# Virtual manipulation

Shen Dong
University of Wollongong

**School of Electrical, Computer and Telecommunications Engineering**

# Virtual Manipulation

**Shen Dong**

**This thesis is presented as full requirements for the award of a**

**PhD**

**at the University of Wollongong**

**January 2008**

# ABSTRACT

An empirical research on developing a new paradigm for programming a robotics manipulator to perform complex constrained motion tasks is carried out in this thesis. The teaching of the manipulation skills to the machine commences by demonstrating those skills in a haptic-rendered virtual environment. This is in contrast to conventional approach in which a robotics manipulator is programmed to perform a particular task.

A manipulation skill consists of a number of basic skills that, when sequenced and integrated, can perform a desired manipulation task. By manipulation means the ability to transfer, physically transform or mate a part with another part.

Haptic-rendering augments the effectiveness of computer simulation by providing force feedback for the user. This increases the quality of human－computer interaction and provides an attractive augmentation to visual display and significantly enhances the level of immersion in a virtual environment.

The study is conducted based on the peg-in-hole application as it concisely represents a constrained motion-force-sensitive manufacturing task with all the attendant issues of jamming, tight clearances, and the need for quick assembly times, reliability, etc. The state recognition approach is used to identify and classify the skills acquired from the virtual environment.

A human operator demonstrates both good and bad examples of the desired behaviour in the haptic virtual environment. Position and contact force and torque

data, as well as orientation generated in the virtual environment, combined with a priori knowledge about the task, are used to identify and learn the skills in the newly demonstrated tasks and then to reproduce them in the robotics system. The robot evaluates the controller's performance and thus learns the best way to produce that behaviour.

The data obtained from the virtual environment is classified into different cluster sets using the Hidden Markov Model (HMM), Fuzzy Gustafson–Kessel (FGK) and Competitive Agglomeration (CA) respectively. Each cluster represents a contact state between the peg and the hole. The clusters in the optimum cluster set are tuned using a Locally Weighted Regression (LWR) algorithm to produce prediction models for robot trajectory performing the physical assembly based on the force/position information received from the rig.

The significance of the work is highlighted. The approach developed and the outcomes achieved are reported. The development of the haptic-rendered virtual peg-in-hole model and structure of the physical experimental rig are described. The approach is validated though experimental work results are critically evaluated.

Keywords: Haptic, PHANToM, ReachIn, Virtual Reality, Peg-in-hole, Skill acquisition.

# STATEMENT OF ORIGINALITY

This is to certify that the work described in this thesis is entirely my own, except where due reference is made in the text.

No work in this thesis has been submitted for a degree to any other university or institution.

Signed

Shen Dong

18th of February 2008

# ACKNOWLEDGEMENTS

I would like to extend grateful appreciation and sincere respect to my supervisor, Professor Fazel Naghdy, for his generous advice and guidance, during all stages of my postgraduate study.

I am very thankful to the staff in School of Electrical, Computer and Telecommunications Engineering for their assistance.

I am also deeply indebted to my friends and those who helped me throughout.

A very special appreciation is due to my family members. Without their patience, understanding and encouragement, completion of this study would not have been possible.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| 3D | Three Dimension |
| ACMP | Artificial Constrained Motion Primitive |
| ALVINN | Autonomous Land Vehicle In a Neural Network |
| API | Application Programming Interface |
| APO | Assembly Plan from Observation |
| CA | Competitive Agglomeration |
| CAD | Computer Aided Design |
| CMU | Carnegie Mellon University |
| DAQ | Data Acquisition |
| DRAMA | Dynamical Recurrent Associative Memory Architecture |
| DOF | Degree of Freedom |
| FCM | Fuzzy C-Means |
| FGK | Fuzzy Gustafson–Kessel |
| FMLE | Fuzzy Maximum Likelihood Estimation |
| GHOST | General Haptic Open Software Toolkit |
| GRAVIS | Gestural Recognition Active Vision System |
| HMM | Hidden Markov Model |
| ILP | Inductive Logic Programming |
| ISA | Industry Standard Architecture |

| | |
|---|---|
| LQ | Linear Quadratic |
| LQR | Linear Quadratic Regulator |
| LWL | Locally Weighted Learning |
| LWR | Locally Weighted Regression |
| MLP | Mutli-layer Perception |
| NN | Neural Network |
| NURBS | Non-uniform Rational B-Splines |
| ODMM | Observation-driven Markov Model |
| OLDs | Deformable Liner Objects |
| p.d.f. | Probability Density Function |
| PCA | Principal Component Analysis |
| PbD | Programming by Demonstration |
| PID | Proportional Integral Derivative |
| RMS | Root Mean Square |
| SDK | Software Development Kit |
| SHOSLIF | Self-organising Hierarchical Optimal Subspace Learning and Inference Framework |
| SRPT | Stochastic Recursive Partition Tree |
| SVM | Support Vector Machine |
| VPCM | Virtual Polyhedron for Constrained Motion |
| VR | Virtually Reality |

| | |
|---|---|
| VRML | Virtual Reality Modelling Language |
| VSF | Virtual Small Face |

# CHAPTER 1 INTRODUCTION

## *1.1 The Problem Statement*

Robotics manipulators are primarily employed in industry to improve productivity. Many complex manufacturing tasks such as assembly require force sensitive manipulation. The robotics manipulators are not usually considered an ideal solution for such applications due to the complexity associated with their programming and employment. In fact, one of the major barriers to the utilisation of robots in assembly tasks to date has been the lack of an effective and reliable method to program them to carry out constrained motion.

In a constrained motion manipulation, the object manipulated by the robots is in contact with the surrounding environment or other objects. This requires the manipulator, the object and the environment perform compliant motion along certain directions.

A task cannot be structured accurately in compliant motion. A human operator adapts to uncertainty associated with the task through kinaesthetic information received from the hands. Similarly, a robot arm engaged in compliant motion is dependent on the force and tactile signals received from interaction with the environment. Integration of the sensory data with the position control of a robots is quite complex.

In this study, the feasibility of teaching manipulation skills to a robotic manipulator to perform a constrained motion task is explored. This will replace the conventional method of programming a robot. A skill consists of a sequence of actions which collectively complete a specific task. A human operator develops skills through training and practice in psychomotor domain. This is best achieved by learning the skill from another human being. For a robot, acquiring skills from a human operator has been a viable approach to pursue.

## *1.2 Haptic Rendered Virtual Environment*

In the approach developed in this study, the instructor demonstrates the manipulation task in a haptic rendered virtual environment using a haptic device. The process of haptic-rendering consists of using information collected from the virtual environment, and evaluating the forces and torques to be reacted at a given position, velocity, etc. of the operational point of a haptic interface. The operational point is the physical location on the haptic interface where position, velocity, acceleration, and sometimes forces and torques, are measured.

In order to display a virtual environment, the following problems must be addressed [Pearce 1999]:

(a) Finding the point of contact: This is the problem of collision detection, which becomes more difficult and computationally expensive as the model of the virtual environment becomes more complex.

(b) Generation of contact forces and torques: This creates the “feel” of the object. Contact forces can represent the stiffness of the object, damping, friction, surface texture, etc.

(c) Dynamics of the virtual environment: Objects manipulated in a virtual environment can collide with each other and move in a complicated way.

(d) Computational rate: Computational rate must be high, around 1 kHz or higher, and the latency must be low. Inappropriate values of both of these variables can cause hard surfaces in the virtual environment to feel soft as well as creating system instabilities.

## *1.3 Assembly States*

The state recognition approach is used to identify and classify the skills acquired from the virtual environment. The process of identification of assembly states is quite critical. States not correctly classified can result in the failure of physical system. An assembly consists of a sequence of states. The assembly process itself is the chain of transitions from the initial state to the goal state. Different learning strategies will be applied according to the current state.

State identification methods can be divided into direct [Burdea 2000] and indirect approaches [Potts 2000]. The commonly used direct identification methods are dependent on quasi-static motion with negligible friction and noiseless sensing [Burdea 2000]. Hence, they cannot be applied to assembly tasks performed in an environment with high uncertainty. The indirect methods, on the other hand, do not have such restrictions and can be employed in uncertain environments with friction and noisy sensory signals [Potts 2000].

Consequently, indirect identification methods are more appropriate for this work, and are employed for the classification of a state. The trajectory of the operator is classified into several states. The trajectory is learned by identifying constraints among the state variables in the operation process. These constraints determine the corresponding desired current and next state and decide on the best trajectory to reach the goal. Actions that choose the desired trajectory are computed using knowledge of the system dynamics, learned by nonlinear function approximators. An on-line incremental learning algorithm is used to identify the trajectories of the controller's

skills at each state.

## 1.4 Conceptual Model behind the Approach

A manipulation skill is the ability to transfer, physically transform or mate a part with another part. A specific manipulation skill consists of a number of basic skills that when sequenced and integrated can achieve the desired manipulation outcome. The manipulation task $(M_s)$ is applied to the part by the human operator through an action $u^h(t)$, transferring the part from an initial state of $x^h(t_i)$ to a final state of $x^h(t_f)$. The control action command $u^h$, provides position, orientation, and dimension of the part or its contact forces/torques with the environment. The measured state variables at any instant of time $t$ will represent the output of the manipulation system $y^h(t)$. The variables $x$, $u$ and $h$ are vectors.

The overall approach pursued in this thesis is presented in Figure 1.1. As illustrated in this diagram, the robotic manipulator mimics the behaviour of the human operator by acquiring the skills and producing the machine control action $u^m(t)$ from $y^h(t)$. Different stages of the work are described as below:

(a) The human operator performs the manipulation task in a virtual environment using a haptic device. The haptic device provides the operator with contact forces and torques similar to those in a real life operation.

(b) The information produced in the virtual environment, $y^h(t)$, is used by the Perception module to identify the basic skills and functions employed in the operation and to extract the algorithm sequencing the applied skills.

(c) The information produced in stage (b) is passed to Data Refinement Module in order to remove the noise data, and to extract the most significant data. The refined data is then passed to the Manipulation Task Planner to be translated into position/force/torque trajectories and associated control algorithms for the robotic manipulator. Initially $u^m$ is generated based on the information received from the Perception module, the output of the machine manipulation system $y^m(t)$, prior knowledge about the task according to the following relationships [Pearce 1999]:

$$S_{i+1} = \lambda(S_i, T_i) \quad (1.1)$$

$$u_i^m = l(E_i, I_i) \quad (1.2)$$

In (1.1), $S_i$ denotes the discrete constraints of the task at instant $i$, $T_i$ represents the discrete information occurring at instant $i$, $\lambda$ is the trajectory constraints learning function used to predict the next constraint of the task. The function $l$ stands for trajectory learning function, which computes the output $u_i^m$, based on the current state $E_i$ and the current information $I_i$.

(d) The performance of the manipulation $u^m$ is then compared with the expected behaviour. The manipulator trajectory and $u^m$ are adjusted according to the error to produce a behaviour as close as possible to the manipulation performance by the human.

(e) After satisfactory imitation, information from the Learning Module will be taken into account to calculate $u^m$. The Learning Module performs various optimisation processes to enhance the performance.

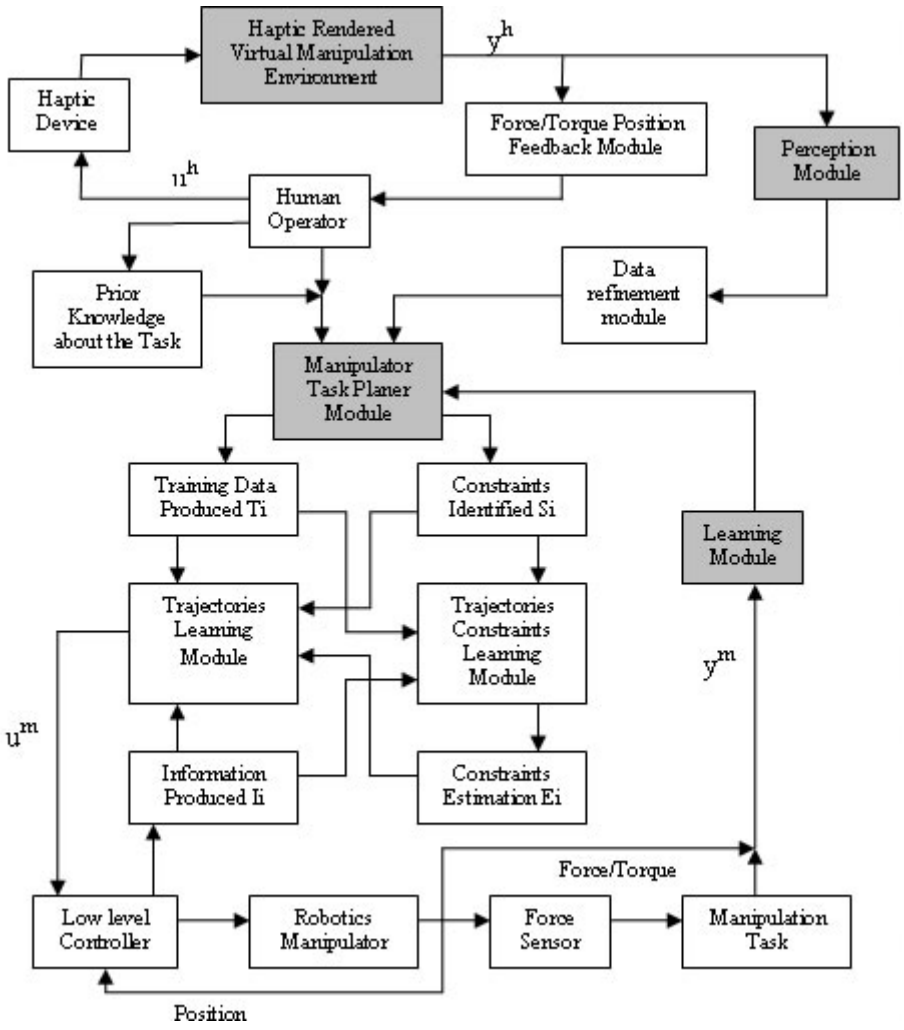



Figure 1.1 Overall model of system

Such a system will be most effective when the Perception and Learning modules are generic. The Manipulation Virtual Environment will be dependent on the application and the Task Planner will be dependent on the manipulator employed.

## *1.5 Significance of the Thesis*

This study has its focus primarily on the developing a haptic-rendered virtual model for the peg-in-hole assembly and classifying the acquired skills by applying the state

recognition methods to the data acquired from the virtual environment.

The peg-in-hole insertion problem is used as a case study, which represents a typical constrained motion force sensitive manufacturing task with the attendant issues of jamming, tight clearance, and the need for quick assembly times. In the developed system, position and contact force and torque as well as orientation data generated in the haptic rendered virtual environment combined with a priori knowledge about the task are used to identify and learn the skills in the newly demonstrated task.

This study represents a significant progress on the work carried out on programming a robotics system through demonstration. It is distinctively different from the previous works, as it offers the following unique and original contributions:

i. Broadly, the project has a generic scope that is novel and innovative. It explores how an intelligent machine can replicate human motor manipulation skills. The model identified for human psychomotor learning will be emulated in the machine to achieve different stages of motor learning.

ii. Hence the proposed work, is unique in its approach and hypothesis, and provides a new insight into the nature of transfer of manipulation skills from human to machine. It proposes new generic intelligent algorithms and methodologies to emulate different stages of human psychomotor learning in machines, including perception, imitation, mechanism, and complex/overt response. This makes the project significantly different from previous work depending on machine learning in the cognitive domain.

iii. More specifically, the project explores the feasibility of online transfer of physically constrained manipulation skills from a human operator to a manipulator through demonstration, utilising a virtual training environment with tactile sensing. Constrained motion manipulation is an important aspect of any realistic manipulation environment, such as automatic assembly.

iv. The acquisition of manipulation skills begins with demonstrating the necessary skills by the human operator in a virtual environment with tactile sensing (haptics). The use of the virtual environment simplifies the process, as the training data is acquired directly from the haptic system.

v. The project also explores how more complex manipulation skills can be constructed from basic skills, as words are formed from letters.

vi. The proposed concepts are at the leading edge of tele-robotics and intelligent systems research.

## *1.6 Overall Approach*

The project will be carried out by realising the following stages: Virtual Learning Environment, Perception, Imitation, Forming Habits, and Validation,

### 1.6.1 Virtual Learning Environment

The data used by the machine to acquire basic manipulation skills is generated through a haptic-rendered virtual environment. This approach offers a number of advantages compared to other methods. The training data (for example, velocities, angles, positions, forces and torque) can be extracted and recorded directly, which

simplifies the data-collection process. The environment can be easily modified and changed as the manipulation process and its requirements are changed. The risk of breakdown and breakage of the system is very low. Dangerous and costly environments can be developed in the virtual environment without associated risks.

The peg-in-hole insertion with a tight fit representing a constrained motion manipulation is used to demonstrate the feasibility and validity of the developed concepts and methodologies. A six degrees of freedom (6 DOFs) haptic device, called PHANToM (PHANToM Premium 1.5 from SensAble Technologies Inc., USA) is deployed [SensAble (online)]. A ReachIn hardware platform and application programming interface is used in the implementation of the virtual manipulation environment [ReachIn (online)].

### 1.6.2 Perception

The second stage will be to explore a methodology through which the behaviour of the operator using the haptic device can be perceived by the Perception Module. In reality, the key question asked will be how the sensory data produced by the virtual environment can be translated into a pattern of behaviour for training and operation of the physical manipulator. Such behaviour cannot be simply modelled analytically or explicitly described by the static and dynamic equations of motion. This is due to the complexity of the operations involved and the inadequacy of analytical methods to model such behaviour.

### 1.6.3 Imitation

As the third task, the perception of the operator’s behaviour should be translated into

appropriate control strategies for the physical manipulator to imitate the manipulation. This is carried out by the Manipulator Task Planner module. Such a control strategy will be hierarchical. At the lower level, the individual joints of the manipulator will be controlled using a conventional method. At the higher level, the task planning for the robot arm will be carried out as fine-motion planning to achieve certain contact formations between the parts involved to satisfy force constraints defined by the classifier. Part-mating can be then defined as moving from an initial contact formation to a goal contact formation.

### 1.6.4 Forming Habits

Another function of the learning module is to enhance the performance of the manipulator over time and to produce a complex and overt response. The aim is to gradually phase out the error-based corrections at higher control levels and replace them with appropriate actions according to every new situation encountered. This will be similar to developing habits in psychomotor taxonomy and forms the fourth stage of the project.

### 1.6.5 Validation

The peg-in-hole insertion task will be virtually defined through the haptic device and the solid modeller. The developed methodology will be then applied and the validity of the developed perception, control algorithms, and habits will be systematically studied. This will result in further modification and enhancement of the algorithms and techniques developed.

### 1.6.6 Proposed Experimental Rig

The experimental system consists of a 6 DOF articulated manipulator attached to a 6 DOF force/torque sensor. A 6 DOF haptic device is added to the experimental rig for training of the manipulation system. The force/torque data produced by the haptic device will match the information produced by the force sensor. Initial verifications of some of the basic strategies will be carried out on the physical peg-in-hole insertion experimental rig. The host PC will receive the sensory information and training data from the haptic device and will produce the necessary control signals to drive different actuators of the manipulator. It will be necessary to interface the host computer with the controller of the robot arm.

## *1.7 Structure of the Thesis*

The thesis is organised as follows. A review of the literature associated with the study will be carried out in Chapter 2. In Chapter 3, the concept of haptic rendering will be introduced and three popular haptic-rendering technologies will be reviewed. In addition, Haptic-rendering tools including PHANToM, GHOST SDK, VRML and ReachIn API, are described. Penalty-based rendering and proxy-based rendering methods employed in the peg-in-hole insertion procedure will be explained. The algorithm used to calculate the collision force and torque of peg and hole are described. An analysis of the force and torque data generated by the haptic-rendered virtual environment will be also carried out in this chapter.

An overview of the acquisition of manipulation skills from haptic-rendered

virtual manipulation will be provided in Chapter 4. The concept of acquisition of manipulation skills from haptic-rendered virtual manipulation will be also explained in this chapter. Skill-acquisition based on behavioural cloning methods, the Hidden Markov Model and Fuzzy Clustering algorithms, will be explained in detail.

In Chapter 5, the experimental validation of the algorithms of the Fuzzy Gustafson–Kessel (FGK) and Competitive Agglomeration (CA), and Hidden Markov Model (HMM) will be provided. The physical peg-in-hole experimental rig proposed to employ these algorithms in the real task will be introduced first. The experimental rig will be described and the experiments designed to validate the work will be presented. Finally the validation results will be provided.

Chapter 6 will draw some conclusions and will highlight the main contributions of the thesis. Some suggestions for further research on the project will be also provided.

