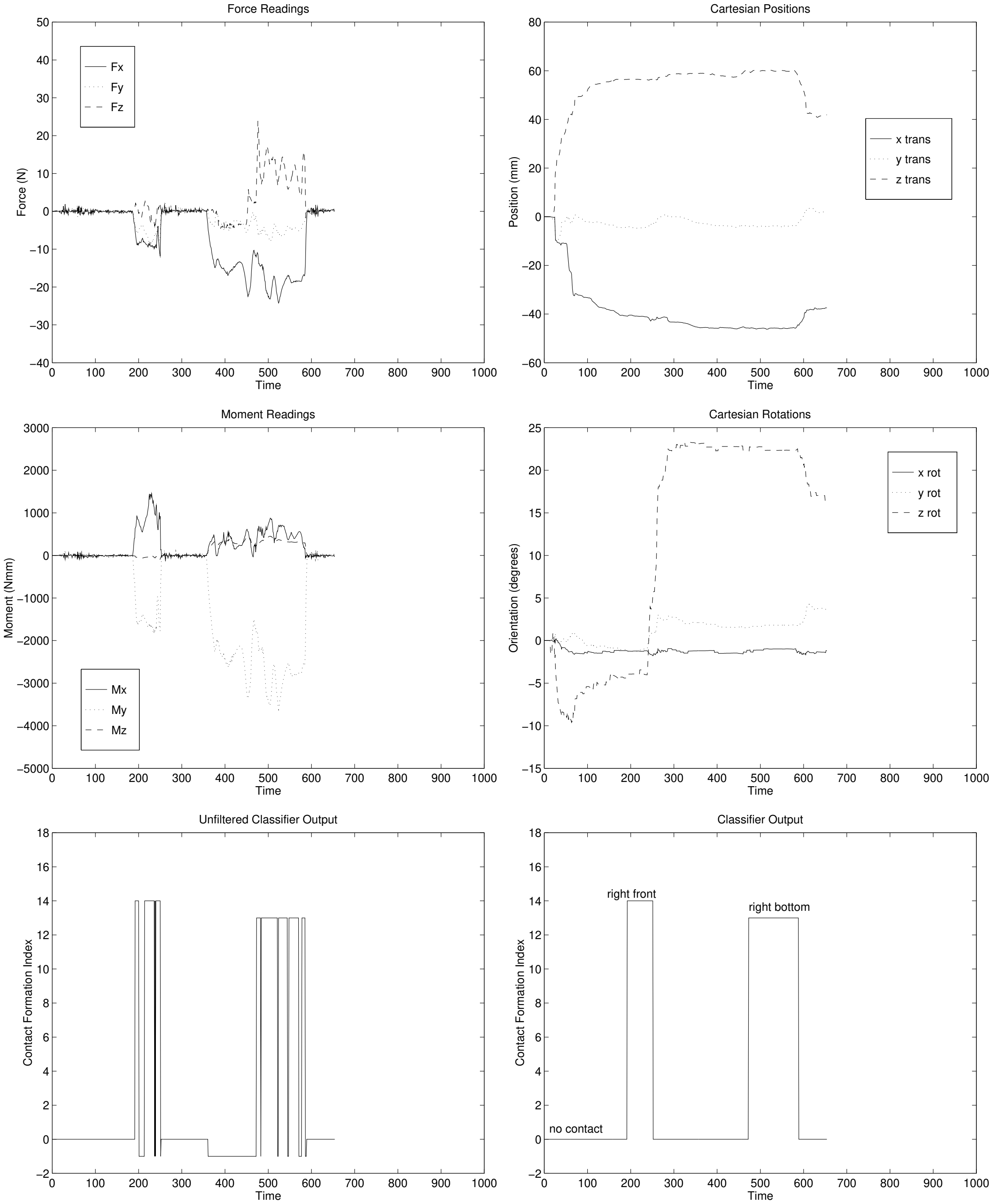


Figure 73. Sequence A Demonstration - Trial 2

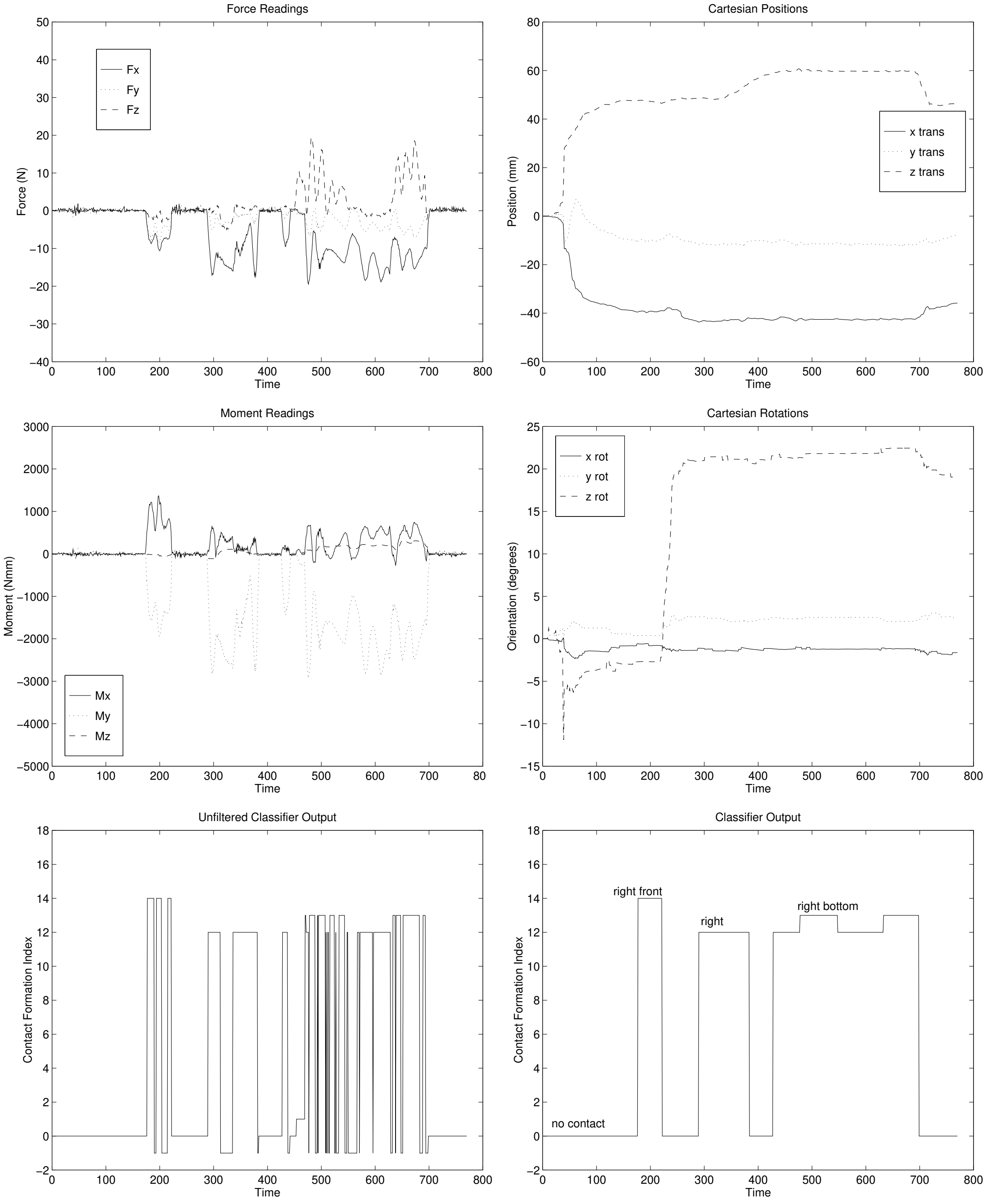


Figure 74. Sequence A Demonstration - Trial 3

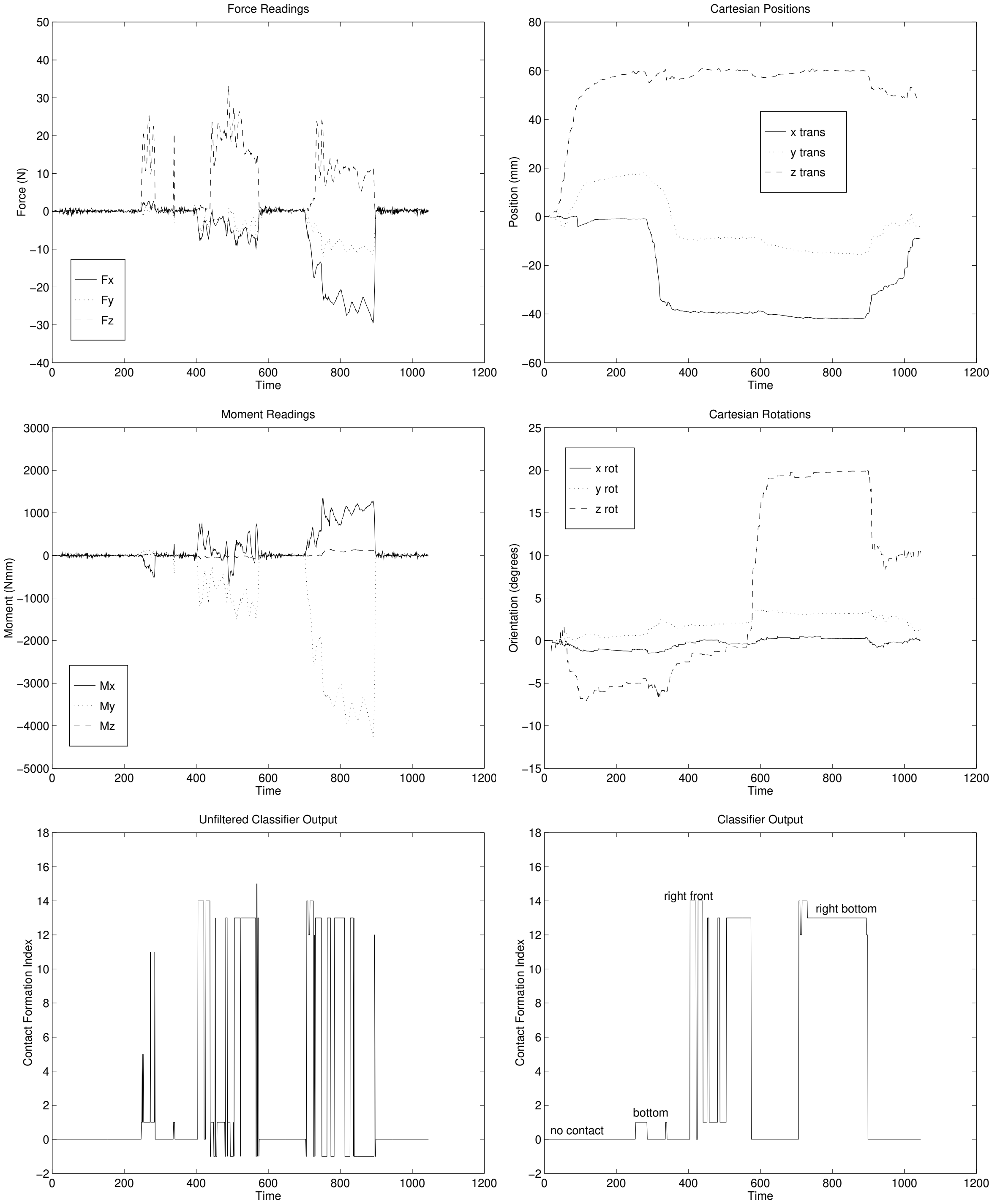


Figure 75. Sequence B Demonstration - Trial 1

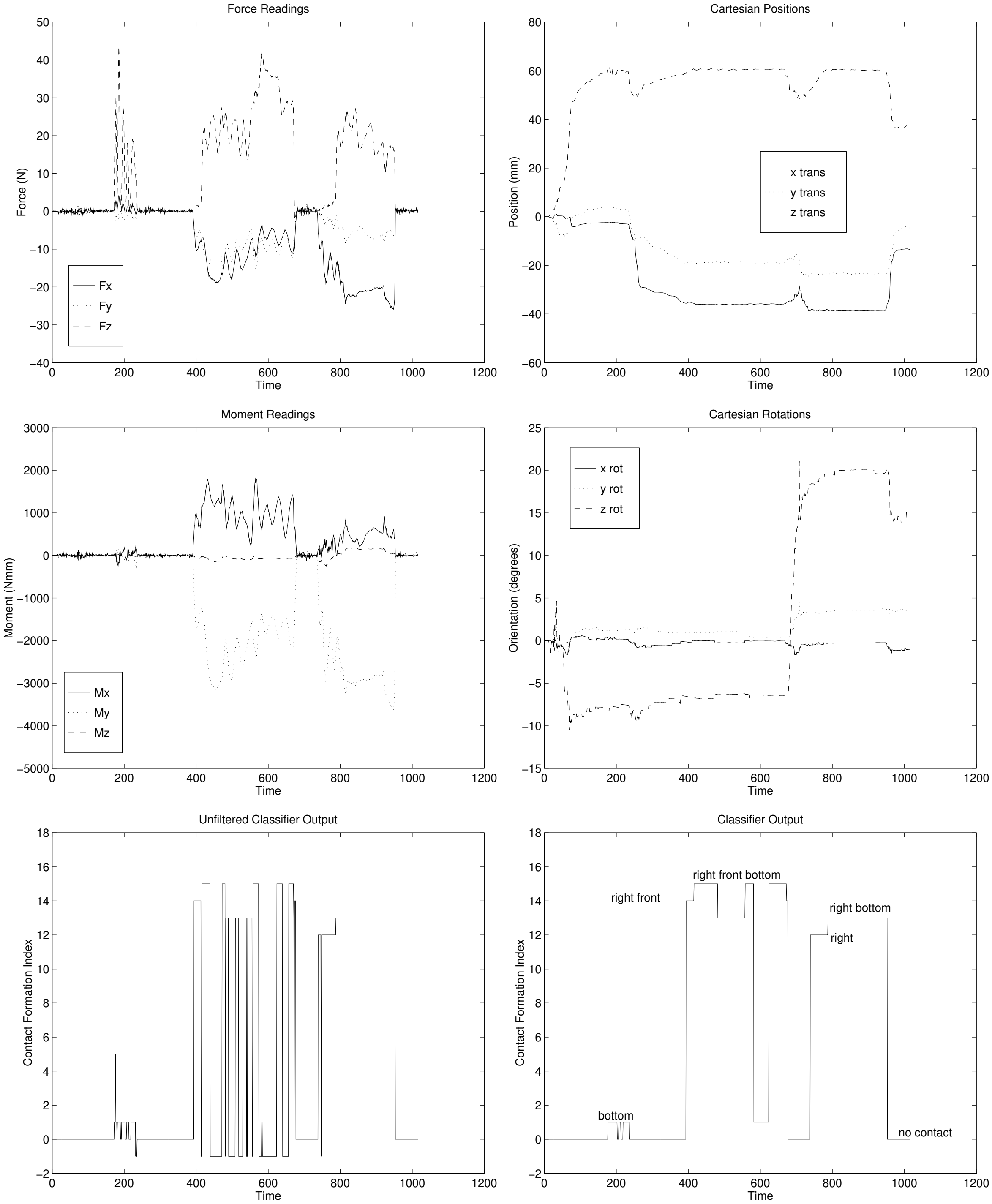


Figure 76. Sequence B Demonstration - Trial 2

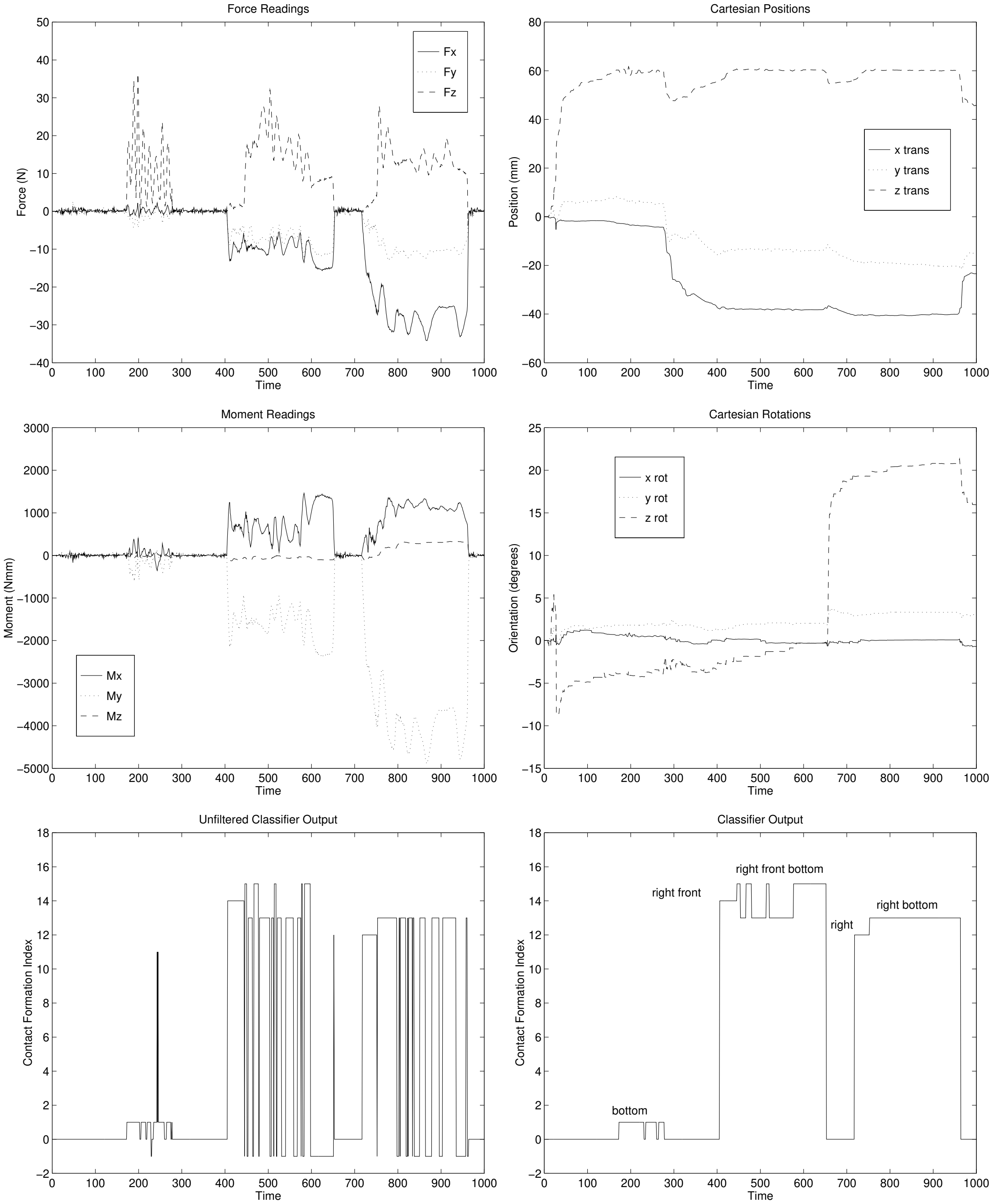


Figure 77. Sequence B Demonstration - Trial 3

Test Suite I

This section includes attempts to execute *Sequence A*. Test runs I.a through I.f were performed without the translational scale factor (tsf). Test runs I.g through I.k were performed with the translational scale factor (tsf). Force control parameters were varied, as summarized in Table X. The force control was implemented as a simple proportional gain that was brought into effect whenever the force and/or moment threshold was exceeded. The force control parameters include the translational force gain (gain), the rotational moment gain (rot. gain), the force threshold (F thres.), and the moment threshold (M thres.).

TABLE X

Test Suite I Summary

Test Run	Description	gain (mm/N)	rot. gain (deg/Nm)	F thres. (N)	M thres. (Nmm)	Status
I.a	without tsf	0.0040	0.00126	7.1	2260	Unstable
I.b	without tsf	0.0040	0.00111	7.1	2260	Unstable
I.c	without tsf	0.0048	0.00111	7.6	2825	Unstable
I.d	without tsf	0.0032	0.00095	7.6	2825	Unstable
I.f	without tsf	0.0044	0.00111	7.6	2825	Unstable
I.g	with tsf	0.0040	0.00111	7.1	2825	Successful
I.h	with tsf	0.0040	0.00111	7.1	2825	Successful
I.i	with tsf	0.0040	0.00111	7.1	2825	Successful
I.j	with tsf	0.0040	0.00111	7.1	2825	Successful
I.k	with tsf	0.0040	0.00111	7.1	2825	Successful