# CHAPTER 2 BACKGROUND

## *2.1 Introduction*

This work explores the feasibility of reconstructing human manipulation skills in complex constrained motions by tracing and learning the manipulation performed by the operator. Manipulation skills are taught to machine by a human operator demonstrating these skills in a haptic-rendered virtual environment. This represents further extension of the concept of “teaching by showing” and can be considered as a new paradigm in programming robotic manipulators.

In this chapter, a review of the previous work associated with this concept is carried out. This will highlight the significance and the contribution of the work.

## *2.2 Programming of Robotics Manipulators*

While the development of sophisticated intelligent humanoid robots has been the ultimate goal of robotics research, other factors such as cost saving have been driving research and development in robotics. In some manufacturing sectors, such as welding, material-handling, spray-painting and assembly, robotics research community has exerted significant effort to design user-friendly interfaces and more powerful programming methods for robotic manipulators. The extensive research conducted by Blume [Blume 1983] and Ránky [Ránky 1985] on the historical evolution of robotic programming highlight the results produced.

The general robot programming paradigm, as shown in Figure 2.1, characterises the real physical manipulation tasks and abstract models representing these tasks. Abstract models can be realised by different robot programming methods. In majority of methods, commands based on the abstract models instruct robots to perform certain manipulation tasks and change the states both in the abstract model and the real task. Internal robot sensors and external sensors, such as force/torque, position, ultrasonic and vision sensors, are used to observe the sequence of actions in both the abstract model and the real world, and ensuring that they are kept consistent with each other [Jones 2004].

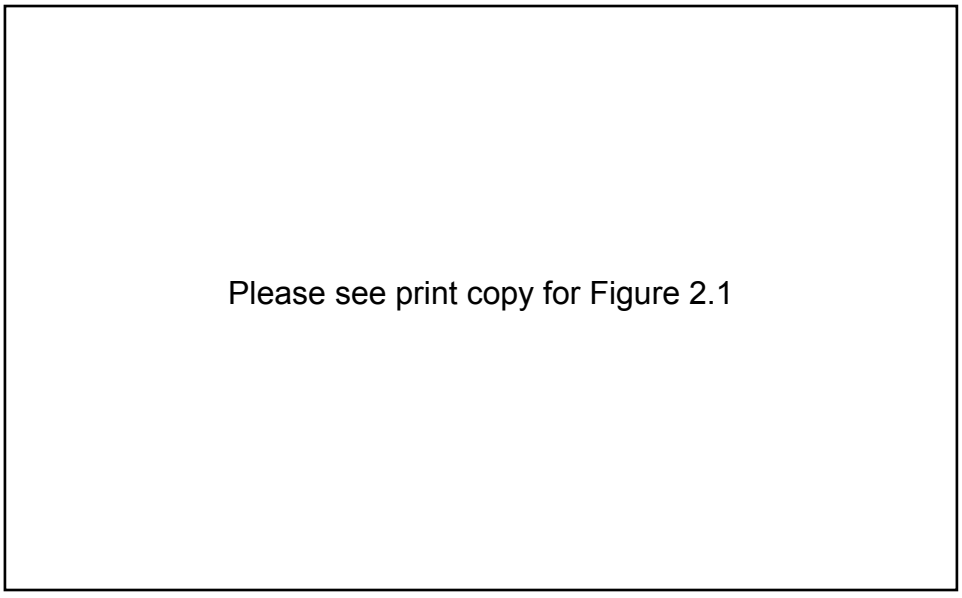



Figure 2.1 General robot programming [Jones 2004]

The traditional programming methods developed for robotics manipulators can be grouped into four categories of text programming, off-line simulation-based programming, inductive learning and teaching by guiding.

Text programming can be applied to complex applications, but it has proven to be quite intensive in code and computing time. The development time is long, and special skills and much effort are required to produce a complete program. This has resulted in the development of task-level robot languages [Caine 1989; Perez 1977]. Task-level programming enables the user to specify the desired goals of the tasks without defining every movement of the robot in detail. The task planner will express each task in terms of necessary manipulator motions and actions. This scheme requires the system to have the ability to perform many planning tasks automatically. In addition, task-level programming tools require a great deal of information about the workspace, the robots, the objects, the initial state of the environment and the final goal to be reached, which can be extremely tedious and time-consuming. A structured text program is usually composed of several major parts, which are Data Types (including derived Data Types), Variable Declarations, Operators and Expressions, Condition Statements, Iteration Statements, and Functions, depicted in Table 2.1 [Craig 2005].

Please see print copy for Table 2.1

Table 2.1 An example of major text programming components [Craig 2005]

The off-line simulation-based methods usually integrate text-programming and model-based motion planners in one common platform [Matsubara 1985; Derby

1982]. Model-based motion planners can automatically generate motions from virtual reality models on graphical simulation platforms so that teaching a robot to perform the task by programming can be relatively easy. The RoboCell robotics off-line simulation programming environment for SCORBOT robots of Intelitek is an example of this method, as shown in Figure 2.2 [Intelitek (online)]. This approach is powerful but requires special hardware and a complete description of the real world, both of which are costly. Although current off-line programming systems mostly provide high-level manipulation language to simplify the programming procedure and comparatively shorten the development, off-line programming environments do not address the issue of sensor-guided robot actions. They are also mostly limited to kinematics or dynamics simulation of a robot, without the provision of advanced reasoning functionality and flexibility in the tasks.

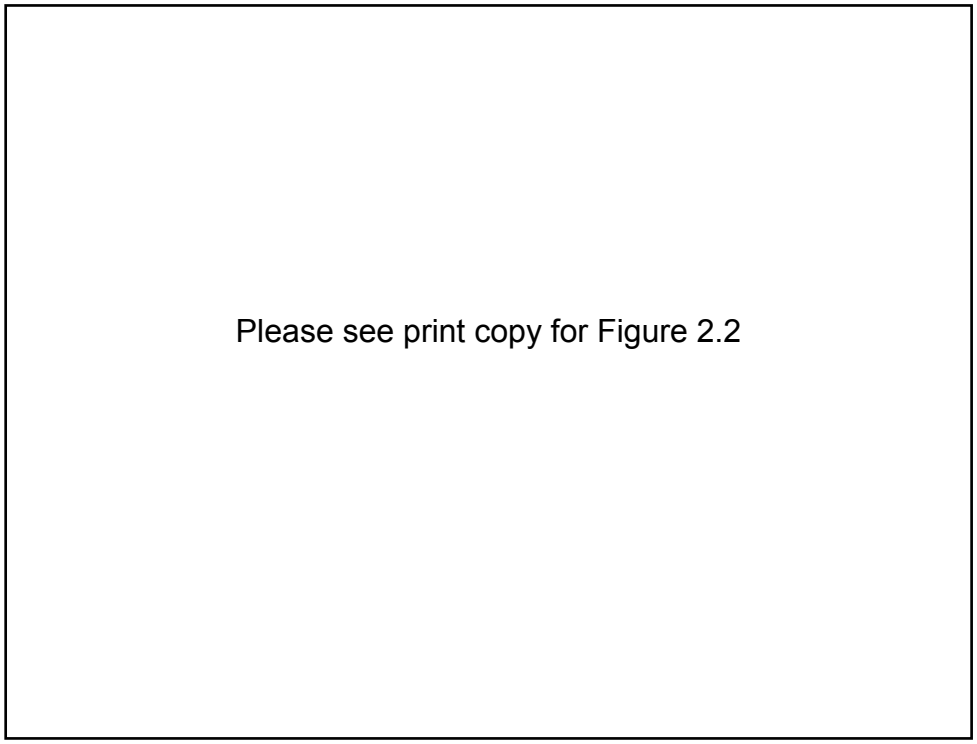



Figure 2.2 Off-line simulation programming environment [Intelitek (online)]

Inductive learning, also called learning from example, is one of the most important approaches developed for programming a robot arm. In inductive learning, a robot arm controls appropriate motion and sensing strategies through trial and error [Dufay 1984]. This is an effective method when it is used to refine other programming methods, but it is not suitable for very complex tasks. The inductive learning rules are acquired, generalised or statistically analysed from a large number of training examples. The performance of inductive learning methods can be measured by the learning curve, which shows prediction accuracy as a function of the observed examples. Inductive learning decision rules can be generalised from data-mining tools,

such as the See5 decision tree algorithm, or from other statistical analysis methods, such as the Hidden Markov Model (HMM), clustering algorithms, or artificial neural networks. Figure 2.3 shows an example of a Cross-Referencing Classifiers produced by See5 data-mining tool. The inductive learning rules are also generalised [See5 (online)].

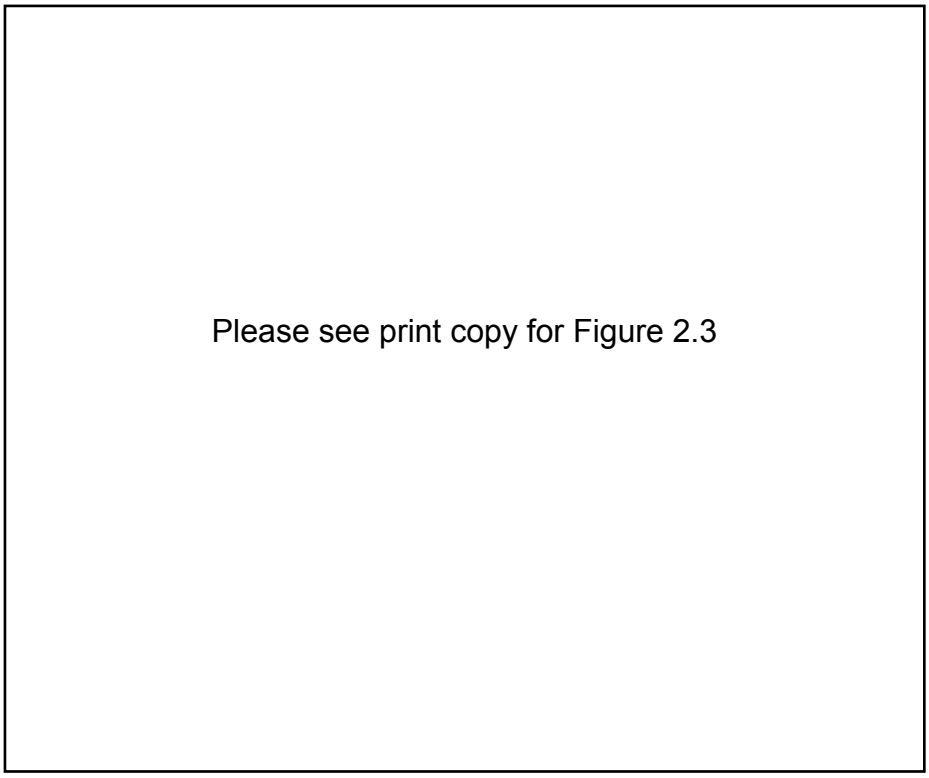



Figure 2.3 Inductive learning rules generalised from See5 [See5 (online)]

Teaching by guiding is a simple method in which a human operator drives a robot's end-effector to all the appropriate locations in the real world to perform the task, while the characteristics of the motion are recorded. In spite of its simplicity, this

method is prone to error, is less portable and has high risk. It is not generic or flexible, and is not applicable to complex tasks. In addition, this method cannot accommodate extensive sensory interaction and can be dangerous for the operator.

The essential component of the teaching by guiding is illustrated in Figure 2.4 [Jones 2004]. The approach consists of four major steps, namely, data acquisition, trajectory reconstruction, task description, and command generation module.

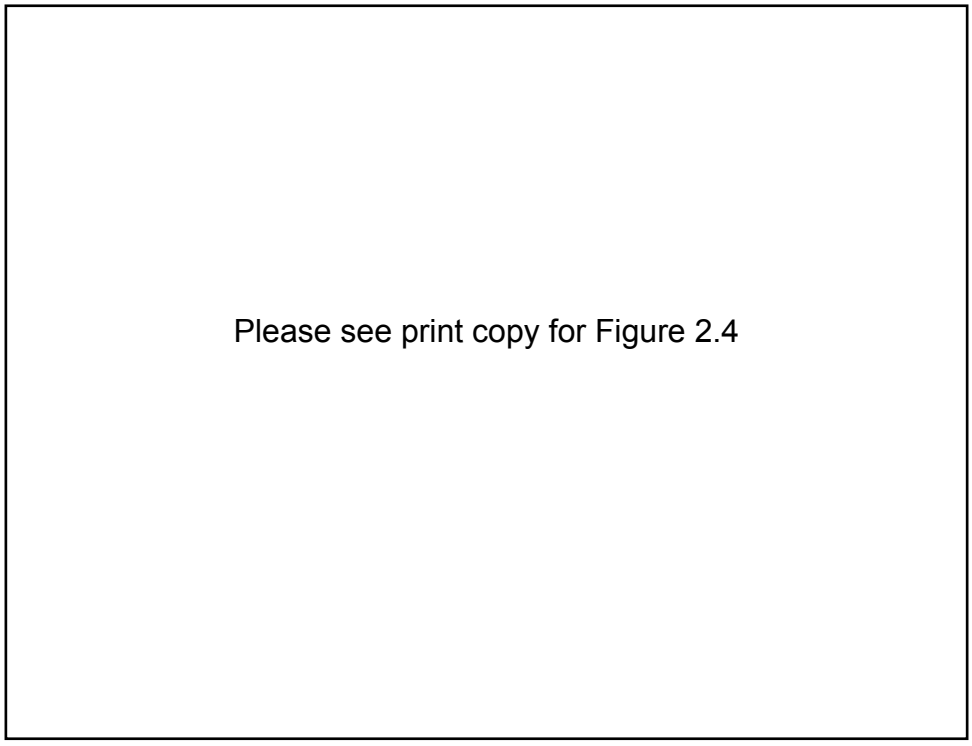



Figure 2.4 Teaching by guiding system [Jones 2004]

## *2.3 Overview of Teaching by Showing/Demonstration*

There has been a number of attempts to overcome some of the shortcomings of the "teaching by guiding" approach. Summers and Grossman [Naylor 1987] embedded a collection of sensory information and interaction with the operator in the task instruction procedure. The concept of "teaching by showing" or "teaching by demonstration" has been another extension of "teaching by guiding", in which a robotic system learns a particular task by watching a human operator performing it. The best method allows the user to make natural movements which can be mapped easily to instruct driving the robotic manipulator (Figure 2.5) [Jones 2004]. This method suffers from a number of shortcomings including the robot operation downtime, the danger exposed to the human operator and the difficulty of making adjustments for new products.

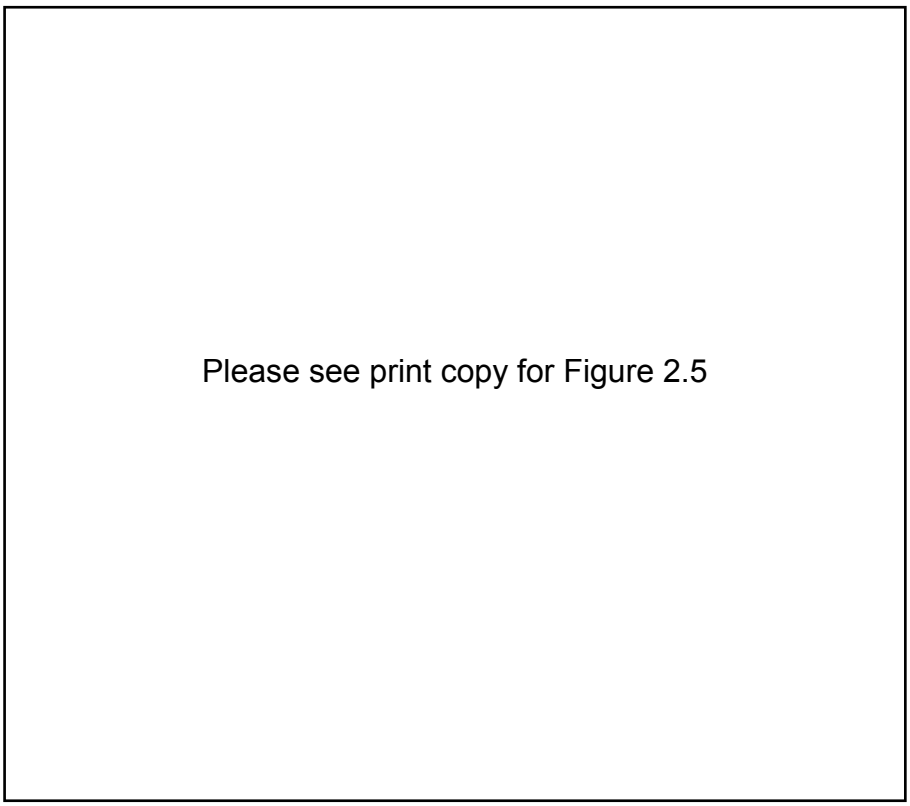



Figure 2.5 Teaching by showing [Jones 2004]

### 2.3.1 Teaching by Showing/Demonstration without Model

Some recent developments have significantly advanced the “teaching by showing” approach in programming a robotics manipulator.

Ikeuchi and Sehiro [Summers 1984] developed a system that could extract a fine motion sequence from the transitions of face contact states obtained by a range of sensors [Ikeuchi 1991].

Smart and Kaelbling proposed a framework for application of reinforcement learning in programming mobile robots [Smart 2002].

Hass integrated a symbolic recogniser and play back module using a visual servo for two-dimensional pick-and-place operations [Hass 1991].

Yamada and Uchiyama conducted a study to determine the essential features of human physical skills based on multi-sensory data and the possibility of transferring them to robots by focusing on two tasks of crank rotation and side matching [Sato 1997].

Kuniyoshi and Inbana developed a robotics system that could learn reusable task plans in real time by watching a human performing assembly tasks [Kuniyoshi 1994]. The method was based on visual recognition and analysis of human action sequences. The effectiveness of the method was demonstrated for a block assembly task.

Montgomery and Bekey developed a model-free “teaching by showing” methodology, which trains a fuzzy neural controller for an autonomous robot helicopter [Aleotti 2004]. The controller, combined with a hybrid fuzzy logic controller and general regression neural network controller, is generated and tuned using training data gathered while a teacher operates the helicopter.

A vision-based approach to robot path planning has been reported in literature [Ude 1994]. A desired motion is demonstrated by a demonstrator manipulating an object with his hand. The manipulation performance is measured with a stereo vision system. A non-parametric regression technique with robustness to measurement noise

is used to reconstruct the demonstrated motion. The generated path is given as a linear combination of natural vector splines.

The approach of programming by demonstration is utilised in the rehabilitation robotic system FRIEND to execute a daily life action of pouring a beverage into a glass [Martens 2001; She 2003]. A pouring trajectory, which is independent from specific bottles and glasses, is acquired through human demonstration. In a given pouring task, this trajectory is used as the motion reference of the bottle, where the actual location in the trajectory is controlled by the beverage flow [Ruchel 1999]. The robustness of the approach is improved by means of abstracting the pouring trajectory independent from specific objects, as well as integrating additional sensor information to control the trajectory.

Voyles et al explored another type of teaching by showing using a programming robotic agent. The approach, known as gesture-based programming [Voyles 1997; 1999a; 1999b], teaches a manipulation to the robotics arm using demonstration by a human operator. Robotic skills obtained from skilled experts used in the development of robotic systems. The gesture-based programming deals with the issues of how gesture-based programming systems captures the intention behind finger poses, hand motions, contact conditions, and even cryptic utterances in real time. The system retains previously acquired skills to enhance gesture interpretation during training while providing feedback control at the same time.

Nicolescu, from Interaction Lab of University of Southern California, has developed an approach for teaching robots that relies on the key features and the general approaches people use when teaching each other [Nicolescu 2001; 2002; 2003]. The method is initiated from an expert's demonstration, and then the learner practises the task several times under the expert's supervision to refine the acquired skills. The expert may demonstrate the task again for several times during the learner's practice, depending on the complexity of the task. The teacher can also provide simple instructions and informative clues to help the learning performance during the practice process. In this method, expert's demonstration, generalisation of the demonstration and the learner's practice the essential components. Additional information beyond the expert's demonstration is also provided to the robot through verbal instruction, enabling the robot to learn the more effectively.

Billard et al has reported experiments conducted to teach certain behaviour to a doll robot through demonstration [Billard 1998]. The doll robot has the capacity to learn, imitate and communicate. The robot can imitate the arm and head movements of the demonstrator through the robot's simple phototaxis behaviour. In the experiments, different sequences of actions are taught to the robot, and labelled accordingly. In further experiments, the robot is taught to grammatically correct sentences and to describe its actions and perceptions of touch on different parts of its body. The robot is controlled by a Dynamical Recurrent Associative memory Architecture (DRAMA) [Billard 1999]. DRAMA is a fully connected network with

self-recurrent connections on each unit that are associated with two parameters, a time parameter, which records the time delay between activation of each unit linked by the connection, and a confidence factor, which records the frequency of activation of the connection.