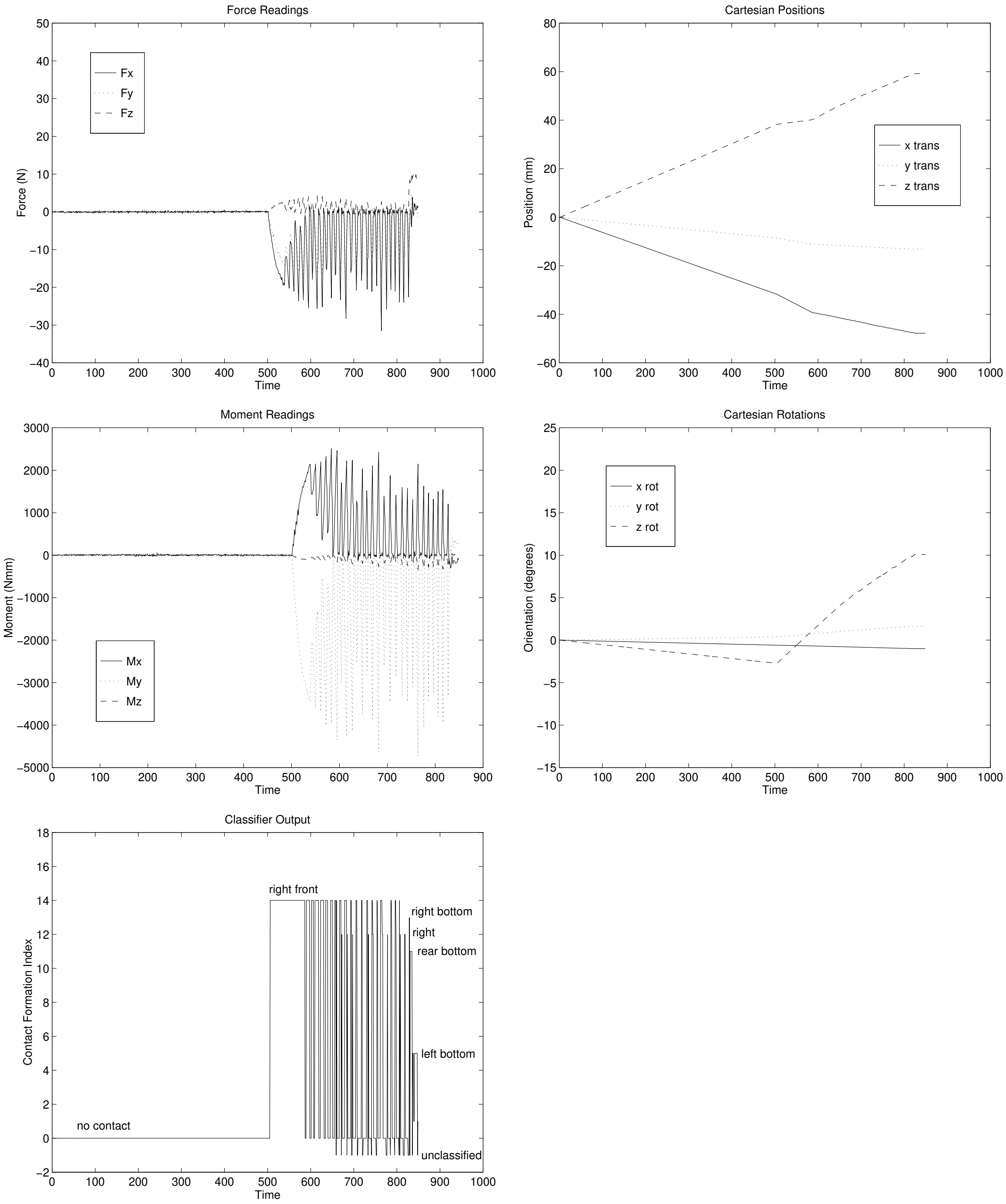


Figure 78. Test Run I.a

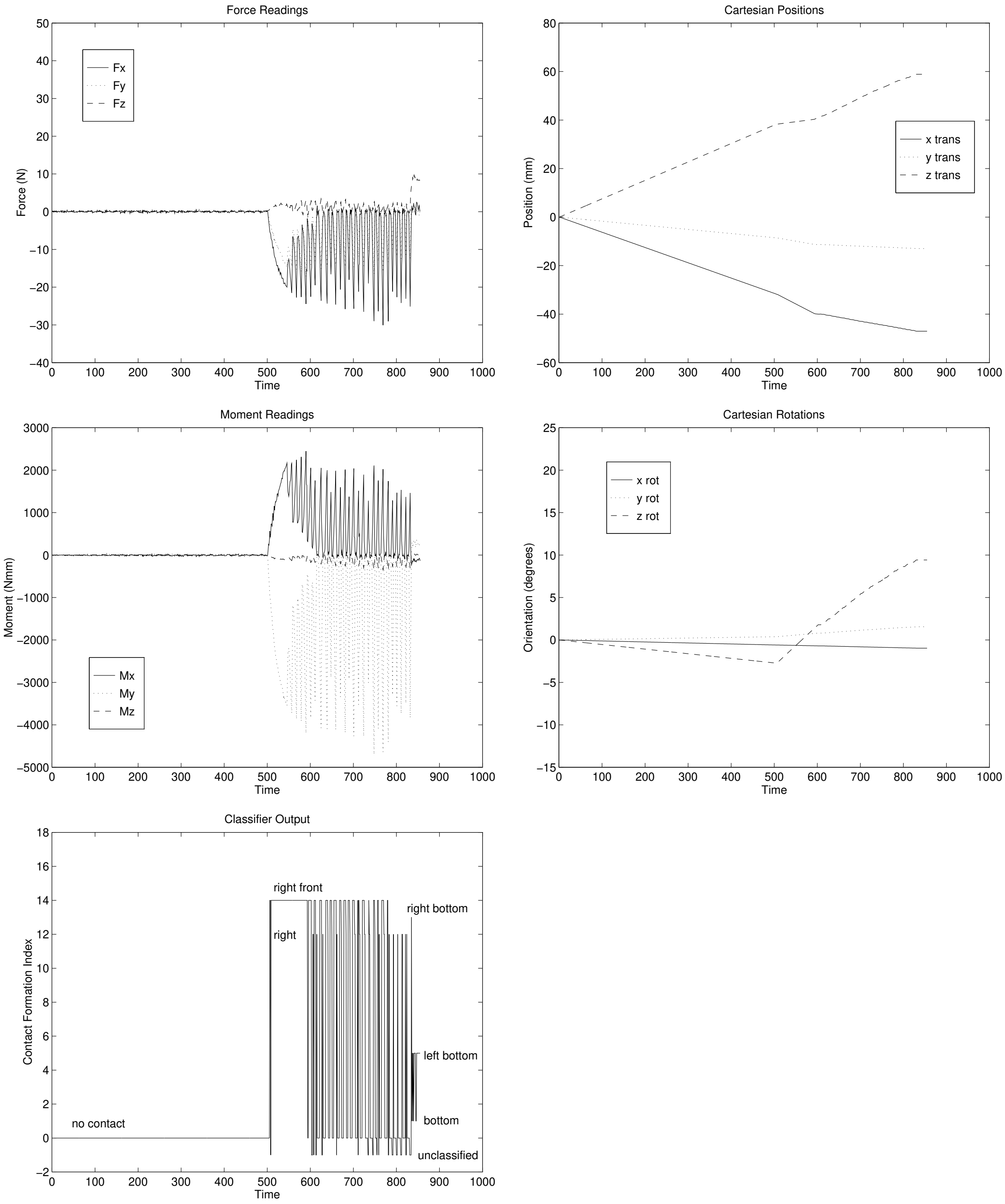


Figure 79. Test Run I.b

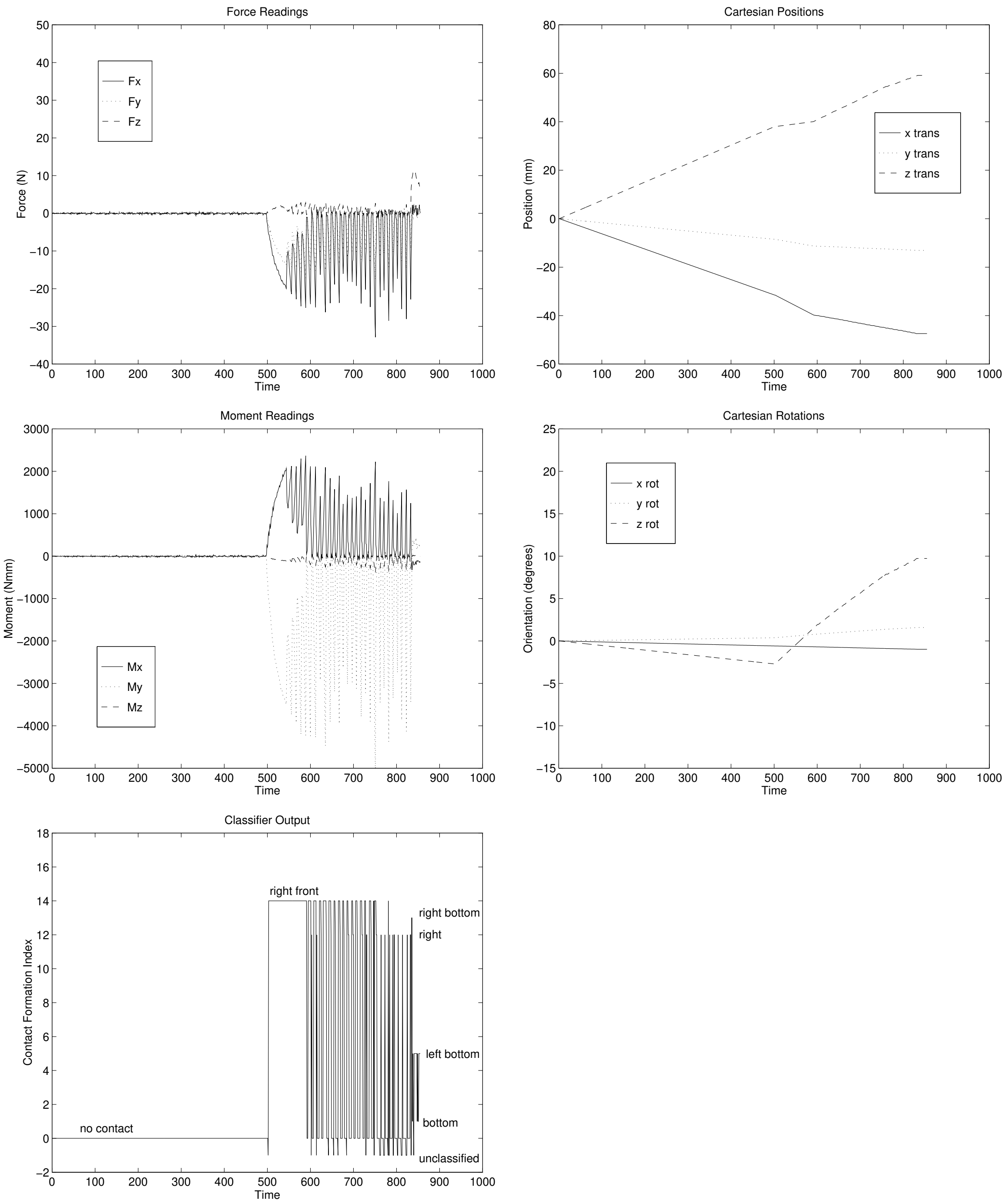


Figure 80. Test Run I.c

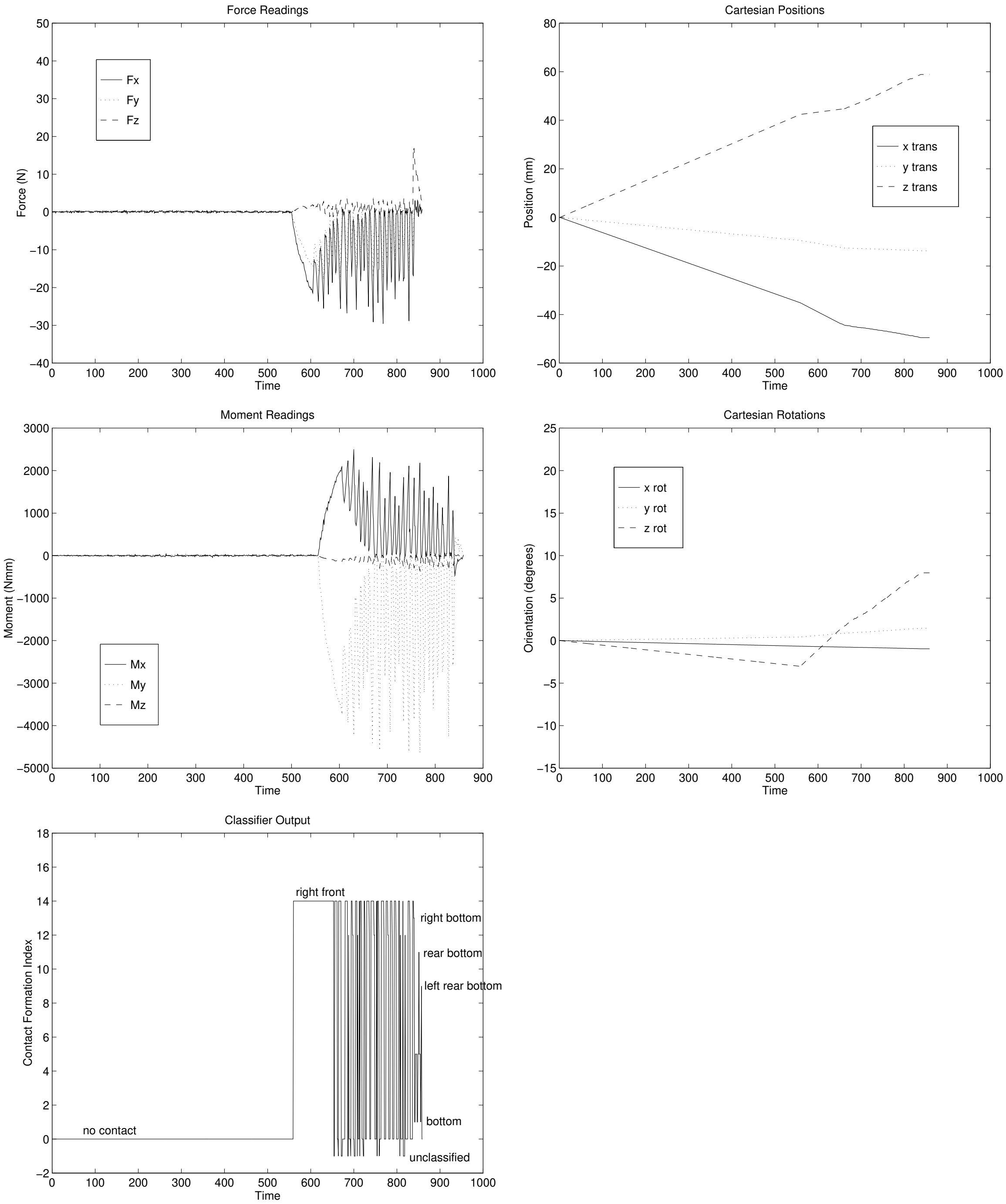


Figure 81. Test Run I.d

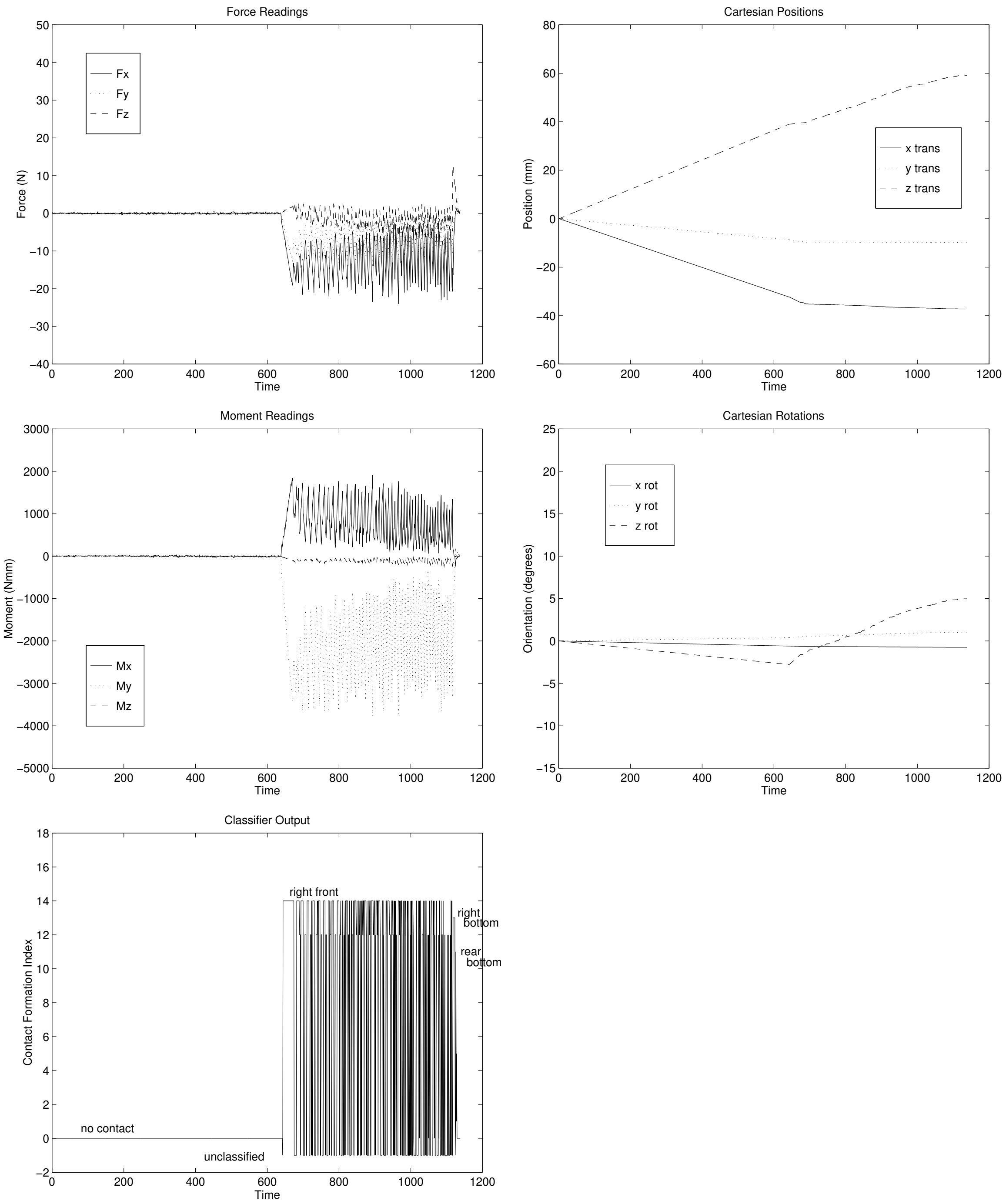


Figure 82. Test Run I.f

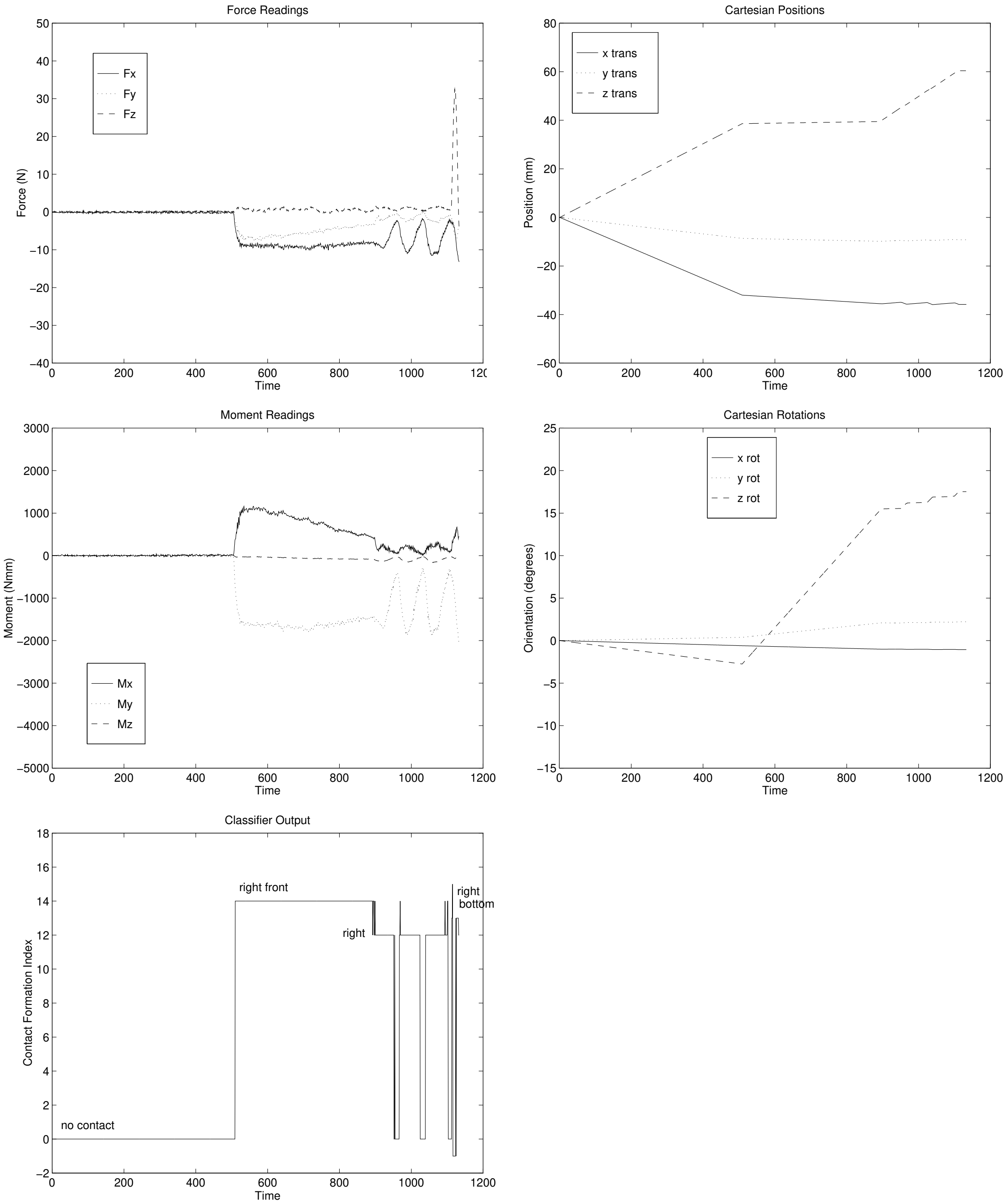


Figure 83. Test Run I.g

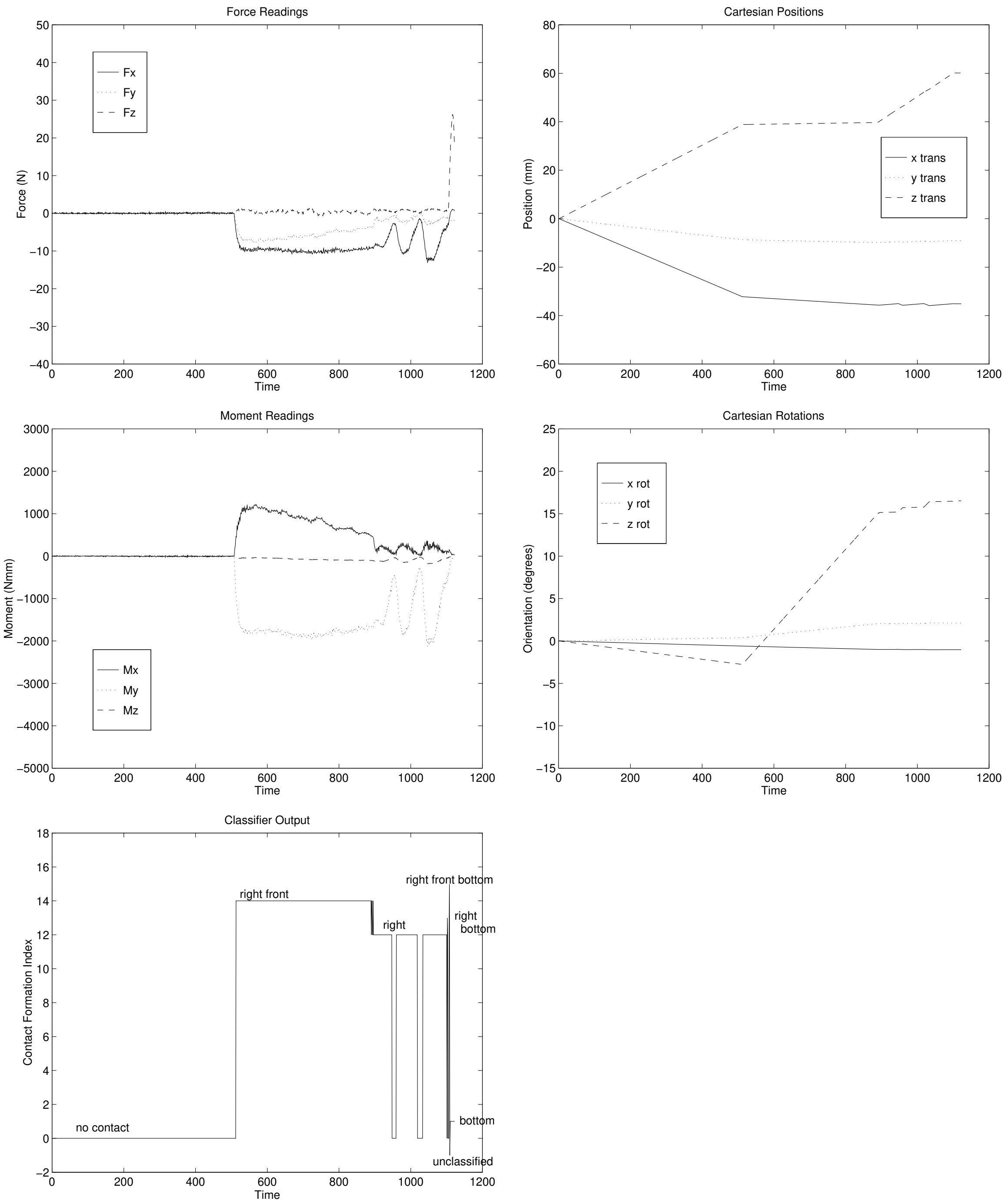


Figure 84. Test Run I.h

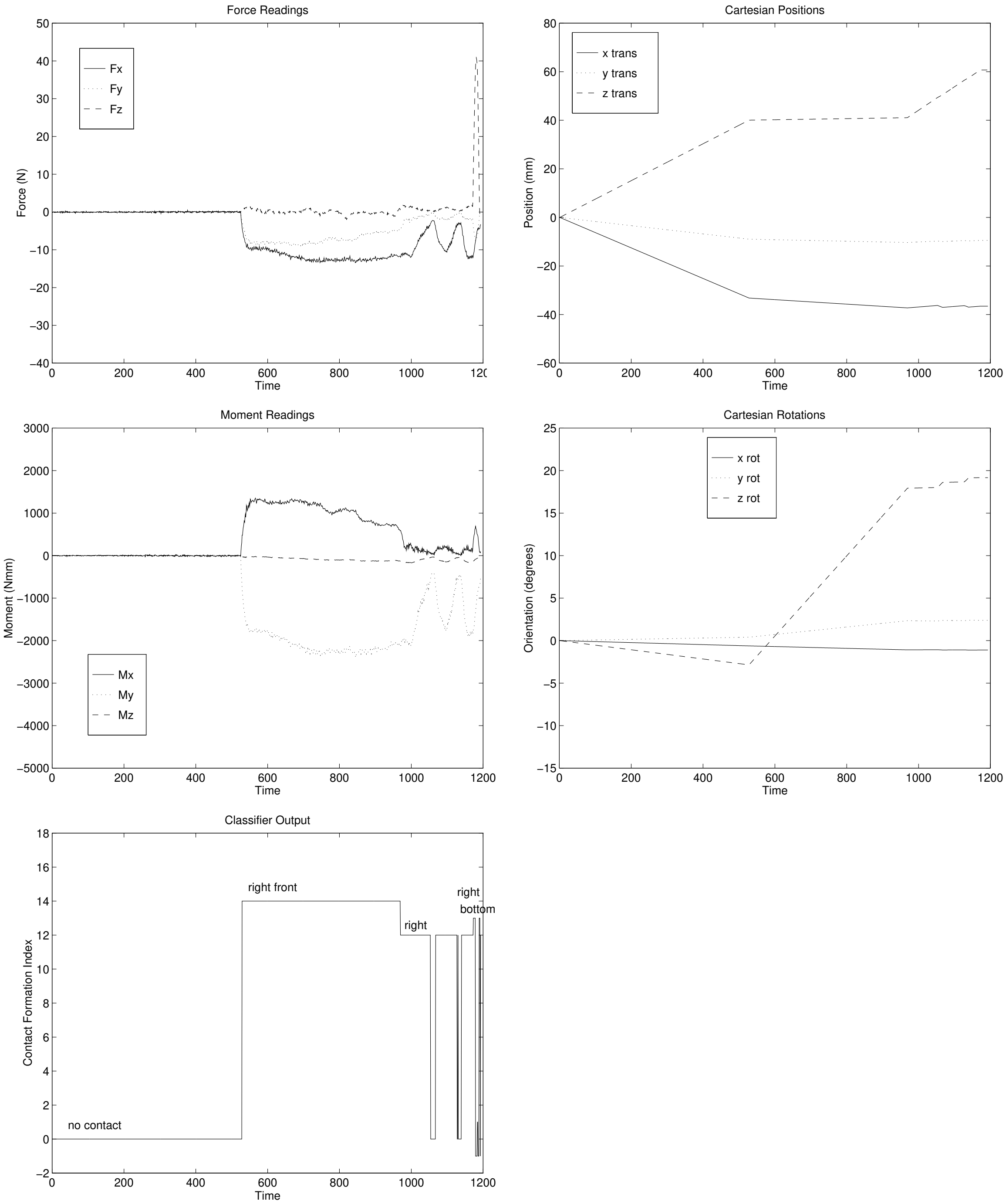


Figure 85. Test Run I.i

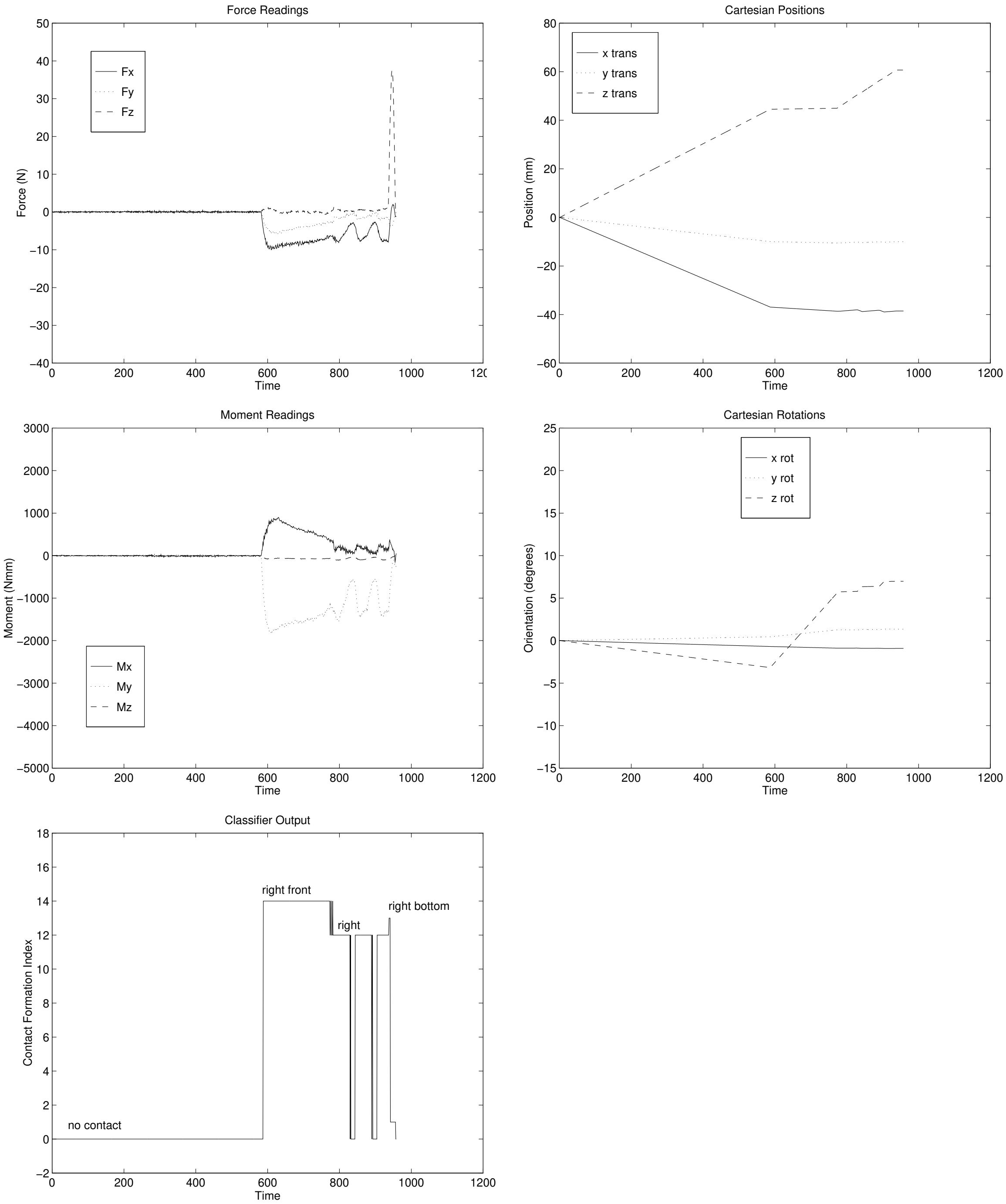


Figure 86. Test Run I.j

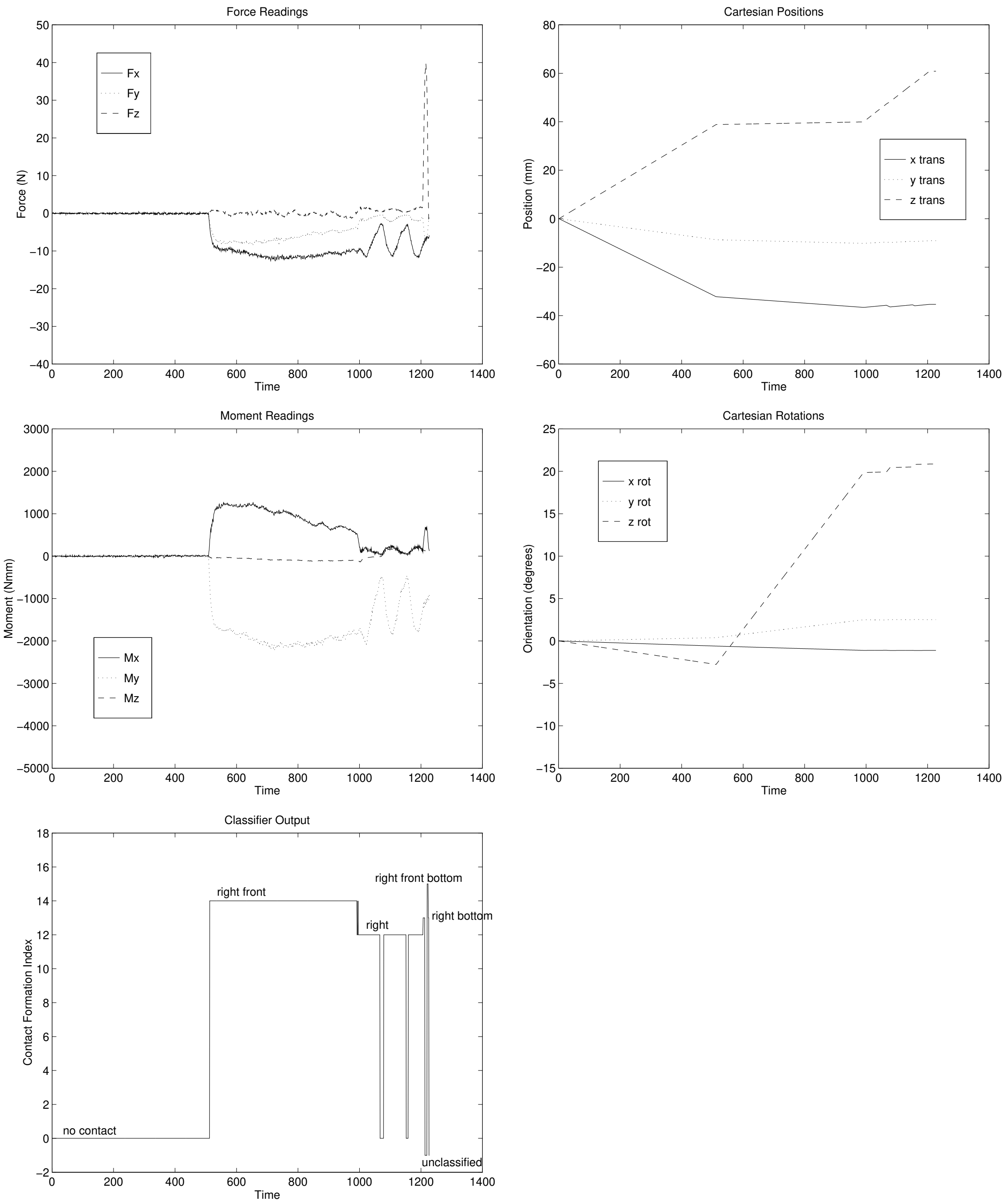


Figure 87. Test Run I.k

Test Suite II

This section includes attempts to execute *Sequence B*, as well as attempts to produce additional, undemonstrated sequences. Test runs II.a through II.d did not result in successful completion, as the program stopped before the last rotation was completed. Test runs II.e through II.i were performed after the *right front bottom* class was retrained. The starting position and orientation of the environment fixture was varied between test runs. Test runs II.e through II.i all resulted in successful skill completion. For the tests included in this group, the force control parameters were held constant.