Billard and Mataric have developed a model to imitate two-arm movements [Billard 2001a; 2001b]. The model investigates a valid model of a biomechanical simulation of human movements, developes an architecture for visuo-motor control and built biologically plausible models of animal imitative abilities. The model consists of a hierarchy of artificial neural networks. It gives an abstract and high-level representation of the neurological structure from brain regions that are involved in visuo-motor control. The model is validated in a biomechanical simulation of a thirty-seven degrees of freedom (37 DOFs) humanoid robot. Data from the human arm movements are recorded using video and marker-based tracking systems. The results show high qualitative and quantitative correlation between the human data and the performance of the humanoid robot.

The Light-Weight Robots project, developed by the German Aerospace Centre, provides a practical example of teaching the task of automatic insertion of a piston into a motor block [Light-Weight (online)] to a robotic manipulator. Teaching commences by guiding the robot by a human demonstrator using the internal torque sensing. The axes of the holes in the motor block are vertically oriented. In the teaching stage, orientation stiffness is assigned a high value, while the translational

stiffness is set low values, allowing only translational movements to be carried out by the human demonstrator. In the second state, the teaching trajectory is automatically reproduced by the robot. High values are assigned for the translational stiffness, while the stiffness for the rotation is low. This enables the robot to compensate for the remaining position errors. The result shows that the automatic assembly task is executed four times faster than by the human operator in the teaching stage. The insertion task had been implemented previously using an industrial robot and a compliant force－torque sensor. The insertion process had to be performed much slower to avoid possible jamming. A well-tuned Cartesian force controller is also deployed.

Yano has described a novel teaching method for a mobile manipulator [Yano 2003]. In this method, the user teaches a nominal trajectory of the hand to a service robot to perform a fetch-a-can task. The task consists of opening the door of a refrigerator, picking up the can, and closing the door. In the teaching stage, the user does not explicitly consider the structure of the robot. The focus is on the movement of the hand and the relationship between the hand and the manipulated object. After searching for a sequence of feasible hand positions and orientations within the given tolerance, the robot generates a feasible trajectory. Although the nominal trajectory may be infeasible due to the structural limitation, the robot can search for a feasible trajectory within the given tolerance. If necessary, each divided task can be performed at a fixed location by dividing the task into subtasks. The advantages of the proposed

method are that a user can teach a task without explicitly considering the structure of the robot, and the robot can generate a feasible trajectory relatively easily from the given task specification.

Steil et al have investigated the use of gestures for controlling a vision-based robot called the GRAVIS-robot (Gestural Recognition Active Vision System robot) [Steil 2001]. The robot consists of a binocular camera head, a six degrees of freedom (6 DOFs) robot arm and a nine degrees of freedom (9 DOFs) multi-fingered hand. Nonverbal communication based on gestural commands of a human instructor is used to direct the attention of the robot so that it can enable its vision system to more easily find objects that are specified in instructions.

As an extension of the GRAVIS-robot (Gestural Recognition Active Vision System robot), a multi-model system is developed to provide a more cognitively oriented environment [Steil 2001; McGuire 2002]. Information from various sources, including vision, gestures and voice may be used. For example, an instruction to pick up a cube could be given by voice, indicated by a gesture that indicates to pick up the cube, or signalled by the vision system by reflecting an infrared ray on the cube.

A method using hand gestures to control a domestic cleaning robot has been developed by Strobel [Strobel 2002]. In this work, a robot's stereo vision system is used to capture the static hand and arm gestures, while the magnetic tracking system is used to capture the dynamic gestures. The cleaning robot also has spatial

knowledge, which is used to reinforce the robot's intention. In the course of cleaning, the user could point to a surface that should be cleaned.

Lauria et al have designed a natural language system that can provide directions to a robot [Lauria 2002]. This system is used to teach the robot to find different locations by travelling through specified routes. The system includes fourteen primitives with natural language constructs. Unknown commands may be used by the user in the course of the project performance, and some form of clarification and learning system would be needed.

### 2.3.2 Teaching by Showing/Demonstration in an Interactive Environment

Direct transfer of skills from a human operator to a machine in an interactive environment has been the next stage in the programming and training of a robotics system. In the field of mobile robots, Pomerleau used a three-layer perceptron network to control the CMU ALVINN autonomous vehicle [Pomerleau 1993].

Grudic and Lawrence used an approximation method as a means for creating the robot's mapping from sensor inputs to actuator outputs in the transfer of skills to a mobile robot [Grudic 1996].

In the acquisition of manipulation skills, particularly in constrained motion, the work carried out by Kaiser and Dillman is of significance [Kaiser 1996]. The work proposes a general approach to the acquisition of sensor-based robot skills from

human demonstration. An adaptation method is also proposed to optimise the operation with regard to the manipulator. The method is validated for two manipulation tasks of the peg-in-hole insertion and opening a door.

Myers et al from Intelligent Automation, Inc. (IAI), developed a system that learns from demonstrations of a human operator and then produces a sensor-driven program which controls a robot without operator intervention [Myers 2001]. The system generates an operation procedure, which is identical to conventional operation procedure developed from robot programming language that can be easily reused in similar but different application tasks. Based on the skills demonstrated by a skilled operator, a robot can be automatically programmed to perform complex assembly tasks. As the task requirements change, the robot can be reprogrammed as easily as it was initially programmed.

Skoglund and Aleotti introduced a new “programming-by-demonstration” approach, using a supervised learning method [Aleotti 2004; Skoglund 2005]. A “programming-by-demonstration” system prototype is presented for position teaching of a robot arm. The data of human arm movements are recorded from a wearable input device, and is used in the software controller of a robot arm. The method does not require analytical modelling of either the human arm or robot, and can be customised for different users and robots.

Soshi et al introduced a novel approach for programming robots interactively through a multimodal interface [Iba 2005]. The key characteristic of this novice-

friendly system is its intuitive interfaces based on speech and hand gesture recognition, and its ability allowing the user to provide feedback interactively during the time of both the programming and the execution. By allowing human control during the time of execution, this interaction capability helps the user to deal with loosely calibrated position sensors. The system is demonstrated by interactively controlling and programming a vacuum-cleaning robot. Instead of off-line robot programming, this on-line robot programming method enables the user to identify what to expect from the program execution.

A robot's vision-related and audition-related learning capabilities are detailed in literature [Weng 2000; Zhang 2001]. The teaching of the robot is carried out online in real time through physical interaction between the trainers and the robot. The self-developmental program of the robot generates internal representations and architecture autonomously during the learning procedure without any visual or acoustic model information about the world.

Ehrenmann et al have developed a methodology to teach actions performed in by an operator in household situation to a robotic system [Ehrenmann 2002]. The method work by segmenting between actions performed during a grasp process. The actions include the recognition of particular user actions, the task representation and the mapping strategy itself. The results of the segmentation can be stored to be stored for application to a real robot.

Chen et al presented a framework for robot programming by human demonstration [Chen 1998; 2000; 2001]. The framework builds a high-level robot controller using information extracted from the demonstration. The high-level robot controller is broken down into steps, each fulfilling a different function during execution. Multiple demonstrations are used to build a partial view of the robot's configuration space. Optimal paths are generated between steps in a task. The assumption is that demonstrations rarely contain the best path between steps. This introduces a significant variation to task performance, as the task can be biased towards maximum execution speed or maximum accuracy.

The use of sensors on the fingers to detect fine manipulation of objects has been studied by Zollner [Zollner 2002]. Finger movements and forces on the fingertips are gathered and analysed while an object is grasped. The approach deploys a data glove and integrated tactile sensors. For classifying a grasp, a time delay method based on a Support Vector Machine (SVM) is used.

### 2.3.3 Teaching by Showing in Virtual Environment

Handleman and Lane have carried out some preliminary work on a knowledge-based "tell" approach to describe the task to be carried out by the robot and the required corrective control measures to be taken up [Handleman 1996]. The task is defined by a rule-based, goal-directed strategy. The proposed method has been verified through computer simulation only for a typical peg-in-hole insertion problem. The

development of the rule-based system has been intuitive and rather complicated. The developed rules are very much context based and have to be built from scratch for any new application.

Takamatsu and colleagues developed a system that referred to as the Assembly Plan from Observation (APO) system [Takamatsu 2000]. The robot has the capability of observing human performance, recognising tasks and generating programs that perform the same task within the APO system. The robot's hand movement is used as a case study in the project. Trajectory is created after observing a human performance. In further extension of the work, a task model is developed by integration of observations made on multiple demonstrations on a single task [Ogawara 2002]. The data obtained from demonstration is segmented to find important states such as grasps and moves. The multiple demonstrations highlight which segments is important in the task. The relative trajectories corresponding to each essential interaction are generalised and stored in the task model by calculating their mean and variance. The skilled behaviour is reconstructed from these trajectories. Further study undertaken by Takamatsu et al has produced a more robust algorithm by correcting possible errors in the demonstrated data [Takamatsu 2002]. Two methods are used to clean up the errors from the vision system by using contact relations and their transitions. The first one corrects the observed configuration from the observed contact relations. The second approach identifies wrongly classified contact relations from an analysis of configuration space. Contact states are checked to ensure they do not create problems

such as having two objects in the same location. This ensures that incorrect results from a vision system do not produce erroneous trajectories.

Yuan et al investigated assembly planning in virtual environments [Yuan 2000] [Yuan 2004]. This interactive approach of assembly planning provides a hand-based interface for human operators to perform assembly operations directly in the virtual environment. It also incorporates a biologically inspired neural network into its assembly planner to automatically determine collision-free motion paths at run time. As a result, this approach creates a favourable and unique feature to enable interactive assembly planning, not only producing alternative assembly sequences from a single user-defined assembly sequence, but also providing assembly operations with robot-level instructions.

Ogasawara et al proposed an integrated teleoperation system for maintenance consisting of a task with teaching function and another with execution function [Ogasawara 1998]. The system integrates a motion teaching system, a geometric modelling module, and a task execution system. Automatic analysis of the contact states of objects is embedded into the motion teaching system to help the operator to teach assembly motion. A surface-based geometric modelling system uses the “Teaching Tree” method. This is generated from geometric data of the object. Manipulation skills and used to combine the planning system and the task execution system.

Onda et al developed a “teaching-by-demonstration-in-VR” system that automatically generates a robot program to work in the real world from the assembly tasks performed by a human operator in a virtual environment [Onda 2001; 2002a; 2002b]. The system generates a program which detects each contact state extracted from demonstration, and realises it by skilful motion primitives (SMP). Some types of motion that are included in the recorded motion given by a human operator in this VR system in the teaching stage cannot be realised at the execution stage if the robot tries to do the same task by using the human demonstration data, even on a contact-state level. A method, called the “non-deterministic-search-type motion”, is proposed to deal with non-deterministic motions when contact-state transitions are made non-deterministically. A virtual small face (VSF) and an artificial constrained motion primitive (ACMP), as well as a virtual polyhedron for constrained motion (VPCM), are introduced. This provides the ability to specify a more general arrangement of the bodies when one body is not in contact with another. In a virtual environment, new states as well as contact states should be known to make the robot more skilful.

An approach that allows the results of the demonstration to be graphically viewed via a 3D simulation and graphical user interface is developed by Friedrich [Friedrich 1998]. The user is able to supervise the operation of the robot during program-generation process. The developed code can be also edited, moved around and even used separately.

### 2.3.4 Other Alternatives to Teaching by Showing

The “teaching-by-zooming” is another alternative to the “teaching-by-showing” method [Dixon 2003]. Visual servo control objectives are formulated by the desire to position/orient a camera based on a reference image obtained by priori positioning the camera on the desired location. Specially, projective geometric techniques are used to formulate a visual servo control problem based on a cooperative camera scheme. By using a second camera with zoom capabilities, the proposed “teaching by zooming” alternative approach eliminates the need for the camera to be priori positioned on the desired location.

The “teaching-by-showing-few-images” method can be applied in navigation of autonomous mobile robots [Matsumoto 2003]. Some scenes are shown by humans as the features of the environment combined with the motion information. An autonomous mobile robot navigates paths between feature points by comparing current image information with given image information. The robot reads the motion command which is associated with the given image information when the robot determines the images are identical.

Yokokohji et al has proposed a teaching-by-demonstration method for training of humanoid robots at home [Yokokohji 2005]. The demonstrator’s motion is captured by a pair of stereo cameras mounted on her/his head, located very close to her/his eyes. By tracking the landmarks attached to the demonstrator’s hand and the working

environment, the algorithm estimates not only the demonstrator's hand motion but also his/her head motion, which can be used for active vision system.

Laschi et al proposed the application of robot vision to the identification of hand posture [Laschi 2002]. A skeleton-based approach has been implemented to achieve a high-speed low-cost implementation, instead of identifying surfaces and volumetric primitives. Simplified one-dimensional segment-based models, which are generated in a symbolic way from a selective grouping of points and segments, are used. It is expected that the effectiveness of man—machine interaction in personal robotics will be improved when robots are introduced in unstructured environments in the presence of human beings in applications such as assistance of disabled or elderly persons.

Grunwald et al introduced a "programming-by-touch" method that presents how an untrained user can intuitively interact with the new DLR light-weight robot just by touching the arm [Grunwald 2001]. The seven degrees of freedom (7 DOFs) robot equipped with appropriate sensors can sense the touch. Instead of demonstrating the skills for the robot by gripping the robot arm at a certain point to move it, the demonstrator may hold the robot arm at any point, much as s/he would hold a human arm. The robot is easier and more natural for a non-technical person to use.

A vision-based indoor navigation system for robot navigation has been investigated by Chen [Chen 2000]. The self-organising hierarchical optimal subspace learning and inference framework (SHOSLIF), incorporating states and a visual attention mechanism, are used. This vision-based navigation is formulated as an

observation-driven Markov model (ODMM), which can be realised through recursive partitioning regression. It keeps the history information and incoming video input as an observation vector. A stochastic recursive partition tree (SRPT), which maps a pre-processed current input raw image and the previous state into the current state and the next control signal, is used for efficient recursive partitioning regression.

A developmental robot, which has the ability of developing its cognitive and behavioural skills through real-time interactions with the environment, has been introduced by Zhang [Zhang 2005]. The robot's learning ability is presented with the emphasis on its audition perception and audition-related action generation. The robot conducts the auditory learning from unsegmented and unlabelled speech streams without any prior knowledge about the auditory signals. The actions that the robot is expected to perform are also not available. The robot learns the auditory commands and the desired actions from trainers as well as physical contact with the environment.

A developmental cognitive learning architecture, which learns simple behaviours and chains these together to form complex behaviours by an artificial agent, has been developed by Zhang [Zhang 2002]. The major challenge of this work is that training and testing must be conducted in the same mode through online real-time interactions between the agent and trainers.

A robot programming system using a virtual-reality graphical simulation system with haptic feedback technique has been introduced by Kahl [Kahl 2002]. Deformable liner objects (OLDs) are used in the research. The method aims at describing the

assembly task in a more natural way without precise coordinates. The programmer performs the assembly task in virtual reality in order to tell the robot roughly in which direction to move. Then, the software analyses this demonstration and generates a sequence of elementary skills, such as “establish contact” or “move to edge”, together with estimated coordinates showing where to execute the skills. The skill sequence produced in this process is executed by a robot.

## *2.4 Conclusion*

In this chapter, the four categories of programming methods developed for robotics manipulators, which include text programming, off-line simulation-based programming, inductive learning and teaching by guiding have been briefly reviewed. A review of the various “teaching-by-showing” techniques, which is the extension of “teaching by guiding”, has been carried out.

# CHAPTER 3 HAPTIC-RENDERED VIRTUAL ENVIRONMENT

## *3.1 Introduction*

The effectiveness of computer simulation can be augmented using Haptic-rendering. A haptic interface, or force feedback device increases the quality of human－computer interaction by accommodating the sense of touch in computer simulation. It provides an attractive augmentation to visual display and significantly enhances the level of immersion in a virtual world. A haptic interface has been effectively used in a number of applications including surgical procedures training, virtual prototyping, control panel operations, hostile work environments and manipulation of materials.

In this work, haptic-rendered virtual modelling is used as part of a new paradigm for programming a robotics manipulator to perform complex constrained motion tasks. The teaching of the manipulation skills to the machine commences by demonstrating those skills in a haptic-rendered virtual environment.

The peg-in-hole assembly process is used as a platform to study the concept. The peg-in-hole insertion problem is often taken as a standard assembly problem, as it concisely represents a constrained motion-force-sensitive manufacturing task with all the attendant issues of jamming, tight clearances, and the need for quick assembly times, reliability, etc.

In the developed system, a human operator demonstrates both good and bad

examples of the desired behaviour in the haptic rendered virtual environment. Position, and contact force and torque data, as well as orientation generated in the virtual environment combined with a priori knowledge about the task, are used to identify and learn the skills in the newly demonstrated tasks and then to reproduce them in the robotics system. The robot evaluates the controller's performance and thus learns the best way to produce that behaviour.

In this chapter, the background to the concept of haptic modelling will be reviewed. Haptic-rendered hardware and software, including PHANToM, GHOST SDK, VRML and ReachIn API will be described. This will be followed by two popular haptic-rendering techniques, namely volume rendering and surface rendering. Penalty-based rendering and proxy-based rendering methods which are based on surface rendering technology have been employed in the project. These methods will be also discussed in this chapter.

## *3.2 Background*

Application of robotic systems in repetitive tasks such as pick & place and assembly requires both online and offline programming. In online programming, a human operator should physically manipulate the robot to perform the task. The actual movements of the robot are recorded and then used as skills for robotic operation. These methods are often time-consuming and risky for either the human operator or the robot. Offline programming usually takes place in the computer environment. This

is achieved by integration of diverse technologies and training of skilled operators. The training process is time consuming and costly while the programming procedures are always tedious and complex.

On the contrary, operating complex robotic systems in a haptic-rendered virtual environment is intuitive and safe. In such an environment, the operator is immersed in a 3-D representation of the world with which s/he can interact. Users do not need to write complicated computer programs. Virtual assembly, for example, provides computerised tools to help in making assembly-related engineering decisions through analysis, predictive models, visualisation, and presentation of data [Jayaram 1997]. Computer aided manufacturing has the benefits of increased productivity, improved quality, and reduced costs. While the environment changes and object movements fed back to the virtual manipulating system, it can make a haptic-rendered virtual environment more effective.

There are three steps in implementing such a step:

(a) Trajectory creation in the virtual space

(b) Simulation of the task

(c) Execution of the task by the real robot

In a haptic-rendered virtual environment, operators can manipulate the virtual robots as well as move the viewpoint in a very simple and intuitive way. By changing the viewpoint, the operator can look at specific regions of the virtual work space.

The other entities of a manufacturing system can be modelled in a virtual

environment in addition to the manipulators. Each element has its own properties (shape, position, orientation, behaviour laws). Using the data related to the virtual objects, it is possible to automatically generate trajectories to manipulate them (grasp, insert, screw etc.). In this way, the operator can be spared from tedious work. The created tasks can be simulated at any time with a collision-detection algorithm. This is essential to avoid possible damage to the physical robot.

The data used by the robot to acquire basic manipulation skills is generated from a haptic-rendered virtual environment. This approach has a number of advantages compared to other methods:

(a) The training data, including force, torque, position, angles and velocities, can be recorded directly from the computer.

(b) This virtual haptic environment can be easily modified according to different manipulation processes and equipment.

(c) The risk of breakdown and breakage of the system is very low.

(d) Dangerous and costly environments can be easily constructed and simulated.

(e) A user-friendly environment for the human operator can be developed.

A primary advantage of Haptic-rendering is that it provides a bidirectional flow of information via position-sensing and force feedback. The coupling of these two types of information flow results in a more natural and intuitive interaction and utilises additional sensory channel bandwidth of the user. When users are presented with a proper combination of visual and haptic information, they experience a sensory

synergy resulting from physiological reinforcement of the displayed multimodal cues [Durlach 1995].

Working with a haptic-rendered virtual environment has other advantages. Firstly, the size of the real robot is not relevant. Irrespective of the size of the robot, the user is brought to the same scale through virtual reality representation. Secondly, establishing a virtual teleoperation in early stages of development and execution of an application prevents the risks associated with conventional direct link teleoperations, which has physical link between the real robot and the manipulator. In a direct link, an unexpected or wrong movement of the operator could result in the breakage of the robotic system. Moreover, the problems associated with the transmission time delays are avoided. The few seconds delay between the operator's action and her/his perception of the impact of action on the real robot can adversely affect the whole process. Finally, the access of the human operator to actual plant is not required during development. This is critical when the disruption of the production line cannot be accommodated.