TABLE XI

Test Suite II Summary

Test Run	Retrained	gain) (mm/N)	rot. gain (deg/Nm)	F thres. (N)	M thres. (Nmm)	Status
I.a	no	0.0040	0.00111	7.1	2825	Unsuccessful
I.b	no	0.0040	0.00111	7.1	2825	Unsuccessful
I.c	no	0.0040	0.00111	7.1	2825	Unsuccessful
I.d	no	0.0040	0.00111	7.1	2825	Unsuccessful
I.e	yes	0.0040	0.00111	7.1	2825	Successful
I.f	yes	0.0040	0.00111	7.1	2825	Successful
I.g	yes	0.0040	0.00111	7.1	2825	Successful
I.h	yes	0.0040	0.00111	7.1	2825	Successful
I.i	yes	0.0040	0.00111	7.1	2825	Successful

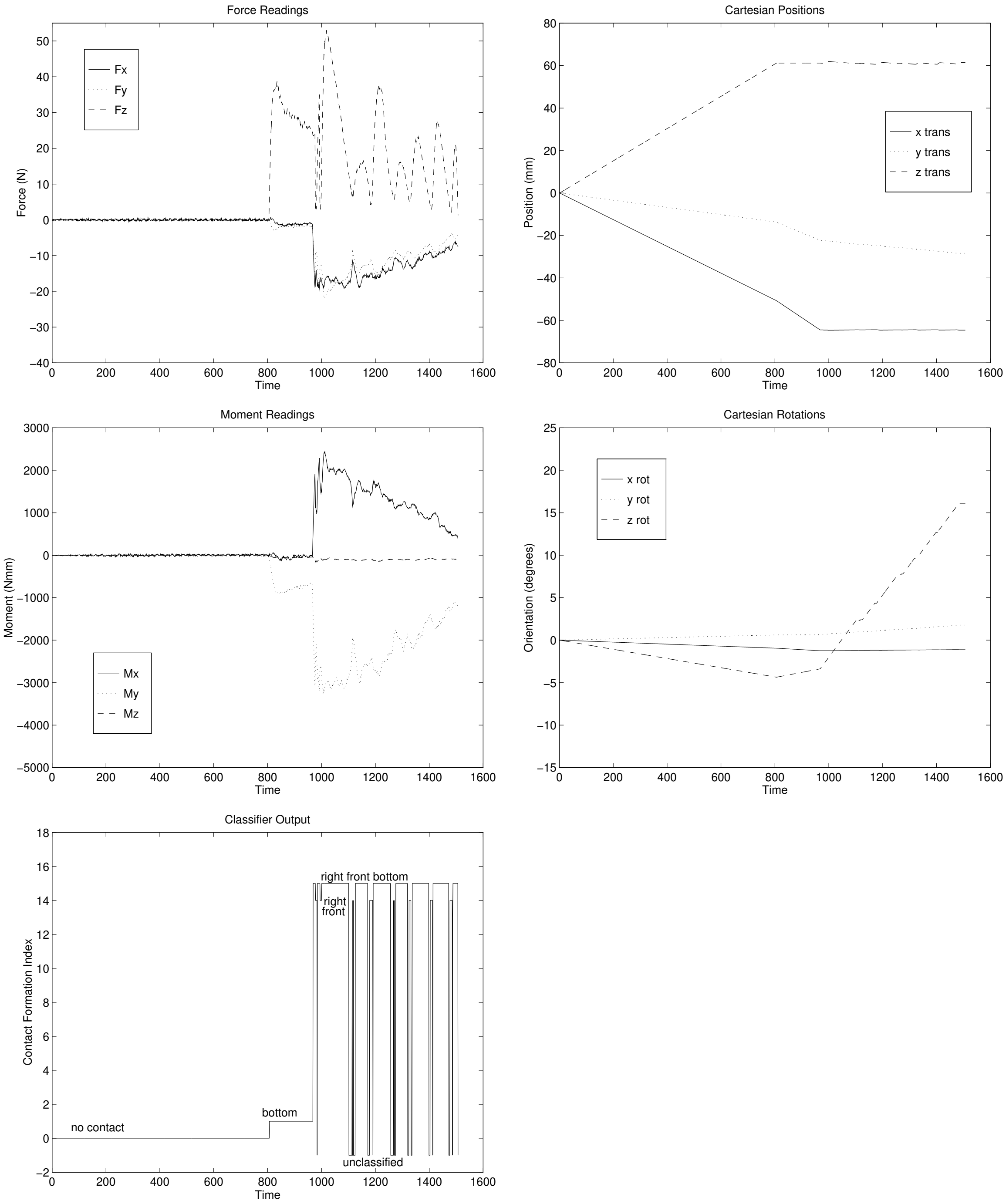


Figure 88. Test Run II.a

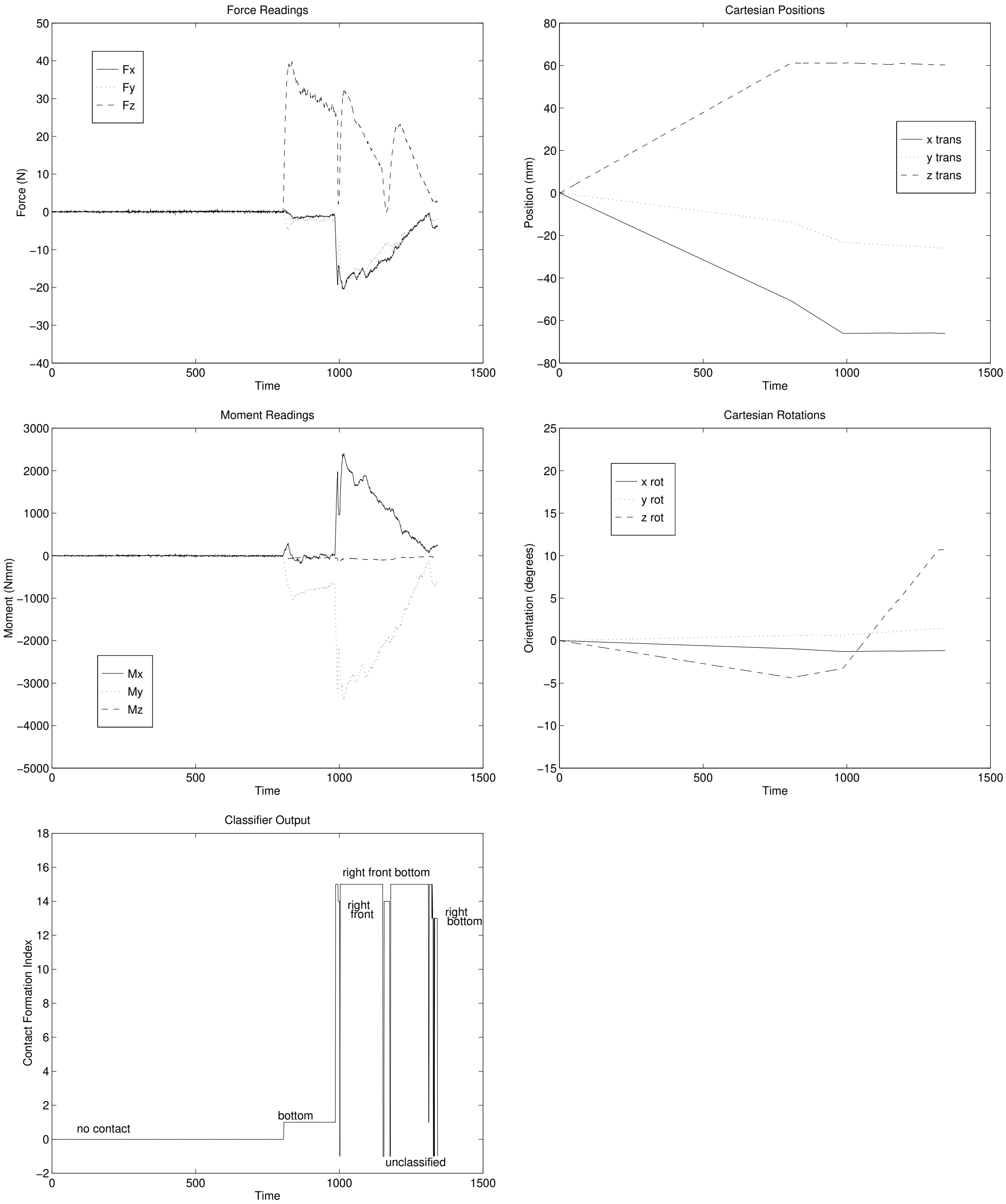


Figure 89. Test Run II.b

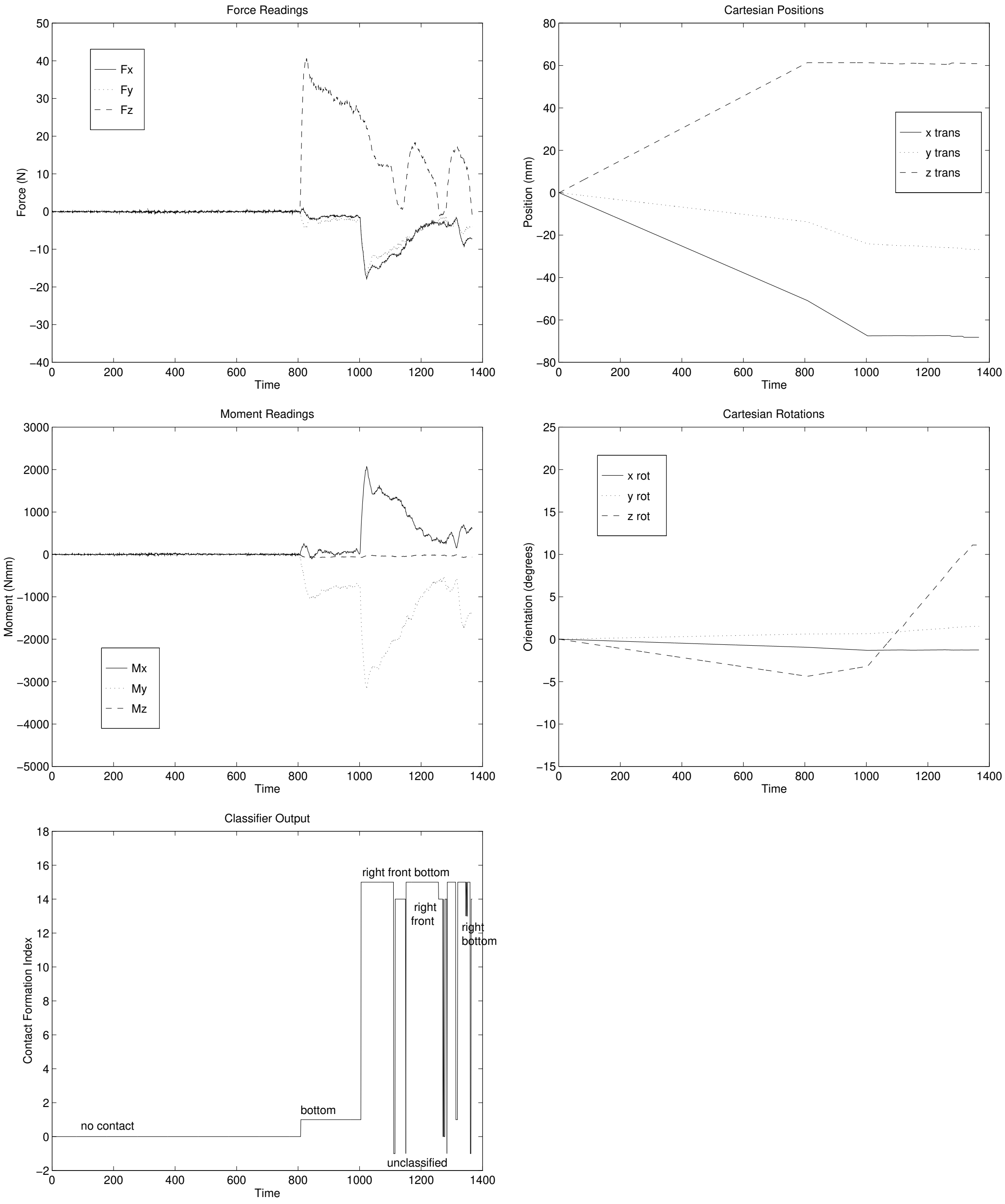


Figure 90. Test Run II.c

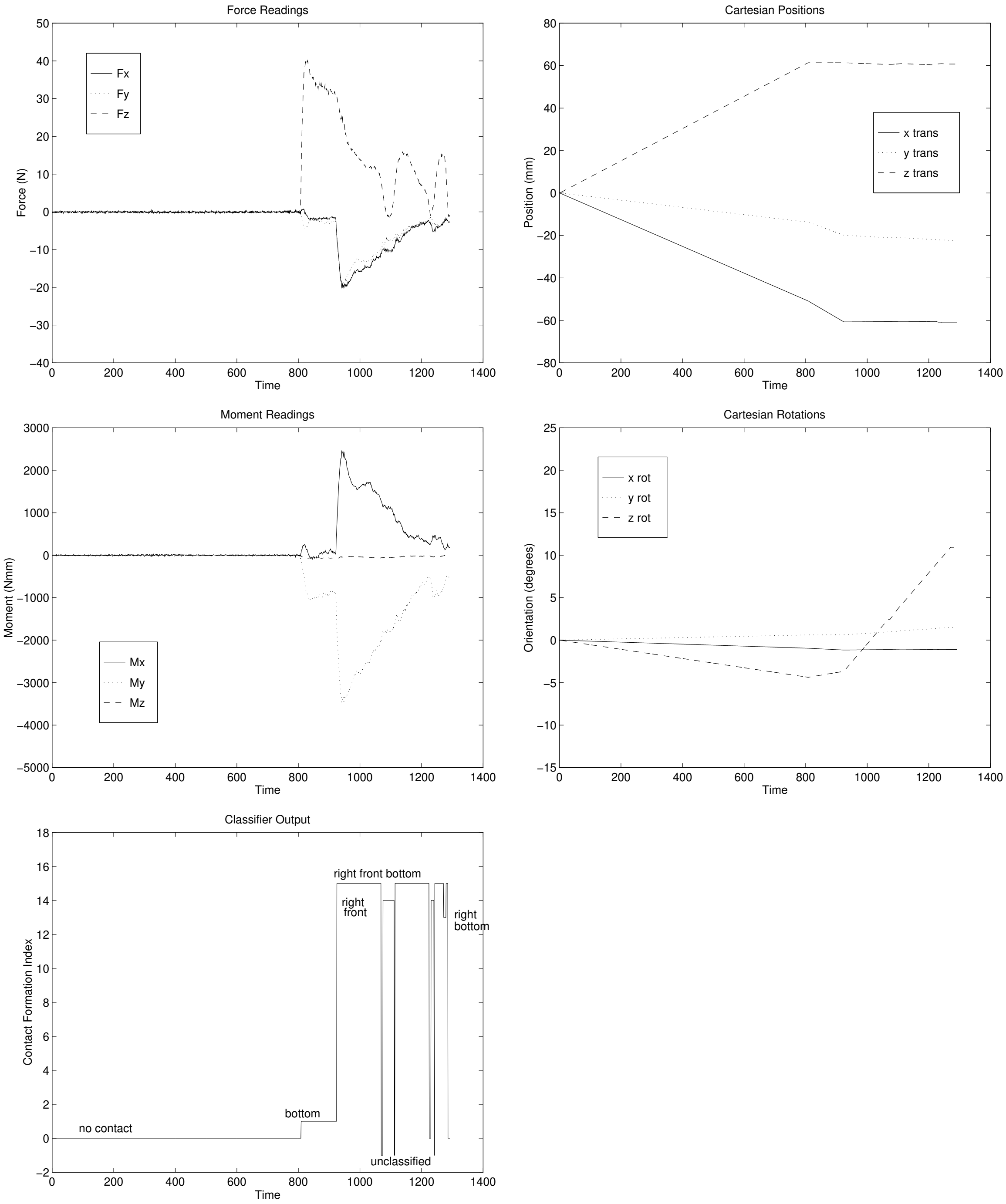


Figure 91. Test Run II.d