A haptic-rendered virtual environment is only effective if all the environmental changes and object movements are represented in the virtual manipulation model. This always needs a highly specialised hardware device and high computing speed with visual or multi-sensory feedback. The control loop should be high speed with low delay to ensure the stability of the system. Therefore, haptic-rendering depends closely on the hardware. Hence, a haptic-rendered virtual environment should

maintain a high update rate, present high-quality force feedback, prevent damage being caused by the performance limit and have a user-friendly interface. Selecting both the hardware and software for the haptic-rendered system must be carefully considered. The existing commercial software packages, such as GHOST from Sensable [SensAble (online)] and ReachIn API from ReachIn AB [ReachIn (online)], have already considered such issues. Hence, the system designer can progress forward to the design of the essential core of the haptic algorithm and the graphical parts.

Haptic devices are computer interfaces that allow users to interact with virtual environments through touch and kinaesthesia. By allowing users to touch and manipulate virtual environments, haptic devices are potentially useful computer interfaces for many applications. They can be used for physical skills training, for scientific data understanding, and for entertainment.

## *3.3 Haptic-rendered Virtual Environment*

The haptic-rendered virtual manipulation environment consists of a six degrees of freedom (6 DOFs) haptic device PHANToM Premium 1.5 and its accompanying software GHOST, which is used to construct the virtual manipulation of the peg-in-hole insertion process. In addition, ReachIn API, along with VRML and Python, are also used to construct the virtual haptic manipulation environment of the assembly.

### 3.3.1 PHANToM 1.5

The PHANToM is an advanced, desktop-mounted, general-purpose, haptic interaction device, designed for effective interaction with virtual objects. The PHANToM family of haptic devices, developed at the Massachusetts Institute of Technology, manufactured by SensAble Technologies (Boston, MA), is currently the most widely used force-feedback interface on the market.

Different models in the PHANToM product line meet the varying needs of both research and commercial customers. The PHANToM comes in three models, namely, PHANToM Omni Device, PHANToM Desktop Device and PHANToM Premium Devices, differing in the size of their physical workspace and the number of Degrees of Freedom (DOFs). The PHANToM Premium models are high-precision instruments. They provide the largest workspace and forces within the PHANToM product line. Some devices also offer 6 DOF. The PHANToM Desktop device and PHANToM Omni device offer affordable desktop solutions. Of the two devices, the PHANToM Desktop delivers higher fidelity, stronger forces, and lower friction, while the PHANToM Omni is the most cost-effective haptic device available today.

Depending on the model, PHANToM devices can provide 6 input DOF, 3 or 6 output DOF, and wrist and shoulder motions. As shown in Figure 3.1, the PHANToM has a stylus-shaped handle, providing a precision grasp. The characteristics of the PHANToM make it well suited for point interaction, for example, operated by a single virtual finger, a pencil or a peg. While using this device, sensors are continuously

tracking the position of the stylus. This position is compared against the position of objects in the virtual environment. Based on these comparisons, corresponding output forces are calculated and transferred to the user by three electromechanical actuators. The size of the physical workspace varies between 16 × 13 × 13 cm and 41 × 59 × 84 cm for different models. The device is compatible with standard PC and UNIX workstations. A PHANToM 1.5 can produce a maximum exertable force of up to 8.5 N at nominal position, and a continuous exertable force of 3 N at nominal position. The haptic update rate is 1 kHz. In a 6 DOF device, the maximum torque, generated by the device actuators is 515 mNm [SensAble (online)]. The relatively large physical workspace, in combination with the precision grasp stylus and high DOF for input and output, makes the PHANToM one of the most popular haptic devices on the market.

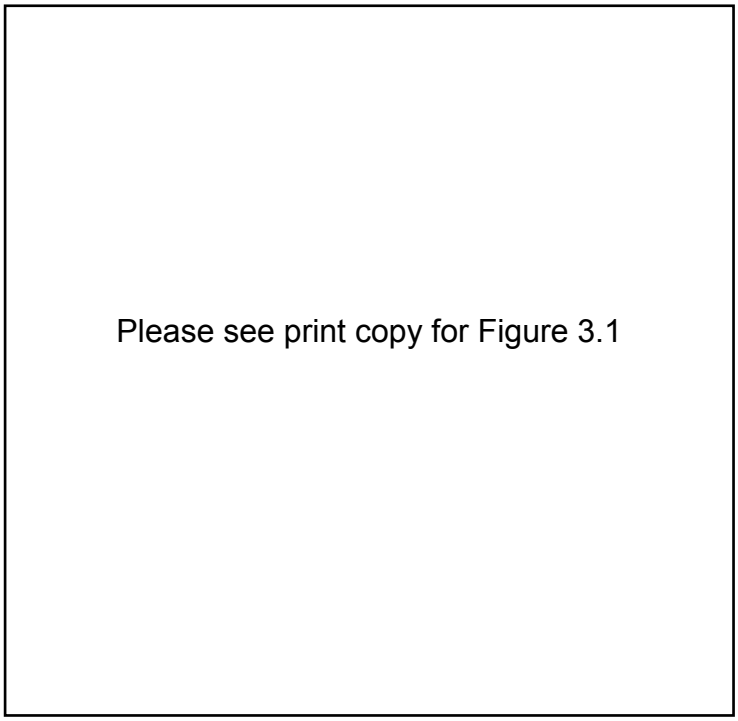



Figure 3.1 PHANToM desktop haptic device [SensAble (online)]

### 3.3.2 GHOST SDK

The GHOST SDK (General Haptic Open Software Toolkit), which accompanies PHANToM, is a powerful, C++ software tool kit that eases the task of developing touch-enabled applications. It has been developed by SensAble Technologies for the PHANToM haptic interface, and consists of a C++ library of objects and methods used for developing interactive, 3D, touch-enabled environment [SensAble (online)].

The GHOST SDK handles the many complex computation required to realistically simulate physical interaction with digital objects and allows developers to specify object geometry, properties, and global haptic effects, using a haptic scene

graph. The GHOST SDK works as the “physics of touch” engine, which takes care of the complex computations and allows developers to deal with simple, high-level objects and physical properties like location, mass, friction, and stiffness. GHOST automatically computes the interaction forces between a haptic point, and objects within the virtual environment. It can also simulate object compliance, friction, springs, impulses and vibrations.

The GHOST SDK provides an abstraction that allows application developers to concentrate on the generation of haptic scenes, manipulation of the properties of the scene and objects within the scene, and control of the resulting effects on or by one or more haptic interaction devices. Developers can use the libraries of 3D prismatic objects, polygonal objects, and touch effects within the GHOST SDK to add a convincingly physical dimension to a variety of applications, including medical simulation, virtual training, geophysics, robotics, teleoperations, assembly path planning, molecular modelling, and nano-manipulation. At the same time, the flexible, extensible architecture of the GHOST SDK makes it a powerful platform for haptics researchers and other developers who need to add new shapes and dynamics, as well as implement the lower-level, direct-force effects. Ghost SDK does not generate graphic scenes. Hence, a different tool such as OpenGL [OpenGL (online)] is needed.

### 3.3.3 ReachIn Desktop Display

The test platform is built on a ReachIn Desktop Display from ReachIn Technologies

[ReachIn (online)]. Figure 3.2 illustrates the ReachIn Desktop Display. The ReachIn Desktop Display is a hand-immersive hardware platform in which scene graph, haptic system and communication system between graphics and haptics are integrated in a consistent, seamless way. The display can provide complex, high-quality haptic feedback through one PHANToM haptic device or cooperation of two PHANToM haptic devices. CrystalEyes shutter glasses from StereoGraphics Corporation [Stereo (online)] are connected to a stereo-output graphics card to provide stereo vision. The screen faces down towards a half-reflecting mirror which reflects the stereoscope monitor image. When working on the three-dimensional workspace on the ReachIn Desktop Display, the user can see the virtual representation of the haptic tool in the same position in which it is placed in the real world. The graphics and the physical interaction device are perceived as being totally co-located within the workspace. In this way, the graphics and haptics are consistent and the virtual environment is seen and felt in the exact same position.

Figure 3.2 ReachIn desktop display

### 3.3.4 ReachIn API

ReachIn API is a scene-graph-based application programming interface for creating multi-sensory, interactive applications. The ReachIn API is based on the structure and standard of VRML [VRML (online)]. It handles the complex calculations required for the touch simulation and the synchronisation with graphic and Haptic-rendering, freeing the user to focus on more important issues such as developing application behaviour or experimenting with haptic algorithms. The ReachIn API is built on C++ but it also integrates Python, VRML, and special Haptic-rendering technology. It is

strictly object-oriented and utilises an advanced object-oriented system.

The virtual reality world is defined by using the scene-graph definition language VRML. Both the standard graphical and structural nodes are from VRML in the ReachIn API. Additional nodes handling the haptics and dynamics of the scene are implemented in C++. A user can also define own nodes implemented in C++. The script language Python [Python (online)] is used in the programming of the objects' behaviour in the virtual reality environment. A special Python script node is provided in ReachIn API.

## *3.4 Haptic-rendering Techniques*

A haptic-rendering technique is a method that calculates the contact forces generated during manipulation in a virtual environment and apply them to the force-feedback device. Most of the rendering techniques used at present are based on a single-point, which means the haptic output forces are calculated at only one point at any given time [Petersik 2002].

At present, surface rendering and volume rendering are the two main haptic-rendering techniques widely used.

### 3.4.1 Volume Rendering

Volume rendering is concerned with the haptic representation of volumetric objects. This technique is developed for accurate representation of object surfaces.

Volumetric objects are usually represented by a three-dimensional array of small volume elements or voxels. Each voxel contains a number of scalar attributes such as colour and density. An appropriate interpolation function is applied to these attributes in order to produce continuous output forces [Avila 1996].

Volume rendering includes three procedures [Lichtenbelt 1998]:

(a) Representing the volumetric object by three-dimensional arrays, namely the RGBA volume data set. The RGBA volume data set is a four-vector data set, where the first three vectors are the R, G, and B colour components respectively, and the last vector, A, represents opacity of the value range between 0 and 1, where 0 means totally transparent and 1 means totally opaque.

(b) Reconstruction of an appropriate interpolation function from this discrete data set. Continuous forces are used as output.

(c) Projecting it onto a 2D output image from the desired point of view.

The main advantage of volume rendering is that internal object structures can be visualised using haptics, so that people can look at the 3D interior information as a whole. Interpretation of the interior is rather difficult during volume rendering. The performance of the volume rendering is significantly computing intensive compared to surface rendering as each frame might take several minutes to be rendered.

### 3.4.2 Surface Rendering

Compared to computational intensity of volume rendering, surface rendering uses surface representations to calculate output forces. Currently, this is the most popular haptic-rendering technique. Objects in the virtual environment are represented by geometric surface polygons; usually triangles. The main advantage of surface rendering is that it can utilise the same object representation as graphical rendering [Petersik 2002]. This haptic-rendering technique is suitable for surface rendering of rigid bodies, such as various geometric objects. The rendering algorithm does not maintain any information about the internal object structure provided by the geometric data representation [Petersik 2002].

The development of surface Haptic-rendering has undergone three main stages. They are the penalty-based-rendering, the god-object-based rendering and the proxy-based-rendering [Mark 1996]. The basic concepts underlying the three approaches are the same. When users try to penetrate the haptic-rendered surface with a haptic probe, a force feedback will push it out of the surface.

#### 3.4.2.1 Penalty-based Rendering

The penalty-based rendering method is a primitive method among the three haptic-rendering technologies. This method is inspired by the fact that when two geometrically rigid objects collide small deformations take place at the contact surface and these deformations can be modelled with springs.

Consider two colliding objects A and B. When penalty-based collision occurs, the following parameters, namely contact point *p*, normal *n* perpendicular to the surface across the point *p* and a penetration depth *d*, can be measured. The penalty-based spring force and torque applied to object A are defined as follows:

$$F_A = -k \cdot f(d) \times n \tag{3.1}$$

$$T_A = D(p - c_A) \times F_A \tag{3.2}$$

The spring force $F_A$ is calculated as the product of the force function *f* of distance vector *d* and the stiffness constant *k*. Opposite forces and torques are applied to object B. The force function *f* can be linear or a complicated non-linear function. In these relationships, $c_A$ is the centre of mass of A and D is the distance vector between *p* and $c_A$.

The penalty-based rendering method has several attractive properties [McNeely 1999] [KiM 2003] [Johnson 2003]:

(a) The force model, based on objects interpenetration, used to compute each contact point, is simple to construct.

(b) Object contact decision is made in every simulation frame, which makes the penalty-based rendering method best suited for interactive applications with fixed time steps, such as haptic rendering.

However, several severe problems prevent penalty-based rendering from becoming the primary rendering method [Wu 2000] [Larsen 2001]:

i. As the tip of the haptic probe penetrates into the object surface, internal

forces are generated to halt the tip's advance through the surface. However, a deep penetration does not produce sufficient reactive forces as the internal volume of the object is not modelled.

ii. Moreover, the tip of the haptic probe penetrating one side of the object's surface is often close to the other side of the surface. As a result, the generated force would eventually push the point off the surface.

iii. A severe problem is that several points in an object may have the same distance to the surface. In these situations, forces generated from the haptic environment become unstable, since force directions may start to change rapidly. This results in haptic probe oscillation. Since force fields should be computed in advance, this method is not appropriate in dynamic environments.

#### 3.4.2.2 God-object Rendering

God-object rendering technology was first employed for haptic applications to address the limitations of penalty-based rendering by Zilles and Salisbury [Zilles 1994]. This method employs a strategy to keep track of a virtual contact point, namely the god object, which remains on the surface during haptic interaction so that it can prevent the virtual contact point of the haptic interface from penetrating the object. The force direction will be accurately applied to the operator, while the position of the god object on the surface is determined.

Although the actual positions of objects may overlap in the virtual environment, the generated interaction forces should be based on a contact state where the objects cannot overlap. Given the previous location of the god object and the current location of the haptic interface, the algorithm will identify a number of surfaces on the rendered object which are currently involved in the interaction and denote them as active. A surface is active if the god object is on one side of the rendered surface, and the haptic interface is on the other, and the action takes place within the boundaries of the surface. Once this set of surfaces, or constraints, has been identified the new location of the god object can be computed. By finding the closest point on the active constraint surface to the current haptic interface point, the new location of the god object can be determined. By choosing planar constraints, the solution can be found by solving a set of linear equations.

The god-object rendering algorithm technique has successfully solved the problems raised by the penalty-based haptic-rendering technique, because in this case, the contact is between objects instead of one point and one object [Salisbury 1995]. The god object tries to be as close to the haptic probe tip as possible and any displacement of the god object relative to the tip gives rise to a haptic force. The god object is restrained by a predefined topology calculated from the objects in the virtual environment. Hence, when the tip penetrates into a surface, the god object is left outside. This will generate a force pulling the instrument out of the surface. The god-object can slide on the surfaces of polygons when the area is defined as legal in the

topology of the objects. Since the god object is always located on a surface, it can not penetrate through thin objects. However, the god object could slip through very small gaps between adjacent polygons, hence the predefined topology is used to restrain the god object instead of the polygons.

#### 3.4.2.3 Proxy-based Rendering

The proxy-based rendering method, which performs the same function as god-object rendering, was first described in [Ruspini 1997]. It is an extension of the god-object rendering technology that employs virtual proxy as a representative object to substitute the phantom probe in the haptic-rendered virtual environment. By using an intermediate virtual object, namely proxy, with a defined size instead of the point-shaped god object, the problem of slipping through gaps between polygons is solved. Because the virtual proxy has a finite size, it does not slip through the tiny numerical gaps found in most polygonal meshes [Zilles 1994]. The proxy is restrained by polygons which makes a dynamic environment possible. This is different from god-object rendering in which a predefined topology is calculated from the objects in the virtual environment. With the proxy method, the proxy remains on the object surface when the haptic probe tip penetrates the object surface. Rather than calculating the force applied to the device directly from the virtual object, the haptic probe is controlled to move towards the proxy position. The proxy itself can be controlled to move over the surface of the object with regard to the location of the haptic probe.

Figure 3.3 illustrates the motion of the virtual proxy. In the absence of an obstacle, if the virtual proxy's path does not collide with any obstacles, the virtual proxy moves directly towards the object. The proxy's position is advanced until it makes contact with the first obstacle in its path. When the proxy encounters one or more interfering primitives, direct motion becomes impossible. The user can still reduce the distance of the proxy relative to the goal by moving the proxy along one or more of the constraint object surfaces. The motion is chosen to locally minimise the distance relative to the goal. The proxy stops when it is unable to decrease its distance relative to the goal, due to jamming.

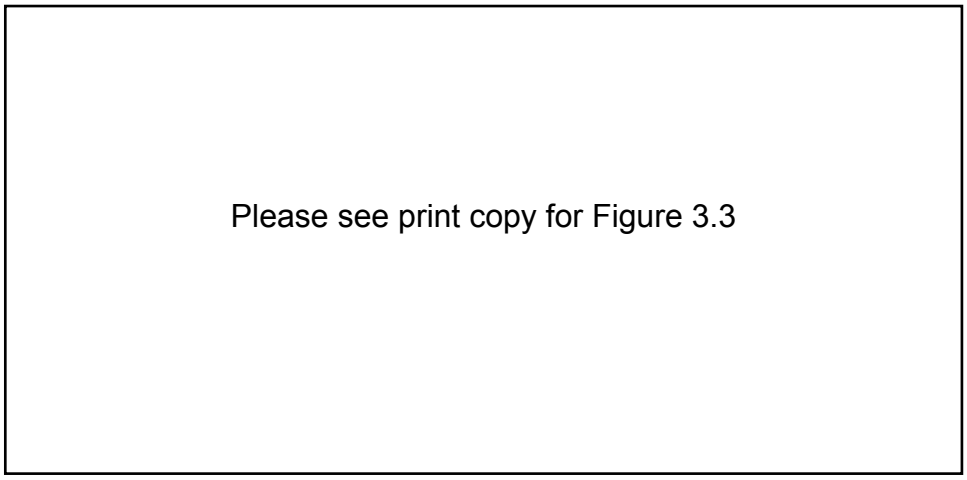



Figure 3.3 Motion of the virtual proxy [Ruspini 1997]

To model the interaction between virtual proxy and obstacle, a configuration space of the proxy (a constraint plane), namely *configuration space obstacles* (C-obstacles), which consists of all the points within one proxy radius of the original obstacle's surface, is defined (Figure 3.4). In this configuration space, the position of

the proxy is identified as a point while all C-obstacles have continuously defined surfaces and nonzero thickness [Ruspini 1997]. The periphery of the C-obstacles, namely the constraint plane, can then be formed.

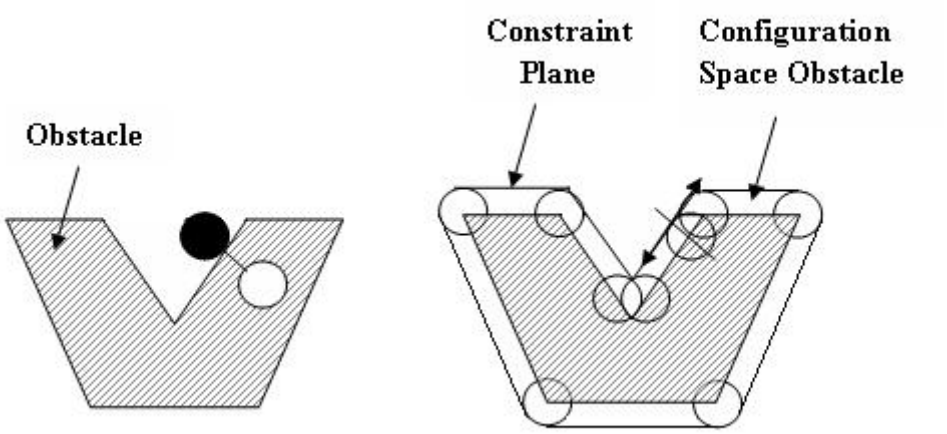



Figure 3.4 Configuration space of the proxy

The difference vector between the proxy and the device position is used to calculate a proportional output force. In order to update the proxy position while the device moves within the object, the distance between the device and the proxy position, $\|x - p\|$, is locally minimised. This will be subject to:

$$\begin{aligned} &\hat{n}_1^T x \geq 0, \\ &\hat{n}_2^T x \geq 0, \\ &\vdots \\ &\hat{n}_m^T x \geq 0, \end{aligned} \tag{3.3}$$

Where p represents the vector from the current proxy position to the user's

position, x represents the new sub-goal vector, and $\hat{n}_i, 0 \le i \le m$, are the unit normal of the constraint planes [Ruspini 1997].

Constraint-based rendering methods eliminate the problem of unstable haptic output. Since the output forces are calculated in real-time, these methods are also suited for dynamic environments. Because of the real-time rendering, god-object and proxy rendering methods are more computationally expensive than penalty-based rendering methods.

## *3.5 Force feedback in Haptic-rendered Environment*

### 3.5.1 Spring-damper System

In this configuration, the virtual peg is coupled with the PHANToM Premium Devices (that is the manipulation point) through a spring-damper system [Mathinsite (online)]. An ideal spring-damper system is depicted in Figure 3.5. It is composed of a mass attached to a spring and a damper. By applying Newton's second law and analysing forces applied on the mass (free body), the following relationship can be obtained [Mathinsite (online)]:

$$F_s = -kx \tag{3.3}$$

$$F_d = -Bv = -B\dot{x} = -b\frac{dx}{dt} \tag{3.4}$$

$$\sum F = ma = m\ddot{x} = m\frac{d^2x}{dt^2} \tag{3.5}$$

where

- *k, B* represent spring constant and damper constant, respectively

- $m$ is the mass of the spring-system
- $Fs$, $Fd$ and $\Sigma F$ represent forces applied on the spring, damper and mass respectively
- $a$ or $\ddot{x}$ is the acceleration of the mass
- $\dot{x}$ is the velocity of the mass
- $x$ is the displacement of the mass relative to a fixed point of reference.

Combining the three motion equations, a differential equation for displacement $x$ as function of time $t$ is obtained:

$$\ddot{x} + 2\xi\omega_0 \dot{x} + \omega_0^2 x = 0 \qquad (3.6)$$

where damping factor $\xi = \frac{B}{2\sqrt{km}}$, frequency $\omega_0 = \sqrt{\frac{k}{m}}$.

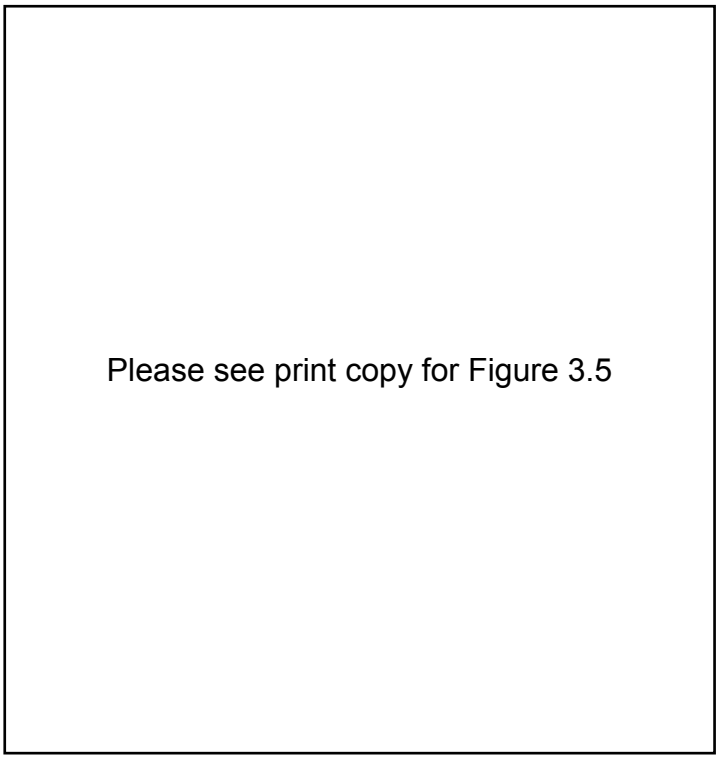



Figure 3.5 Mass-Spring-Damper System [Mathinsite (online)]

The peg is a dynamic rigid object in the virtual environment. The forces reacting on the peg are transferred to PHANToM probe through the spring-damper system. The virtual hole is static in the virtual environment while the peg can be moved and rotated (Figure 3.6) [Chen 2002a]. Forces applied on the PHANToM probe through the spring- damper system is described as followed:

$$F = kx + B\dot{x} + m\ddot{x} \tag{3.7}$$

where the x, $\dot{x}$, and $\ddot{x}$ represent the displacement, velocity, and acceleration of the peg respectively [Chen 2005].

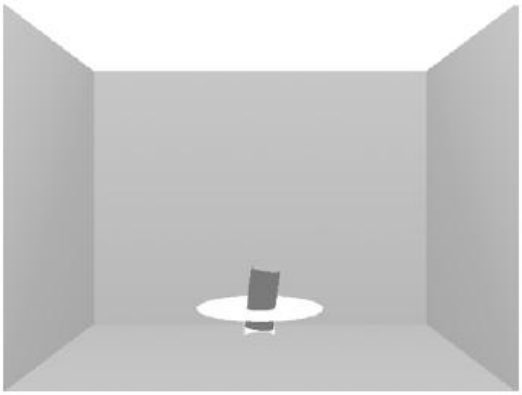


Figure 3.6 3DOF peg-in-hole insertion virtual environment

### 3.5.2 Force Feedback from Haptic-rendered Environment

The force generated at each point is the sum of the normal force and the friction force exerted at that point, as shown in Figure 3.7 [Chen 2002b].

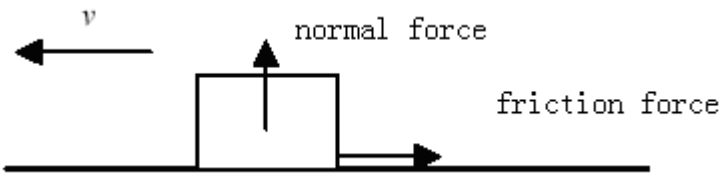



Figure 3.7 Normal force and friction force

The direction of the normal force is perpendicular to the contact surface and points to the moving object. The magnitude of the normal force generated at each point is calculated by

$$fc = k \cdot d + c \cdot ad + b \cdot v \tag{3.8}$$

where

- $d$ is the depth of the point in the contacting static object
- $ad$ is the accumulated depth during a continuous contact between the point and the static object
- $v$ is the velocity of the object and is calculated by the current Depth minus the last Depth divided by the sampling time
- $k$ is the stiffness coefficient
- $c$ is the coefficient for the accumulated depth.
- $b$ is the damping coefficient

The torque generated at each point is calculated by

$$t = fc \times D \tag{3.9}$$

where

$D$ is the distance from the contact point to the rotating centre of the object.

The direction of the friction force is along the contact surface and opposite to the moving direction. The magnitude of the friction force generated at each point is calculated by

$$f = \sigma \times z \tag{3.10}$$

where

- $z$ is the strain describing micro-movements between the two objects, which is not allowed to exceed a small value called the breakaway distance $z_{max}$.
- $\sigma$ is the stiffness relating force to strain, assuming $x_i$ is a point fixed on the moving object, and $y_i$ is an adhesion point on the static object, as shown in

Figure 3.8 [Chen 2002b].

The following relationship is used to calculate $z_i$ by

$$z_i = x_i - y_i \tag{3.11}$$

$$y_i = \begin{cases} x_i \mp z_{\max}, & if \quad |x_i - y_{i-1}| > z_{\max} \\ y_{i-1}, & otherwise \end{cases} \tag{3.12}$$

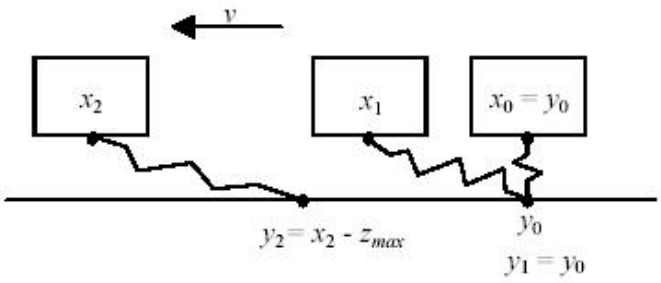



Figure 3.8 Definition of the strain $z$

## *3.6 Graphic-modelling Techniques*

Graphic-modelling provides vision feedback to the operator that bridges haptic-rendering model and virtual manipulation. The graphic models are realistic and can be efficiently constructed. Some of the graphic-modelling techniques can also provide inspiration for haptic-rendering techniques.

The Non-Uniform Rational B-Splines (NURBS) techniques, typically used in Computer Aided Design (CAD) environments, have also been widely deployed in computer graphic displays and vision or touch models acquired by robotic systems [Han 1996] [Chang 1997] [Ikits 2001] [Balasubramaniam 2002]. Using Non-Uniform Rational B-Splines (NURBS) technique, very complicated object can be modelled and modified locally. However, the present of the object requires a large mount of data.

The collision detection is also difficult to perform [Piegl1995].

In the first implementation of the virtual haptic-rendered peg-in-hole insertion project, OpenGL was used to build the peg-in-hole graphic model and GHOST SDK for the haptic model. The GHOST SDK (General Haptic Open Software Toolkit), which accompanies PHANToM, developed by SensAble Technologies for the PHANToM haptic interface, was used for developing interactive, 3D, and touch-enabled environment [SensAble (online)]. The penalty-based rendering, developed through GHOST SDK, cooperates with OpenGL to render both visible graphic objects and haptic objects. VRML was employed in the final implementation of the graphic model, while ReachIn API was used to develop the haptic-rendering [ReachIn (online)]. In this case, the objects, created by proxy-based rendering method, developed through ReachIn API can be seen through the VRML model.

Both OpenGL and VRML work based on the same concept that surface of the objects are made of quadrangle polygons. The visibility of the surface depends on the direction of the normal perpendicular to the surface. The order of the quadrangle vertexes of the surface and the direction of the normal must follow the right hand rule. If the normal faces the viewer, the surface is visible, otherwise it is invisible.

## *3.7 Haptic-rendered Peg-in-hole Environment*

The peg-in-hole assembly process is used as a platform to study the concept. The peg-in-hole insertion problem is often taken as a standard assembly problem, as it

concisely represents a constrained motion-force-sensitive manufacturing task with all the attendant issues of jamming, tight clearances, and the need for quick assembly times, reliability, etc.

In the developed system, a human operator demonstrates both good and bad examples of the desired behaviour in the haptic virtual environment. Position and contact force and torque data, as well as orientation generated in the virtual environment combined with a priori knowledge about the task, are used to identify and learn the skills in the newly demonstrated tasks and then to reproduce them in a robotic system. The robot evaluates the controller's performance and thus learns the best way to produce that behaviour.

### 3.7.1 Penalty-based Peg-in-hole Rendering

#### 3.7.1.1 Haptic-rendering for 3 DOF Device

In the previous research, the haptic-rendered virtual peg-in-hole model was developed for a 3 DOF haptic device by Yuxin Chen [Chen 2005]. The haptic-rendered peg-in-hole insertion model was constructed based on penalty-based haptic-rendering method. This haptic-rendered model, which generates force data, is constructed using the TriPolyMesh and PointShell methods, developed by the GHOST SDK supplied with PHANToM haptic device [Chen 2005].

The TriPolyMesh (triangle polygon mesh) method was used to construct the haptic model of the peg and hole. The surfaces of the virtual peg and hole were

formed by rotating the triangles around the coordinate centre (Figure 3.9) [Chen 2002a].

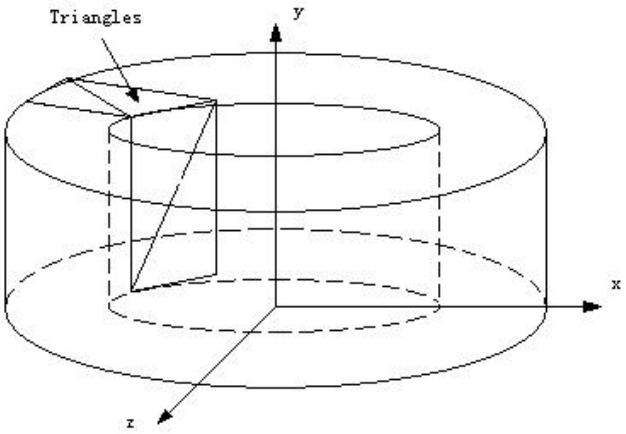



Figure 3.9 TriPolyMesh method

The PointShell method was employed to generate force data from the haptic-rendered model. In the PointShell method, an object is represented as a collection of a group of important points on their surface. The point *P(x, y, z*) of the curve *C(x, y, z)* in the PointShell model must be singular. The following formula must be applied:

$$C(x, y, z) = 0 \tag{3.13}$$

$$\frac{\partial C}{\partial x} = \frac{\partial C}{\partial y} = \frac{\partial C}{\partial z} = 0 \tag{3.14}$$

where *Cx, Cy, Cz* are the partial derivatives of function *C* relative to *x, y, z* [Chen 2005].

A surface normal vector pointing inwards is assigned to every point on the PointShell to provide the normal force direction [Renz 2001]. Figure 3.10(a)

illustrates the normal vectors of a PointShell. In the PointShell developed for the peg-in-hole insertion, the directions of the vectors assigned to singular points are not pre-determined, as they depend on the normal of the contact surface (Figure 3.10(b)) [Chen 2002a]. The directions are assigned when the peg and hole are in contact.

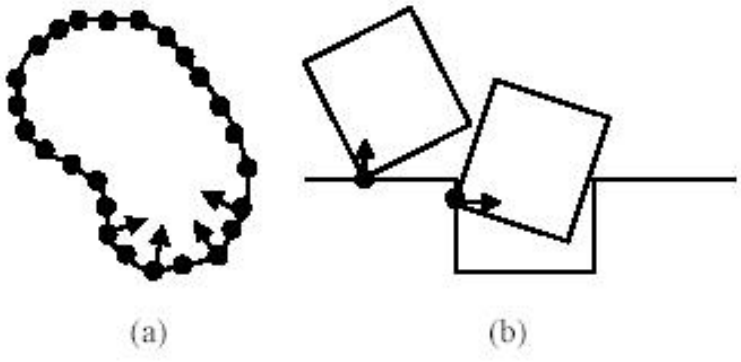



Figure 3.10 PointShell method

The developed haptic-rendered model proved quite stable when the peg-in-hole insertion was performed by the three degrees of freedom (3 DOFs) PHANToM Premium 1.0 [Chen 2005]. However, when at a later stage, the haptic device was upgraded to a six degrees of freedom (6 DOFs) device, PHANToM Premium 1.5, strong oscillation occurred during the virtual peg-in-hole insertion and consequently the simulation could not be carried out successfully. Force and torque data collected from one unsuccessful peg-in-hole insertion is depicted in Figure 3.11.

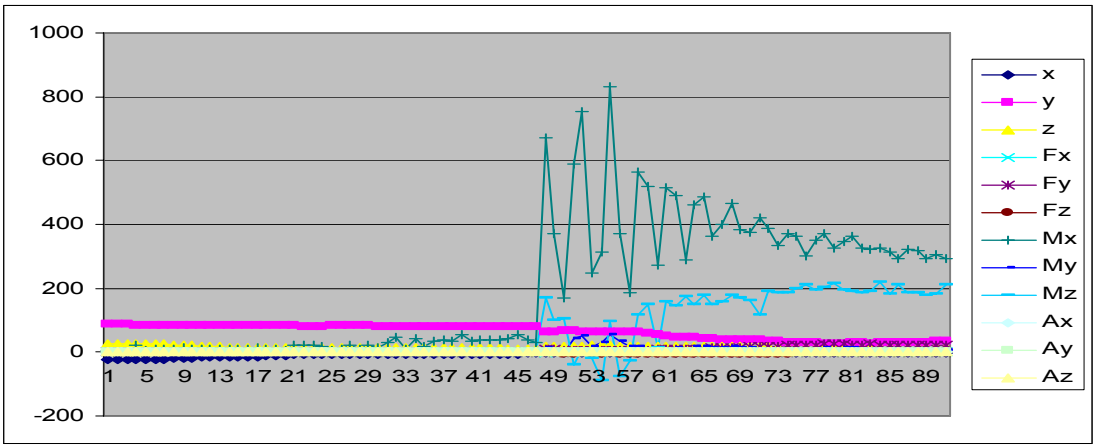



Figure 3.11 Strong oscillation indicated by high torques

Further investigation of the problem revealed that the PointShell and TriPolyMesh algorithms used in the model were not sufficiently accurate for operation with a 6 DOF haptic device.

### 3.7.1.2 Haptic-rendered Model for 6 DOF Device

In order to stabilise the virtual peg-in-hole insertion with tight fit for a 6 DOF haptic device, three new algorithms have been developed and applied to the physical model of the process. They include a modified PointShell algorithm, modified TriPolyMesh algorithm and dual-gstCylinder algorithm.

In the modified TriPolyMesh algorithm, the haptic hole is constructed by a triangle polygon mesh algorithm and the haptic peg is a gstCylinder, which is a cylinder-shape class defined by GHOST SDK, representing a geometric primitive cylinder. The inside, outside and top surfaces of the hole are formed by rotating triangle polygons around the y-axis as shown in Figure 3.12. The gstPoints, one of a

variety of data types defined and used in the GHOST API, are added to the vertex of each triangle polygon, representing a Cartesian three-dimensional point class. A group of gstPoints is also added to the end edges of the peg. The gstPoint generates contact force and torque data when it intersects with the body of the peg. Similar to the previous model, the direction of the vectors assigned to each gstPoint is not predetermined in advance, as it depends on the normal of the contact surface. This is determined according to the contact position when the peg and the hole come in contact. The gstPoints also play an important role in approximately removing the gaps produced when a polygon is used instead of a circle, as illustrated in Figure 3.13.

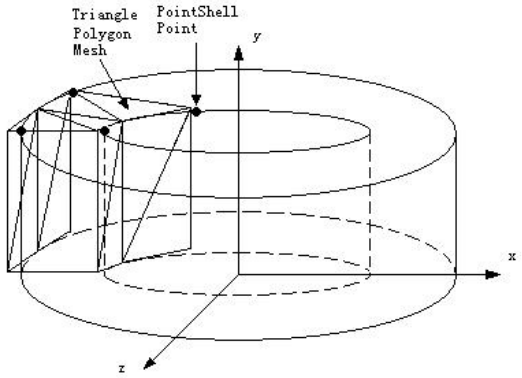



Figure 3.12 Modified TriPolyMesh method (a)

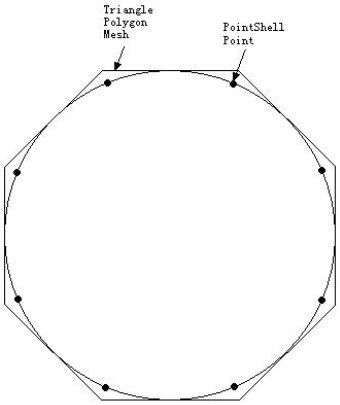



Figure 3.13 Modified TriPolyMesh method (b)

In the dual-gstCylinder algorithm, the hole is constructed using two gstCylinders rather than triangle polygons, forming the inner and outer surfaces of the hole (Figure 3.14). This approach is simple, and the constructed model conforms well to the shape of the hole and hence there is no inaccuracy in the model. The approach also offers a simple technique for the construction of the haptic-rendered model. Since the haptic-rendered model is a real cylinder model, no accuracy problem is introduced according to the approximate cylinder created by the triangle polygon mesh algorithm. The gstPoints are defined for the edge of the hole and the two ends of the peg.

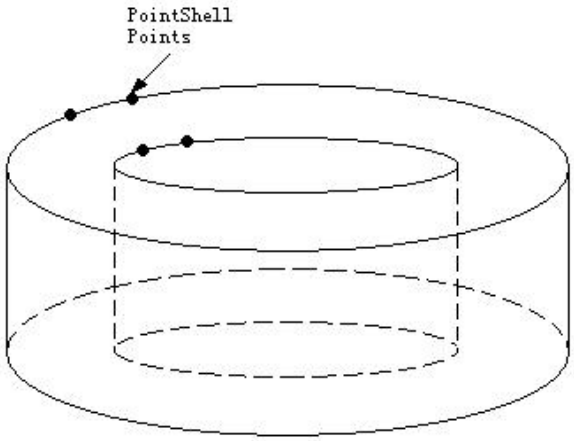



Figure 3.14 Dual-gstCylinder method

In the PointShell algorithm, the hole is defined as a dynamic object created by a group of gstPoints, as shown in Figure 3.15. The peg is also constructed by the gstCylinder. The direction of the force generated at each gstPoint is normal to the surface of the hole at each point. The gstPoints on the hole prevents the virtual peg from penetrating into the inner surface of the hole.

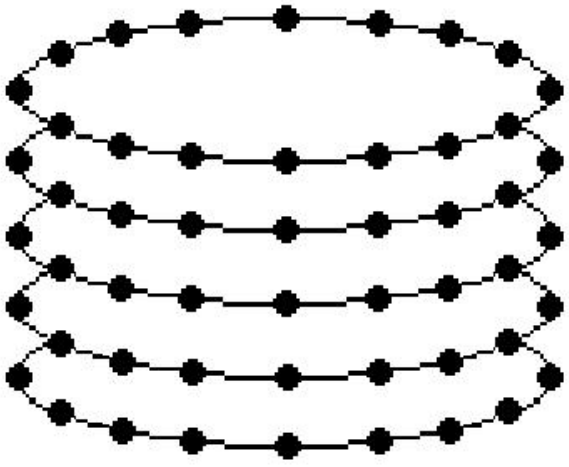


Figure 3.15 PointShell method

The developed algorithms were applied to the virtual peg-in-hole insertion process. The process proved stable, and no jamming was observed when the assembly was performed in the virtual environment with a 6 DOF PHANToM Premium 1.5.