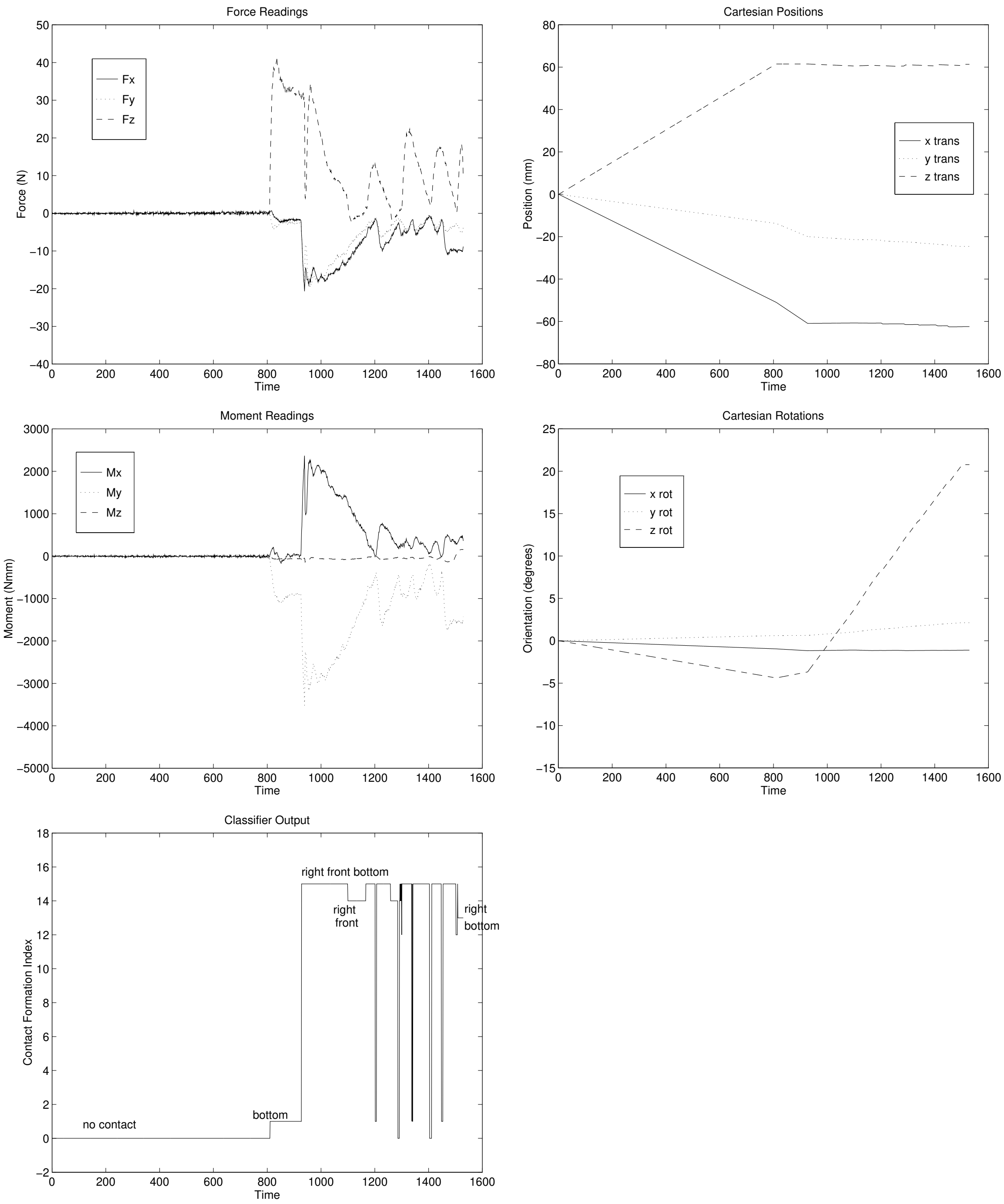


Figure 92. Test Run II.e

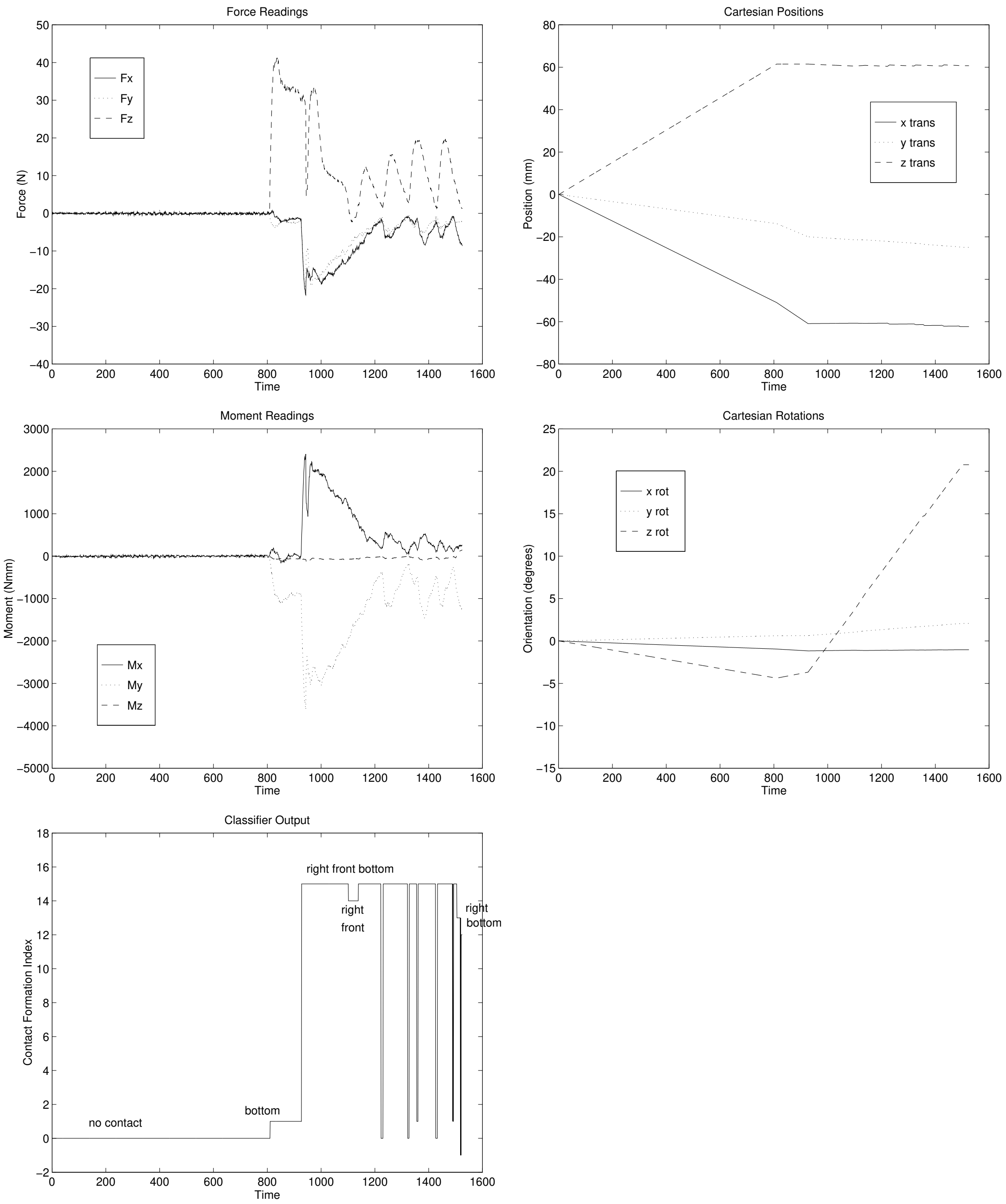


Figure 93. Test Run II.f

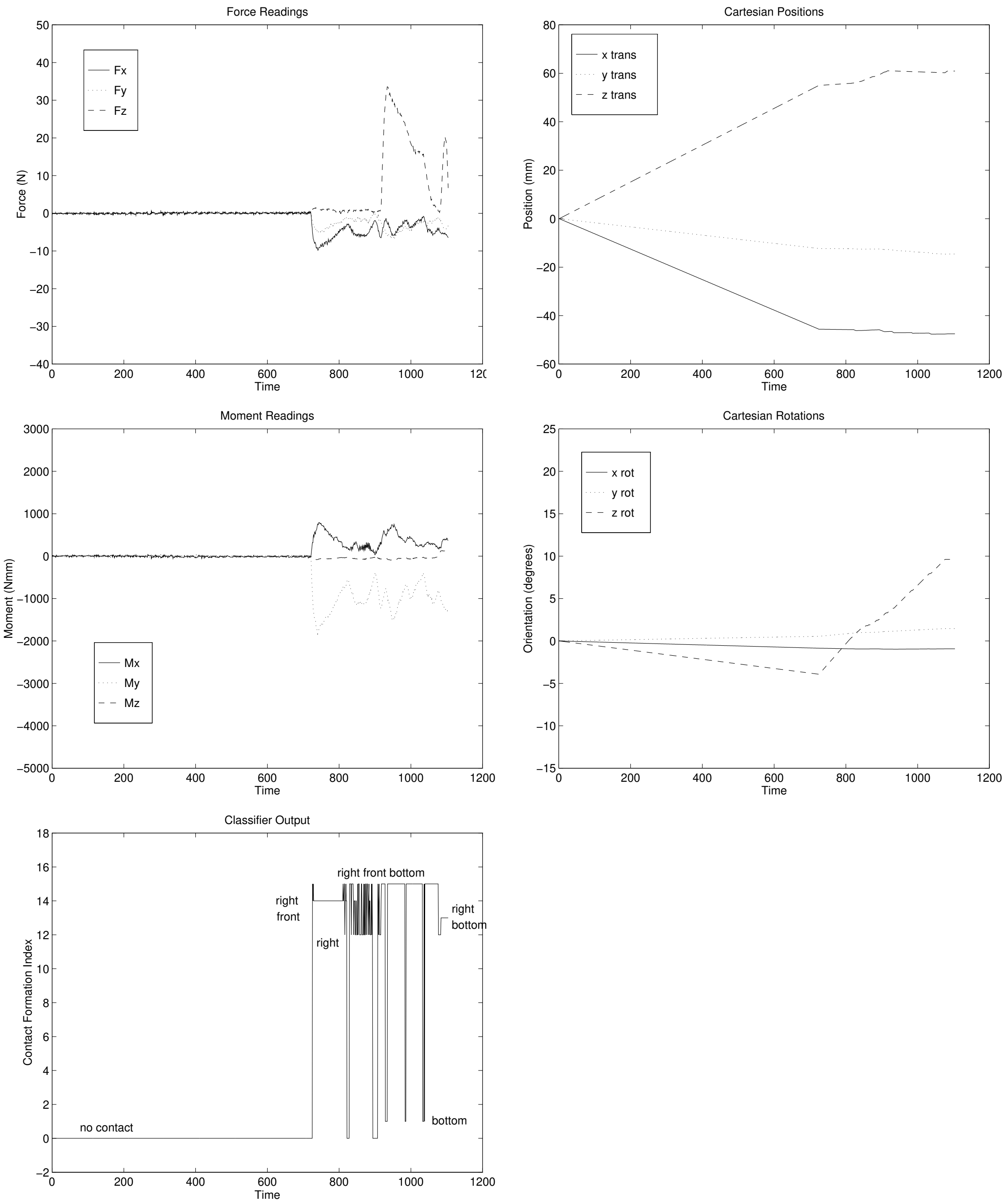


Figure 94. Test Run II.g

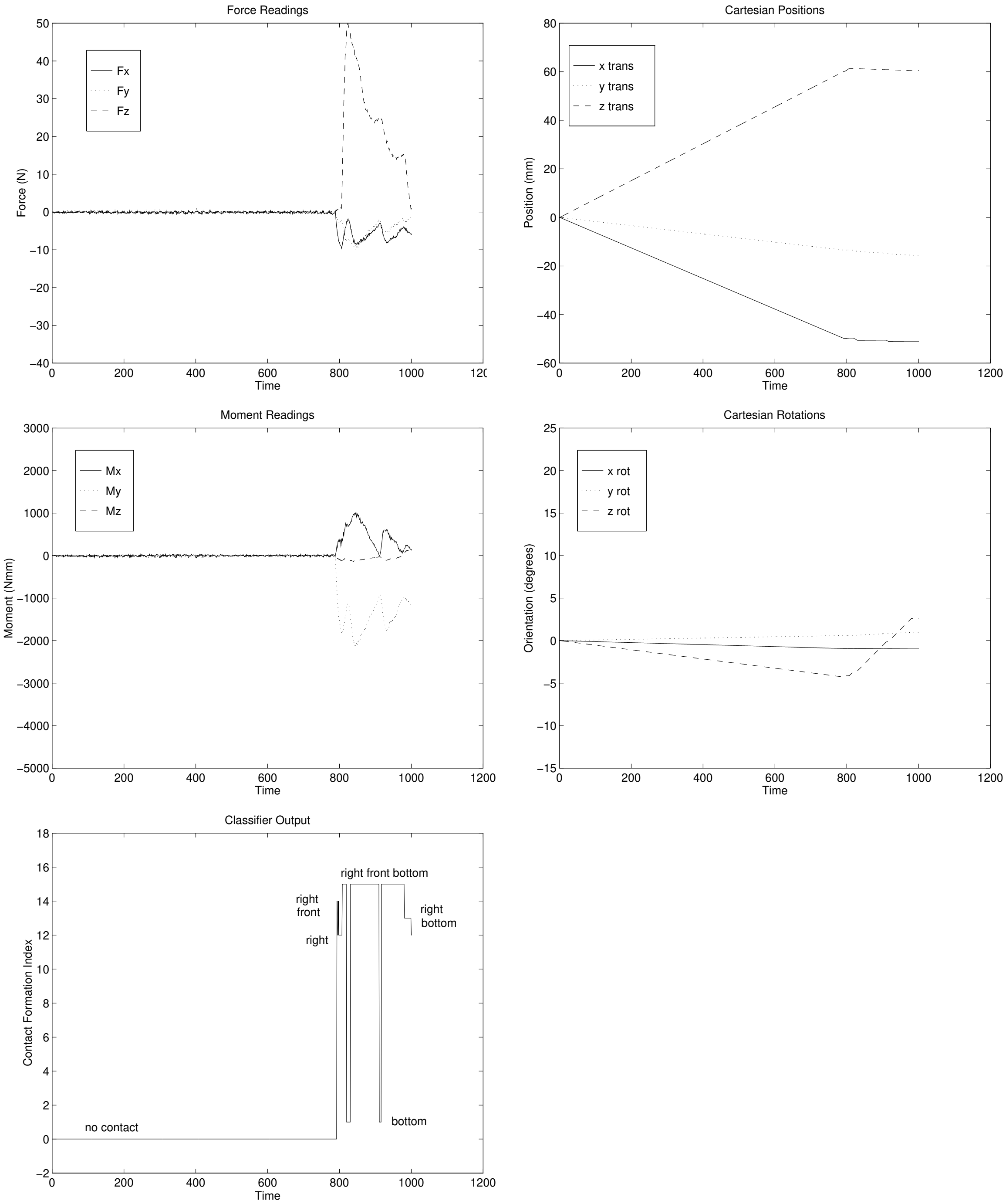


Figure 95. Test Run II.h

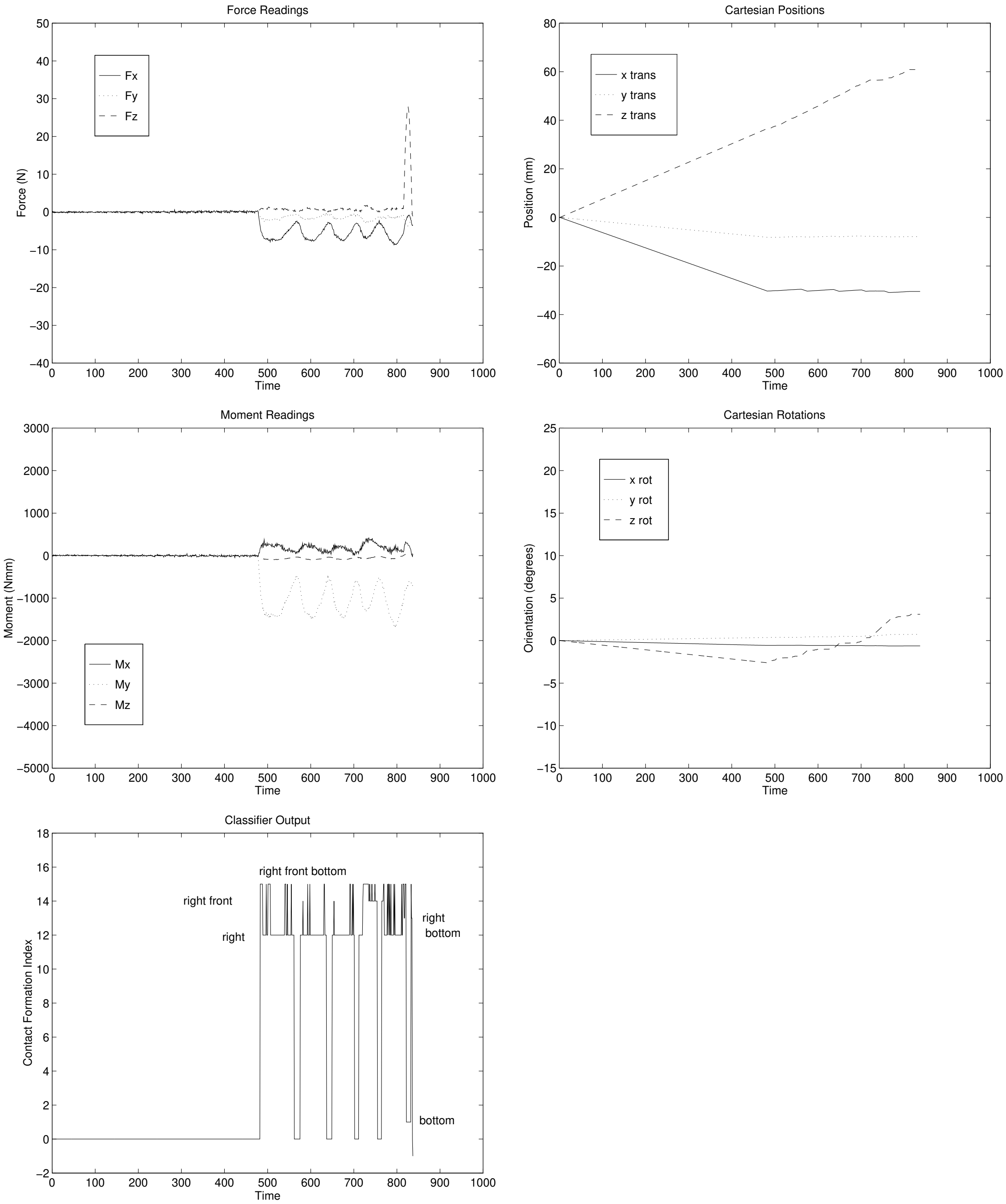


Figure 96. Test Run II.i

APPENDIX B

LETTER OF PERMISSION

VITA

Marjorie A. Skubic received a B.S. degree in Engineering Physics in 1976 from South Dakota State University. For the following fourteen years, she held various software engineering positions in industry, working for companies such as Texas Instruments, TRW, and Staefa Control System, and finally as an independent consultant. In 1990, she began graduate studies at Texas A&I University in Kingsville, TX, and received an M.S. degree in Computer Science in 1992. Immediately following, she began doctoral studies at Texas A&M University and completed a Ph.D. in Computer Science in 1997. While attending Texas A&M University, she held positions as a research assistant in the Robotics Lab, a non-teaching graduate assistant, and a lecturer, all in the Department of Computer Science. Her research interests include sensory perception, pattern recognition, intelligent control, and human-machine interaction, especially as applied to robotics and virtual reality.

Dr. Skubic has accepted a faculty position at the University of Missouri in Columbia, where she will be joined by her husband and two children. She can be reached at the Computer Engineering and Computer Science Department, University of Missouri-Columbia, Columbia, MO 65211.