Some of the experimental results are illustrated in Figure 3.16 – 3.18. The variation of 11 normalised series of *x, y, z, fx, fy, fz, Mx, My, Mz, Ax, Ay* and *Az* are illustrated in these diagrams, where:

- *x, y, z* are positions of PHANToM probe or peg in the world coordinates [millimetres].
- *fx, fy, fz* are reaction forces in the world reference frame from PHANToM [Newtons],
- *Mx, My, Mz* are reaction torques in the world reference frame from PHANToM [Newton*millimetres].
- *Ax, Ay, Az* are equivalent rotations of the current rotation matrix (orientation) based on successive rotations around the x, y, z axes. Angles are in radians and a right-hand rule is used.

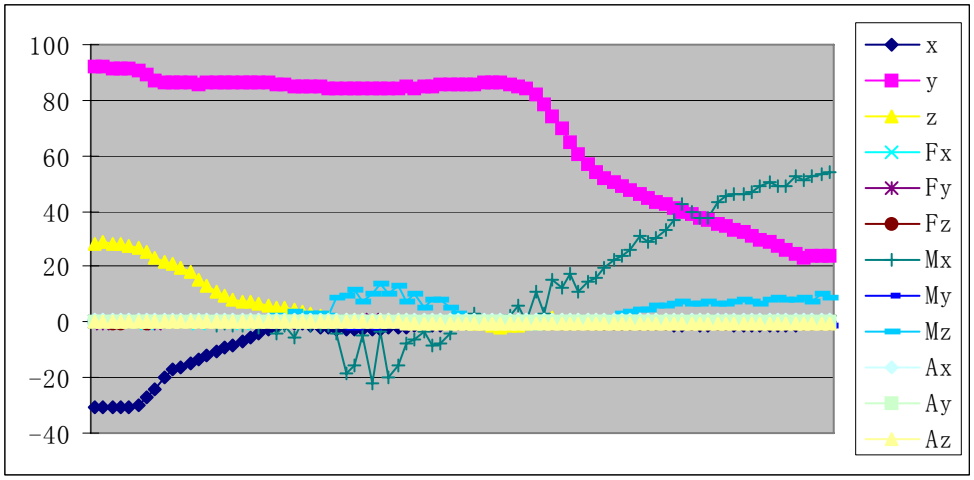



Figure 3.16 Results of using Modified TriPolyMesh method

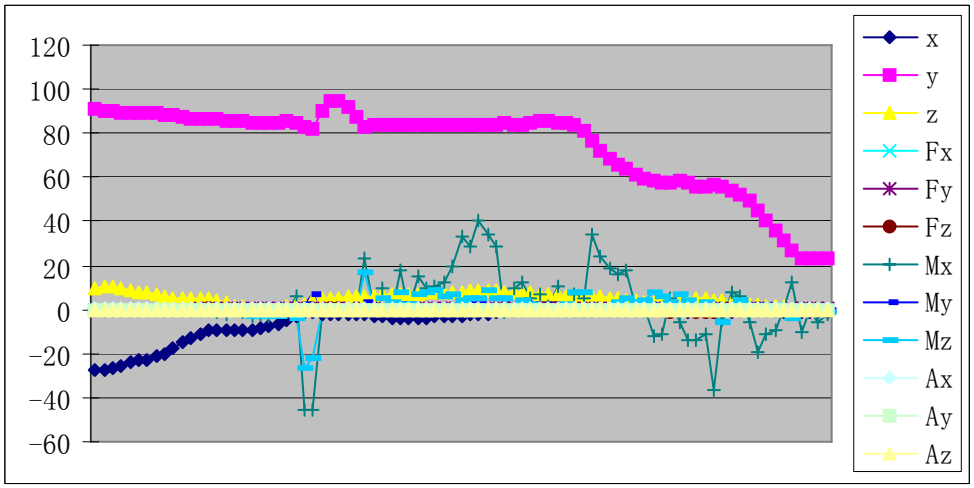



Figure 3.17 Results of using Dual-gstCylinder method

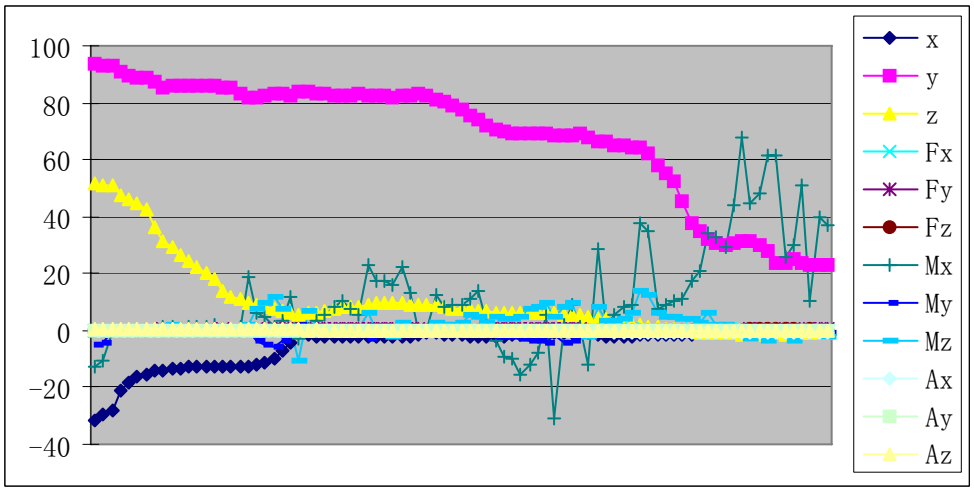



Figure 3.18 Results of using PointShell method

### 3.7.2 Application of Proxy-based method to Peg-in-hole Insertion

Figure 3.19 shows the developed virtual environment using proxy-based rendering process for the peg-in-hole insertion process. The peg is a dynamic rigid object in the haptic-rendered virtual environment. The force and torque reacting to the peg are transferred to PHANToM Premium 1.5 through the spring-damper system. The hole is static in the environment, while the peg can be translated and rotated.

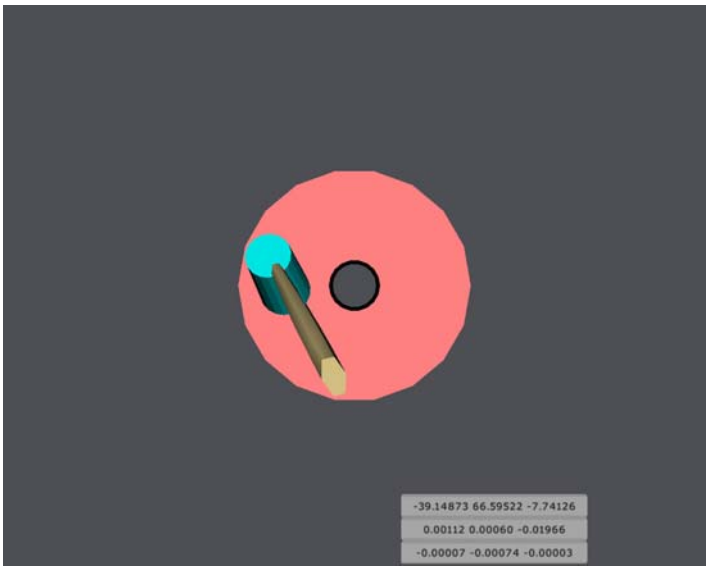



Figure 3.19 6 DOF Peg-in-hole haptic virtual environment

The haptic-rendered model of the peg-in-hole assembly insertion generating force and torque data is constructed using the virtual proxy method [Ruspini 1997]. The virtual rigid peg is defined as a virtual proxy and controlled by the physical ReachIn probe in the haptic-rendered virtual environment. The position of the virtual proxy is changed according to alteration in the probe's position.

The surfaces of the peg and the hole are constructed using polygons. This results

in numerical errors, which can produce gaps in the common edge of the peg and hole. The size of the virtual proxy is chosen large enough to prevent it from falling into the gaps.

The force generated at the proxy is the sum of the normal and the friction forces. The generated torque is the product of the contact force vector applied at the contact point and the distance vector from the contact point to the rotating centre of the object [McNeely 1999; Renz 2001]. The full rotation of the proxy is recorded as $(f_x, f_y, f_z, \theta)$. This describes an arbitrary rotation about an axis vector, where $(f_x, f_y, f_z)$ are the axis vectors and $\theta$ is the angle in radians in the right-handed direction. The axis vector is of unit length. The rotation matrix is calculated by:

$$Rot\ (f, \theta) = \begin{bmatrix} f_x f_x v\theta + c\theta & f_y f_x v\theta - f_z s\theta & f_z f_x v\theta + f_y s\theta & 0 \\ f_x f_y v\theta + f_z s\theta & f_y f_y v\theta + c\theta & f_z f_y v\theta - f_x s\theta & 0 \\ f_x f_z v\theta + f_z s\theta & f_y f_z v\theta + f_x s\theta & f_z f_z v\theta + c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $v\theta = 1 - \cos\theta$, $c\theta = \cos\theta$, $s\theta = \sin\theta$ [Niku 2001]

Figure 3.20 shows the position, force and torque data as well as the change in the rotation angle from the last step to the current, obtained from the haptic-rendered virtual environment.

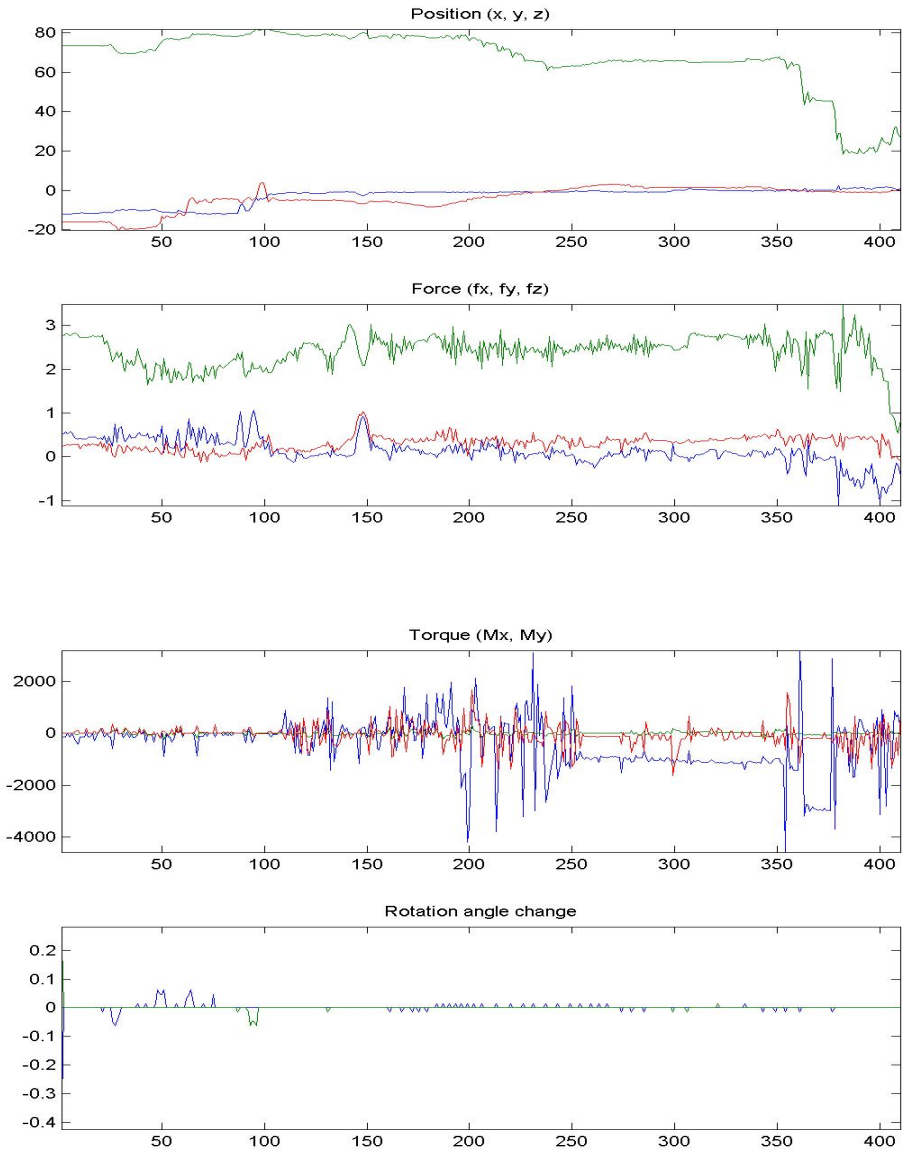



Figure 3.20 Training data obtained from the virtual environment

## *3.8 Summary*

In this chapter, the backgrounds of haptic concept and two popular Haptic-rendering technologies have been reviewed. The haptic-rendered virtual environment, including PHANToM, GHOST SDK, VRML and ReachIn API, has been described. Penalty-based rendering and proxy-based rendering methods, based on surface rendering

technology, employed in the peg-in-hole insertion procedure, as case studies of the haptic-rendered virtual environment, have been studied. The algorithms used to calculate the collision force and torque of peg and hole have been described. The generated force and torque data from the haptic-rendered virtual environment have also been presented.

# CHAPTER 4 SKILL ACQUISITION

## 4.1 Introduction

The paradigm proposed to acquire manipulation skills from a haptic-rendered virtual manipulation is introduced and studied in this chapter. The deployed skill acquisition methods are primarily based on behavioural cloning methods. Manipulation states are generalized offline by peg and hole's physical contact relations or statistic methods, such as Fuzzy Clustering algorithms, according to the characteristics of the generated manipulation data. Hidden Markov Model is used to estimate the next optimal state, in another word the optimal state sequence. These approaches are described in detail and their main characteristics are highlighted.

## 4.2 Background

Traditional control theory uses a mathematical model of a physical process to predict its behaviour and adopt appropriate control actions. Unfortunately, either many processes are too complicated to be accurately modelled or there is insufficient information available about the process environment.

Heuristic methods, such as artificial neural networks, genetic algorithms, fuzzy control, expert systems and reinforcement learning have been developed to replicate the human ability to control and monitor a process without a need to mathematically model it. These methods, contrary to the ability of a human operator, have their own

strengths and limitations. Each method is more appropriate for a particular application and might prove ineffective for another.

Moreover, controlling a complex dynamic system, such as a robot, a plane or a crane, usually requires a skilled operator with full understanding of its operation. Some machine learning methods, such as artificial neural networks, genetic algorithms and reinforcement learning, do not use prior knowledge about the system to be controlled. This results in a low successful learning rate, low robustness, a considerable amount of time-consuming experimentations with the dynamic system, and difficult interpretation and comprehensibility of the system. However, a human operator usually learns initial strategies from their prior knowledge of the system or from demonstrations by experienced operators.

Donald Michie [Michie 1993] introduced the concept of behavioural cloning, aiming at deploying the skills of an expert operator to generate a generic automatic control algorithm using heuristic and machine learning techniques. This method was originally motivated by the difficulties encountered in getting expert controllers to produce detailed explanations of their skills. A skilled operator's control traces are used as examples for machine learning algorithms to reconstruct the control strategy that the operator executes subconsciously. In general, there are two goals in behavioural cloning. One is to generate "good performance" clones, which can reliably carry out the control task. Another is to generate "meaningful" clones, which can help to achieve a better understanding of the human operator's subconscious skill

[Urbancic 1994]. Behavioural cloning has been successfully used in a number of domains. These include pole balancing, production-line scheduling [Kerr 1994], piloting [Bain 1999; Sammut 1992], and operating cranes [Suc 2000]. These experiments are reviewed by Bratko [Bratko 1998].

Decision trees or regression trees are often used in inducing behaviour cloning controllers. Such clones do provide some insight into the control strategy. However, in general, they lack the conceptual structure that would clearly reflect the causal relations in the domain and the goal structure of the control strategy [Suc 1998]. Clones in the form of decision trees do not explicitly show any time ordering between events in the controlled system and the actions taken. Although successful clones have been induced in the form of decision trees or regression trees, such as C4.5 [Esmaili 1995; Pearce 1999] and locally weighted regression [Suc 1998], the following problems have generally been observed with this approach [Suc 1998]:

(a) Behaviour cloning controllers are not stable with regard to small changes in the control task.

(b) The proportion of successful controllers induced is low, typically below 50%.

(c) Induced clone controllers lack the typical elements of human control strategies such as goals, subgoals, phases and causality, and hence cannot adequately generalise the human skill.

The logic-based machine learning method uses the background knowledge, which

is known prior to learning. For example, Inductive Logic Programming (ILP) [Muggleton 1992] is a kind of logical-based machine learning. Some experiments with ILP were performed in the control task modeling a pilot flying a F-16 flight simulation on a leveled left turn [Camacho 1995], but the results were generally not better than those using decision or regression trees.

Some behavioural cloning approaches employed from traditional control theory have also proved effective. In Suc [Suc 1997], cloning controllers take the form of Linear Quadratic (LQ) controllers with subgoals, where in the subgoals are automatically induced from the operator's control traces. The reconstruction of a human operator's skill exploits some elements of the theory of Linear Quadratic Regulator problems [Bertsekas 1987]. Both the dynamics of the system and the example behavioural traces are considered in the learning process. The system model control skills suggested by Isaac [Isaac 2003] are separated into a reactive level and an anticipatory level, that is, the learning of traditional PID (Proportional Integral Derivative) controllers as rule sets and combining them into a goal-directed hierarchical framework. These approaches have significantly improved both the clones' robustness in regards to changes in the control task, and the yield of the cloning process. However, this approach still has difficulties in domains with significant nonlinearities.

## *4.3 Perception Module*

The basic skills are derived and structured by the perception module. It derives the basic skills from the training data produced through manipulation in the virtual environment and stores them in models which can generate outputs according to inputs. These models can also be viewed as databases that store the skills. The skills stored in the database or in the model are used by the planning module to control the manipulator. Skills in the database can be further augmented by the skills learned online during physical manipulation by the manipulator.

The human performs manipulations by choosing from a limited but possibly large repertoire of movement primitives or basic skills. A manipulation task usually consists of a sequence of basic skills. Identification of these basic skills and mapping them on to an equivalent series of robot manipulation primitives form the core of an algorithm for skill acquisition and transfer of those skills from the human to a robotic manipulator. Such skill-based manipulation is an effective way for a robotic manipulator to execute a complex task.

The basic skills are defined according to the contact-state transition of a task, independent from the configuration of a manipulator [Nakamura 1996]. In a virtual manipulation environment, the basic skills can be also identified by the contact states and state changes [Onda 1995; Takamatsu 1999]. Using this approach, the basic skills can be automatically extracted from the manipulation carried out in the virtual environment.

The structure of the Perception Module of this project is illustrated in Figure 4.1. In the first step, the training data, generated from the haptic-rendered virtual environment, are refined by removing noise data and compressing highly correlated vectors. In the second step, manipulation states are generalized either by peg and hole's physical contact relations or by some Fuzzy Clustering algorithms. In the third step, Hidden Markov Model is applied to estimate the optimal state sequence. The physical peg-in-hole assembly can start from any predefined manipulation state. Optimal state sequence can then be used to estimate the next optimal state to follow. The Locally Weighted Regression (LWR) method is encoded as the approximator for the trajectories in each state during physical manipulation. The initial locally weighted regression learning modules are developed by the online training data generated from the haptic-rendered virtual environment. These locally weighted regression learning modules will be also improved in real-time by the data generated from the physical peg-in-hole assembly procedure.

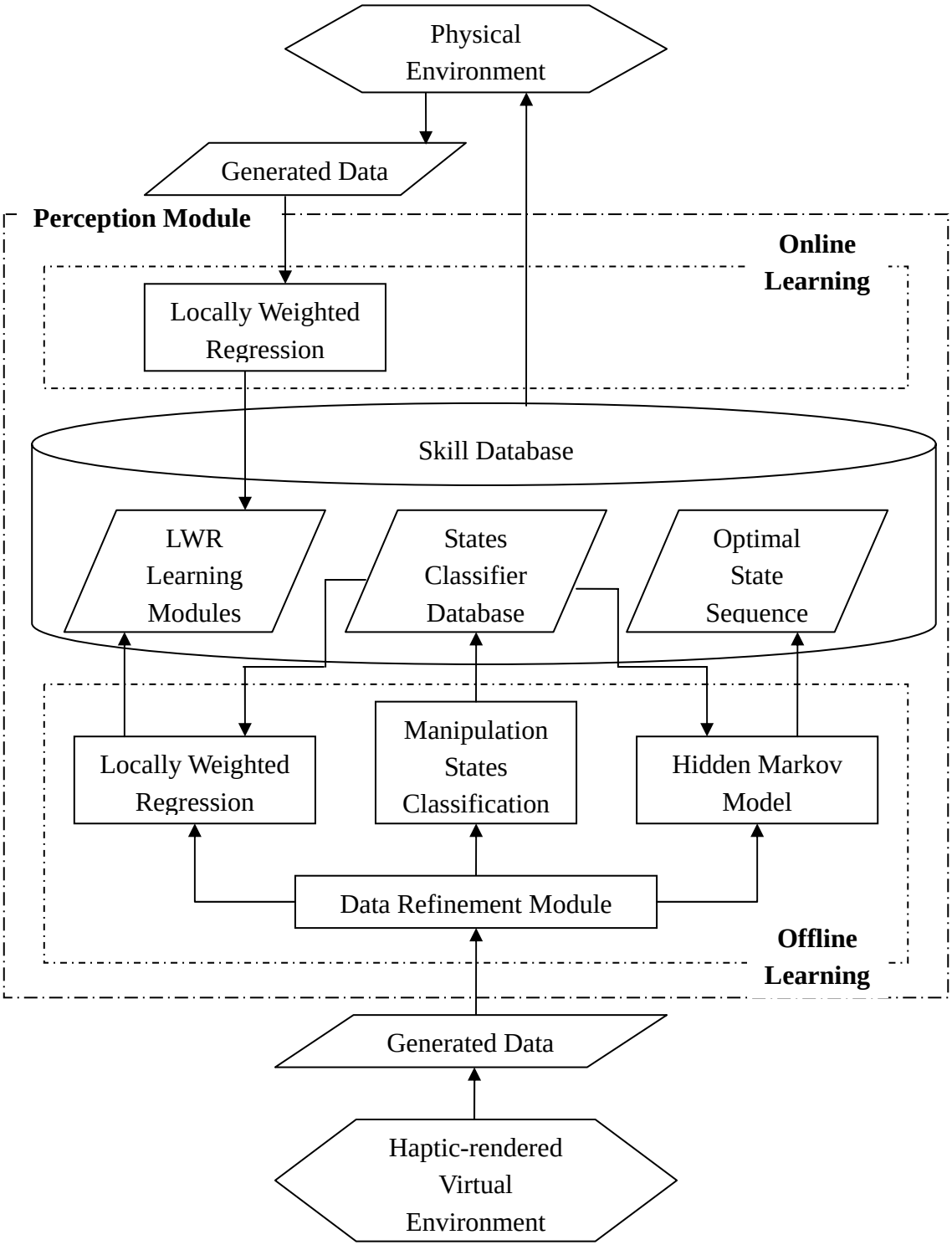



Figure 4.1 Structure of the Perception Module

### 4.3.1 Noise Removal

The training data generated from the haptic-rendered virtual environment may include inconsistent or unintended actions. These data must be identified and removed from the training data in the Perception Module before skill acquisition analysis.

The noise in the data is removed before the analysis. The following algorithm [Kaiser 1996] is applied to refine the data, in which $u$ is the action vector:

1. Remove irrelevant actions that do not enhance an action. This is defined by $\|u\| \le \delta_s, \delta_s \ge 0$, where $\delta_s$ is an application-specific threshold.
2. Remove the operator's rough control. If the differences between two continuous actions are too large, their average value will be used instead.

### 4.3.2 Data Compression

In some situations, the dimension of the input vector is large, but the components of the vectors are highly correlated (redundant). It is useful in this situation to reduce the dimension of the input vectors. An effective procedure for performing this operation is Principal Component Analysis (PCA) [Smith 2002]. This approach generates the following outcomes:

(a) It orthogonalises the components of the input vectors so that they are uncorrelated with each other.

(b) It orders the resulting orthogonal components and principal components, so that those with the largest variation come first.

(c) It eliminates those components that contribute the least to the variation in the data set.

Principal Component Analysis (PCA) is a classical statistical method which identifies patterns in data and expresses the data to highlight their similarities and differences. It is a common technique for finding patterns in data of high dimension. Once these patterns in the data have been found, the data can be compressed. The method is based on linear transformation which has been widely used in data analysis and compression and has found popular use in face recognition and image compression.

The process of human-to-robot skill transfer usually involves recording many task states and sensory information. This results in a large number of dimensions in the recording of human skills data and low efficiency due to the presence of redundant data. Reduction in the dimension of data helps to interpret and understand the data more effectively. In addition, it increases the efficiency of the machine learning algorithms.

Principal Component Analysis is based on the statistical representation of a random variable. The aim is to find a set of $M$ orthogonal vectors in the data space that account for as much data variance as possible. Projecting the data from its original $N$-dimensional space onto the $M$-dimensional subspace spanned by these vectors results in a dimensionality reduction that often retains most of the intrinsic information in the data [Smith 2002].

For given a set of training vectors x from the n dimensional input space $R^n$,

$$X(t_k) = (x_1(t_k), x_2(t_k), ..., x_n(t_k)) \tag{4.3}$$

where $t_k$ is the $k$th observation time, the Principal Component Analysis looks for $y_1(t_k)$, which is a linear combination of the components of $X(t_k)$

$$y_1(t_k) = X(t_k) \times \lambda_1 \tag{4.4}$$

so that the approximation to $X$

$$\hat{X} = y_1(t_k) \times \lambda_1 \tag{4.5}$$

$$e = \{\left\| X - \hat{X} \right\|^2\} \tag{4.6}$$

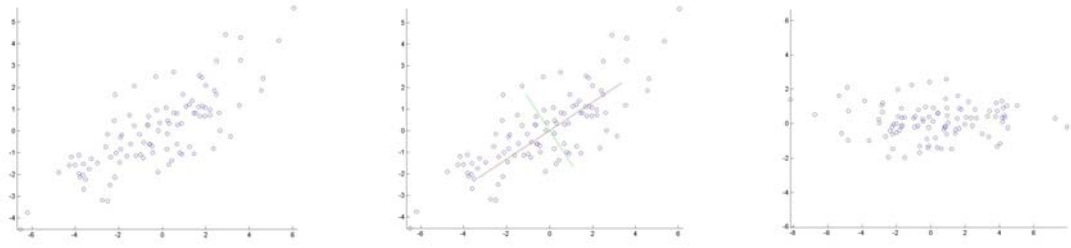
$e$ is minimised.

The residuals, $\lambda_2$ can be found using the same method. In general,

$$\hat{X}(t_k) = \sum_{j=1}^{l} [X(t_k) \times \lambda_j] \times \lambda_j + \varepsilon_k \tag{4.7}$$

where $\varepsilon_k$ is the residual.

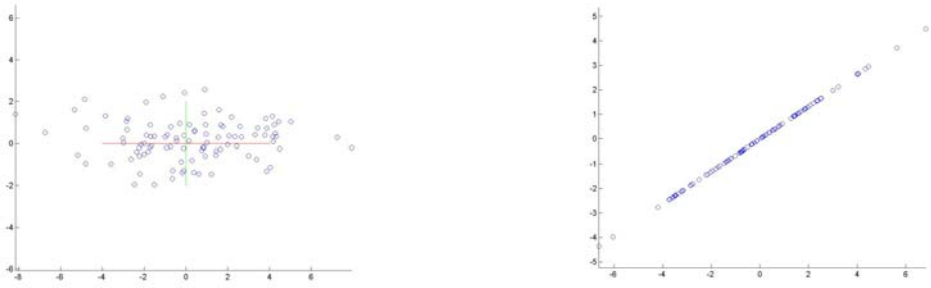
Examples of how to compress two-dimensional data set into one-dimension by applying Principal Component Analysis algorithm, are shown in Figure 4.2. Specific steps of the method are described respectively.



a. Original data

b. Normalised data with the eigenvectors of the covariance matrix

c. Rotate the data



d. New data points by applying the PCA using both eigenvectors

e. The reconstruction from the data derived using only a single eigenvector

Figure 4.2 PCA example

The learning data received from the Manipulation Task Planner Module was refined before being passed to the Learning Module.

## *4.4 Trajectory Cloning by Data Mining Tools*

As the first attempt in this research, the data mining tool See5 was applied in offline analysis on one hundred groups of data, which are collected from the haptic-rendered virtual environment. The peg-in-hole assembly task was used as the case study application. The dataset was recorded in the form of (position, force, torque, orientation), as input into the Perception Module. Output of the Perception Module

are the sensor feedbacks to the control motor controlling the peg, which includes the motions for example, forward, backward and rotation, etc.

One hundred groups of successful peg-in-hole assembly tasks were performed. The experimental data, including position, force and torque information obtained by manually controlling the virtual haptic-rendered peg-in-hole assembly model by the PHANToM were recorded in a data base. The characteristics of insertion were varied for each insertion as much as possible. For example, operations were performed at different speeds, some with less reliability, experiencing occasional oscillations and collisions in some cases. Some insertions were performed conservatively and slowly to potentially avoiding oscillation of the dynamic peg and damaging the peg and the hole in physical insertion. Oscillation of the dynamic peg in the virtual haptic-rendered environment, caused by the gravity of the peg and the vibration of operator's hand when handling the haptic probe, can strongly preclude learning of the manipulation skills. It, however, does not adversely affect the insertion process in the virtual environment. Any damage caused by collision between the peg and the hole can be avoided by the ReachIn maximum force and torque settings. The constraints imposed by this algorithm are scaled to match the limitations of the physical sensor.

The recorded data is interpreted as noise when there are force or torque signals present in the signal without any collision or jamming between the peg and hole. Another kind of noise data can be removed by comparing the data points with the Mathematical Expectation of force and torque at certain positions. For example,

wrong force or torque directions with large or small values are sum of the real force or torque value and noise caused by rough control of the operator. Such data is removed from the training set.

Two methods are used in generating the optimum trajectory, namely, General Strategy and Short Strategy.

In the General Strategy method, the Mathematical Expectation of force and torque data, $E(f_n(force, torque))$, is estimated at every position $p_n$.

In this case, the force and torque data are random discrete values. Following formula is used to calculate the Mathematical Expectation of the one hundred groups of force and torque data at every position $p_n$:

$$E(f_n(force, torque)) = \sum_n \Pr ob(f_n(force, torque)) \cdot f_n(force, torque) \qquad (4.8)$$

Where $\Pr ob(f_n(force, torque))$ is the corresponding probability of the force and torque data at any position $p_n$ and $f_n(force, torque)$ is the force and torque data at that position.

This trajectory provides a general strategy for the operation of the operator. The force and torque results are illustrated in Figure 4.3. The noise positions are removed by employing the method introduced in section 4.3.1. The abnormal positions, which are the positions recorded in the one hundred individual peg-in-hole assembly tasks, but appears less than 10 times (less than 10%) in overall position data base, are removed. The most frequent 2695 positions, appeared in the one hundred attempts, are used as the testing position sample, while the Mathematical Expectation force and

torque value at this position is depicted accordingly.

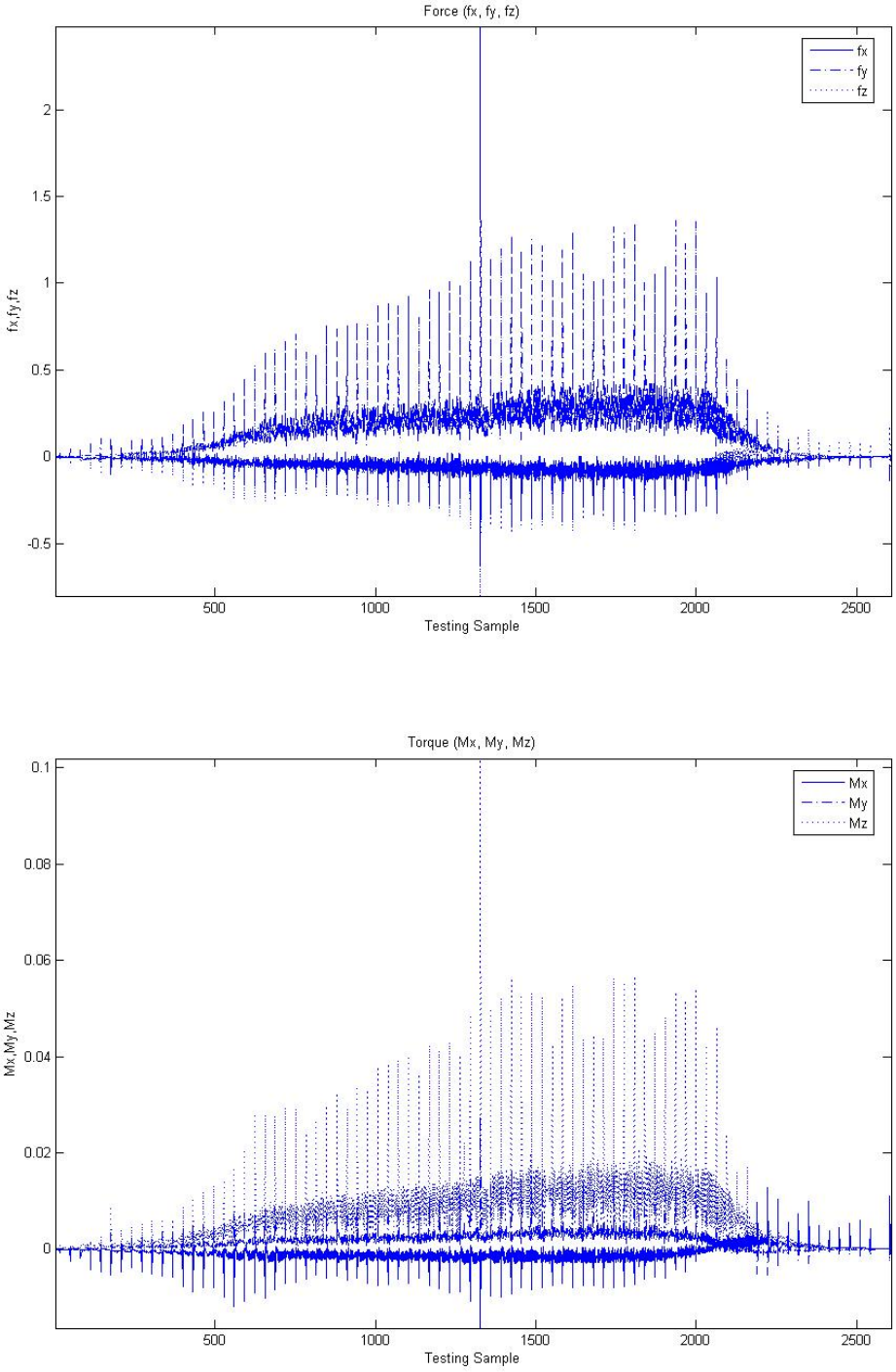



Figure 4.3 Mathematical expectations of force and torque data

In the Short Strategy method, after the removal of the noise, the trace with the shortest time is chosen. The force and torque results are depicted in Figure 4.4.

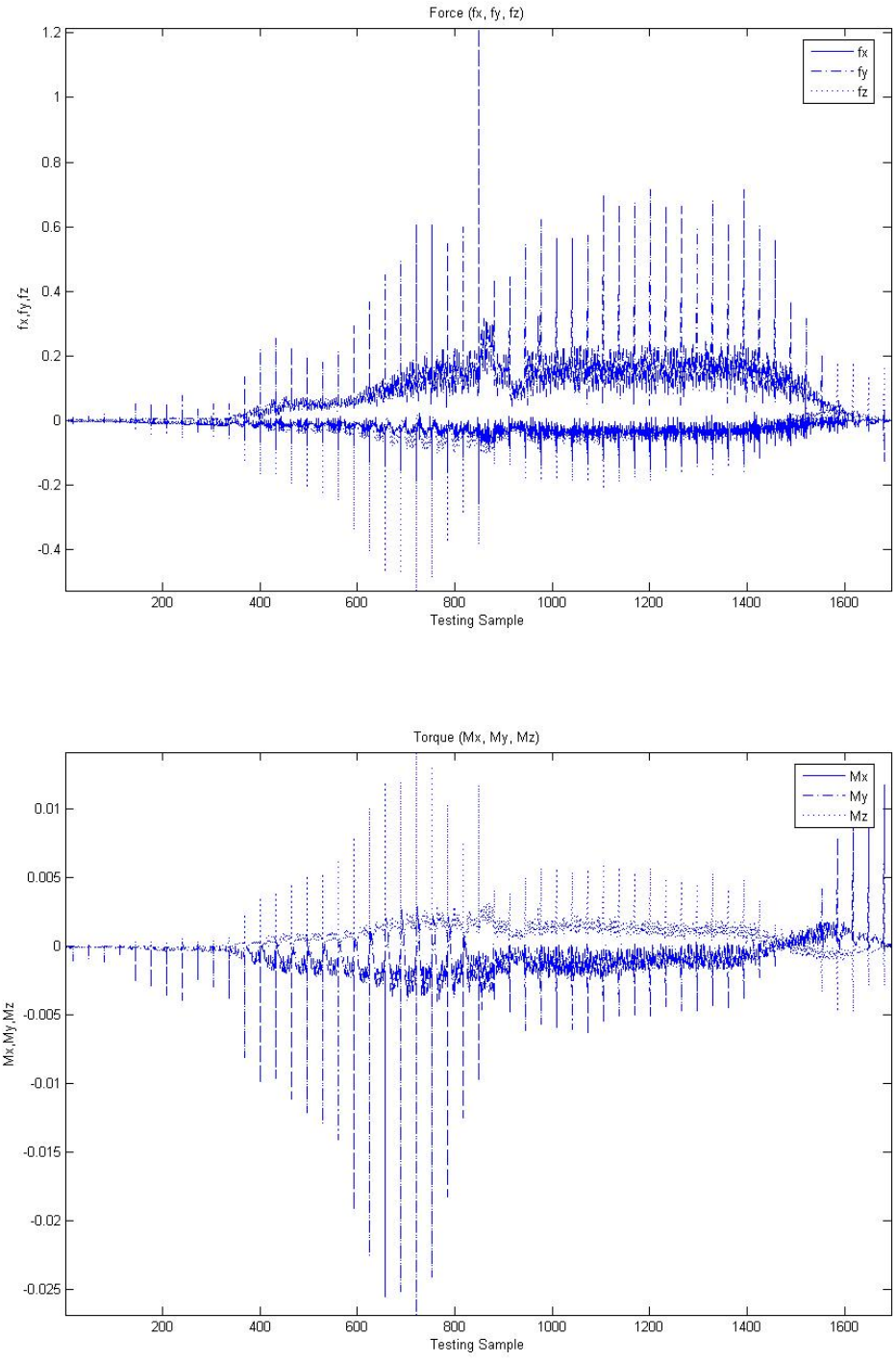



Figure 4.4 Fastest operation result

Since, the trace is sampled at a frequency of 30Hz, a successful trace typically lasts approximately between 40s to 130s, providing about 1200 to 4000 samples. Strategies for faster operators are more complex. This can vary from one operator to another, requiring more advanced skills to decrease oscillations and large collisions due to high acceleration. Hence, the General Strategy produces better results than the Short Strategy in this application.

The C4.5 algorithm in the data mining tool See5 (Windows version) is used to generate the symbolic rules from the data obtained from virtual environment. The rules specify basic skills by deriving the assembly trajectories according to the desired goal or responses. By choosing the “Ruleset” selection as the goal of the algorithm, the decision trees as “if—then statements” are generated. The rules can be easily converted into C language.

The See5 operates based on two input files: the Names file and Data file. The Data file contains the recorded data set in the format of (position, force, torque, orientation). The Names file, which describes the attributes and classes, contains the recorded attributes and classes of the force and torque data. Since it is important to know the current position of the robot tool tip and the direction of the next movement, the directions of the force and torque signals are more important than their magnitudes. Hence, the attribute values of force and torque are defined as negative, positive, N/A and zero. The classes file defines the actions which should be taken by robot on a

particular condition. It includes Forward, Stop, Backward, Right, Left, Backward－Right and Backward－Left. The generated rule has a typical format as shown below:

Rule 1:

fy = Negative

-> class Backward

Rule 2:

fx = Positive

fy = Negative

tz = Positive

-> class Backward-Right

……

Default class: Forward

The C4.5 algorithm is often used in inducing behaviour cloning controllers. This cloning method can provide some insight into the manipulation strategy. However, this kind of method cannot reflect the causal relationships in the manipulation procedure and the goal structure of the manipulation strategy. Hence, the conceptual structure is not comprehensive. Moreover, this kind of method does not explicitly show any time ordering between events in the controlled system and the actions taken.

In the following sections, more innovative algorithms will be employed to identify different manipulation states in order to reflect the overall manipulation control strategy.

## *4.5 Trajectory Cloning using Manipulation State Classification*

The trajectory cloning, widely used in behavioural cloning technology, is employed in this work as a major method for classifying different manipulation states. When the trajectory constraints, groups of dataset, are recorded in a database and then analysed, the characteristic of each manipulation state can be easily generalized. Moreover, cloning different trajectories in different manipulation states will be much easier than cloning of the entire manipulation trajectory. Since in the same state, trajectory constraints are more similar than that of the entire manipulation trajectory, different machine learning algorithms can be used in different states to clone the trajectories. In contrast to cloning entire trajectory as discussed in section 4.5, generalising different state characteristics is more accurate and time efficient. This section has its focus on algorithms which can identify different manipulation state characteristics.

The Hidden Markov Models is applied to the offline data obtained from the haptic-rendered virtual environment to acquire the basic skills applied by the human operator during virtual manipulation. The data and the generalized three groups of state characteristics by three different methods, namely the physical contact relationships, the Fuzzy Gustafson-Kessel clustering algorithm and the Competitive Agglomeration algorithm, are used to generate three optimal state sequences individually.

### 4.5.1 Manipulation States Classification by Physical Contact Relationships

The basic skills are defined according to the contact-state transition of a task, independent from the configuration of a manipulator [Nakamura 1996]. In a virtual manipulation environment, the basic skills can be also identified by the contact states and state changes [Onda 1995; Takamatsu 1999]. Using this approach, the basic skills can be automatically extracted form the manipulation carried out in the virtual environment.

In the first stage, the peg-in-hole insertion procedure is classified into several states according to the peg and hole's contact relationships. In order to express the geometric information of the peg and hole, the edges and surfaces of the peg and hole are indexed by symbols. As shown in the Figure 4.5, *P* and *H* represent the peg and the hole respectively, whereas *e* is the edge, *s* is the surface and *o* is the outer platform surface of the hole.

The recorded training data is classified into several sub-databases and an index is assigned to each according to the peg and hole's contact relationships. When the peg and hole's contact states correspond to one of those states, the input information is directly indexed to a specific sub-training data. This streamlines and speeds up the search of the training database.

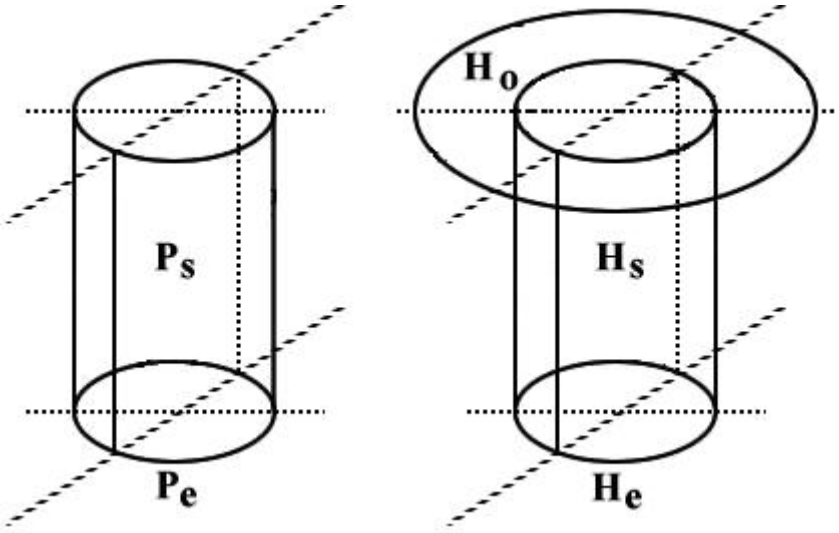



Figure 4.5 Peg and hole indexed by symbols

The contact relations of the peg-in-hole assembly state classifications are shown in Table 4.1. When there is no peg and hole contact, this kind of contact relationship is grouped as state one. The differences between similar contact states, such as peg left edge to the hole outer right surface and peg right edge to the hole outer left surface and so on, are simply distinguished by force and torque directions, which can be grouped as one state. Hence, the peg edge and hole outer surface contact, peg edge and hole inner surface contact, peg edge and hole edge contact, peg surface and hole edge contact, and peg surface and hole inner surface contact can be grouped as state two, state three, state four, state five and state six respectively.

All the remaining states are the jamming states. State seven is a two-point contact state in which the point on the peg edge touches the hole inner surface and hole edge contacts peg surface. State eight is a two point-contact state that the peg edge touches the hole edge. State nine is a three-point contact state in which the peg-edge touches

the hole-edge as well the hole inner surface. Variations in the hole angles and the peg position can transfer the system from jamming state to non-jamming state.

| State | Contact | Condition (6 DOF) | Condition (3 DOF) |
|---|---|---|---|
| 1 | No contact | 0 | 0 |
| 2 | One-contact | (Pe , Ho) | (Pe , Ho) |
| 3 | One-contact | (Ps , He) | (Ps , He) |
| 4 | One-contact | (Pe , Hs) | (Pe , Hs) |
| 5 | One-contact | (Pe , He) | State 2 or 4 |
| 6 | One-contact | (Ps , Hs) | Never happen |
| 7 | Two-contact | (Pe , Hs) & (Ps , He) | (Pe , Hs) & (Ps , He) |
| 8 | Two-contact | (Pe , He) | Never happen |
| 9 | Three-contact | (Pe , He) & (Ps , He) | Never happen |

Table 4.1 6 DOF assembly classification

The physical system has only three degrees of freedom (3 DOFs) compared to six degrees of freedom (6 DOFs) in the virtual model. This implies that that some of the states defined in the haptic-rendered virtual environment do not have any direct correspondence in the physical system. For example, the state 6 defined in Table 4.1 will never happen in the three degrees of freedom (3 DOFs) physical system. It requires the axes of the peg and hole to be parallel and have a distance of 0.05 mm

which is impossible in practice. The centre of the circular outer platform will always be on the axis or the extension axis of the peg. The contact states 8 and 9 cannot happen in the physical system either. As shown in Figure 4.6, such a scenario requires points b and c to be the contact points on the edges of the peg and the hole, and point a on the hole edge would have come into the peg, which is impossible in the real life. The contact state 5, the one-point contact scenario which happens between the peg and the hole edge is quite unlikely. This can be further grouped into contact states 2 or 4 depending on the directions of the force and the torque.

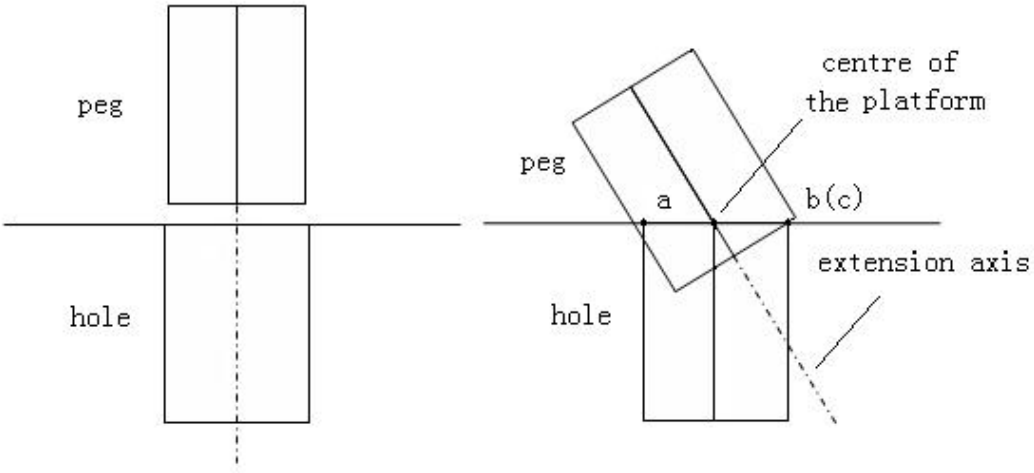



Figure 4.6 Supposed state 8 or 9

Hence, the contact state relationships in the three degrees of freedom (3 DOFs) physical system can be depicted as shown in Table 4.2. Manipulation states are also redefined.

| State | Contact | Condition (3 DOF) |
|---|---|---|
| 1 | No contact | 0 |
| 2 | One contact | (Pe , Ho) |
| 3 | One contact | (Ps , He) |
| 4 | One contact | (Pe , Hs) |
| 5 | Two contacts | (Pe , Hs) & (Ps , He) |

Table 4.2 3 DOF assembly classification

The peg-in-hole insertion procedure is illustrated in Figure 4.7. The contact states are modelled using their kinematic constraints. The contact configurations correspond to the discrete states of a discrete event system.

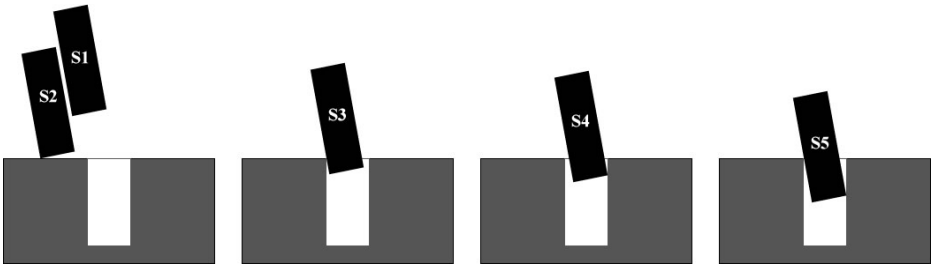



Figure 4.7 Peg-in-hole contact states

The state classification result along with the peg position and force and torque values of one successful peg-in-hole insertion operation is depicted in Figure 4.8. The generated state characteristics will be used to generate optimal state sequences using the Hidden Markov Model as described section 4.7.

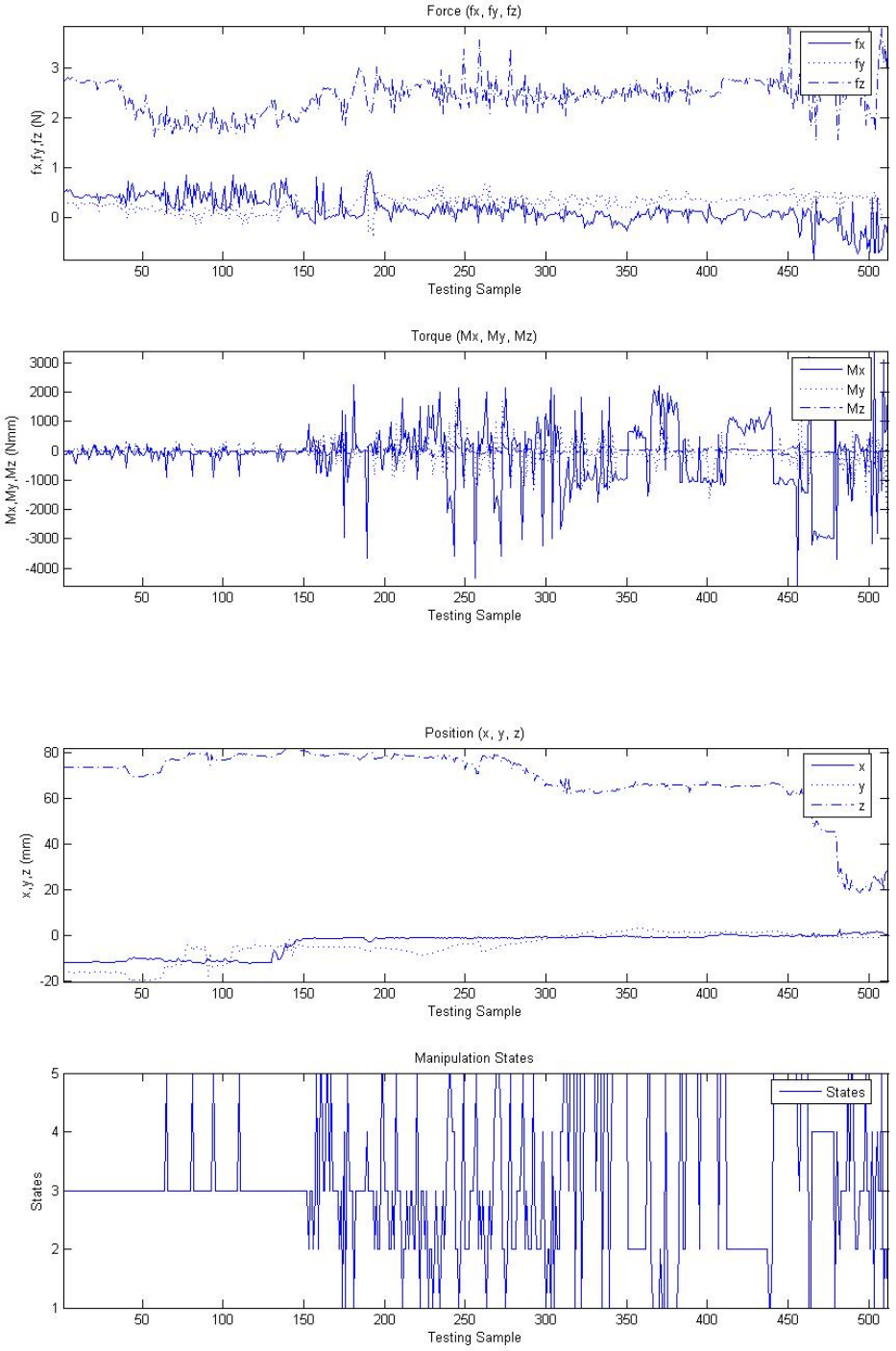



Figure 4.8 State classification result by physical contact state

### 4.5.2 Manipulation States Classification by Fuzzy Clustering Algorithm

Instead of obtaining the basic skills from the contact-state transition of a task, they can be obtained from in virtual environment using statistical methods.

Fuzzy clustering algorithms are described in this section. These methods use statistic analysis of a large amount of data, for example, the data generated from a haptic-rendered virtual environment, and divide them into clusters according to their common characteristics. The fuzzy c-means algorithm (FCM) as the basic fuzzy clustering method will be introduced. The Fuzzy Gustafson-Kessel (FGK) algorithm and Competitive Agglomeration (CA) algorithm, as some variations of FCM designed to produce optimal group number of clusters will be then described.

#### 4.5.2.1 Fuzzy c-means Clustering Algorithm

The fuzzy c-means algorithm (FCM) is a widely used fuzzy clustering algorithm [Bezdek 1981]. All other fuzzy clustering algorithms are evolved from the fuzzy c-means algorithm. It is important to have a detailed understanding of this algorithm. This algorithm divides a given dataset $X = \{x_1, ...x_n\} \subseteq R$ into c clusters by minimising the following function:

$$J(X, U, \beta) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} d^2(\beta_i, x_j) \tag{4.9}$$

Subject to $\sum_{j=1}^{n} u_{ij} > 0$ for all $i \in \{1, ..., c\}$, which guarantees that no cluster is

empty; $\sum_{j=1}^{n} u_{ij} = 1$ for all $j \in \{1,...,n\}$, which ensures that the sum of the membership degrees for each datum equals 1, where $u_{ij} \in [0,1]$ is the membership degree of datum $x_j$ to cluster $i$, $\beta = (c_i)$ is the prototype of cluster $i$, $c_i$ is the centre of cluster $i$, and $d(\beta_i, x_j)$ is the distance between datum $x_j$ and prototype $\beta_i$. The $c \times n$ matrix $U = [u_{ij}]$ is also called the fuzzy partition matrix and the parameter $m$ is called the fuzzifier. Usually $m$ is chosen as two.

The fuzzy c-means algorithm divides a given dataset $X$ into c clusters of equal size and shape. The shape of the clusters depends on the distance function $d^2(c_i, x_j)$. Using the Euclidean distance as the most commonly used option, the data is divided into c spherical clusters.

Although the fuzzy c-means algorithm is widely used, it fails for some classification tasks. If the shape of the clusters is not spherical or if the clusters differ considerably in their size, the result of the fuzzy c-means algorithm is often not very intuitive and only poorly fits the data. Another problem of the fuzzy c-means algorithm is its sensitivity to noise and outliers. This sensitivity is caused by training every datum with the same weight and thus the same influence on the classification result [Ahmed 2002].

#### 4.5.2.2 Fuzzy Gustafson-Kessel Clustering Algorithm

The Fuzzy Gustafson-Kessel algorithm (FGK) is applied to the trajectory associated with the

manipulation performed by the operator in the haptic virtual environment. The algorithm classifies the trajectory into a number of clusters, each representing a state. In the first stage of the process, the constraints on the state variables are learned. These constraints determine the corresponding desired state and choose the best trajectory to the next state. The Fuzzy Gustafson-Kessel classifier, obtained by training the data from the haptic-rendered virtual environment, will be used in this work to estimate the assembly states in online analysis. Principal Component Analysis (PCA) is then applied to each cluster for dimension reduction and feature extraction. The number of clusters is determined based on a flatness index of the clusters combined with the PCA compression rate [Babuska 1995]. Actions that determine the best trajectory from the current state to the next state are computed using knowledge of the system dynamics, acquired through the trained Hidden Markov Model (HMM). Locally Weighted Regression (LWR) method will be encoded as the online nonlinear function approximator in each cluster to generate the trajectories.

Figure 4.9 illustrates the skill-acquisition procedure. The first row represents the offline training procedure. One hundred groups of data obtained from the haptic-rendered virtual environment are used in Fuzzy Gustafson-Kessel classifier training. Thin lines represent the flow of data in the training progress. The second row represents the online data analysis. Thick lines represent the sensory data in the physical assembly procedure obtained from the experimental rig. Dotted lines represent the use of the previously trained module. Data is refined in the Manipulation Task Planner Module by the algorithm introduced in Section 4.3.1.

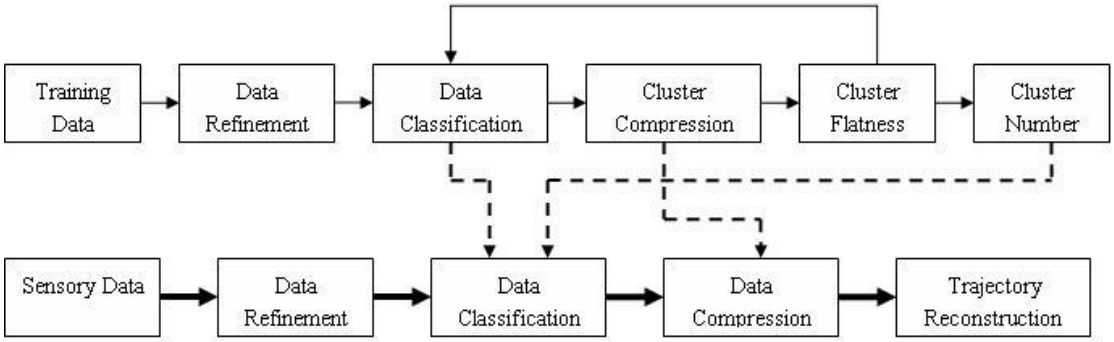



Figure 4.9 Skill acquisition procedure using FGK

The Fuzzy Gustafson-Kessel clustering algorithm has been widely used for fuzzy data analysis and pattern recognition [Babuska 1998], because it is flexible in automatically determining the optimal hyper-ellipsoidal cluster from the training data set. This algorithm uses statistical analysis of the data generated from the haptic-rendered virtual environment, and divides them into groups according to their common characteristic.

A separate norm matrix $M_i$ is calculated to determine the shape of the clusters. These norm matrices are updated together with the centres of the corresponding clusters. Therefore the prototypes of the clusters are pairs ($v_i$, $F_i$).

$$M_i = (\det F_i)^{1/n} F_i^{-1} \tag{4.10}$$

where $v_i$ is the centre of the cluster and $F_i$ is the covariance matrix, which defines the shapes of the clusters. The parameters $v_i$ and $F_i$ are defined in equations 4.11 and 4.12 respectively.

$$v_i = \frac{1}{N_i} \sum_{j=1}^{N} u_{ij}^m z_j, \tag{4.11}$$

$$F_i = \frac{1}{N_i} \sum_{j=1}^{N} u_{ij}^m (z_j - v_i)(z_j - v_i)^T \tag{4.12}$$

The Fuzzy Gustafson-Kessel clustering algorithm computes the distance to the prototypes as:

$$d^2(z_j, v_i) = (z_j - v_i)^T M_i (z_j - v_i) \tag{4.13}$$

where $z_j$ represents each data set.

Fuzzy Gustafson-Kessel clustering algorithm searches for the partition matrix and the cluster prototypes in order to minimise the following function:

$$J(Z, V, U) = \sum_{i=1}^{c} \sum_{j=1}^{N} u_{ij}^m d^2(z_j, v_i) \tag{4.14}$$

where $u_{ij}$ is the membership degree of the data vector $z_j$ in the $i$th cluster.

The prototypes are updated according to equations 4.11 and 4.12 [Hoppner 1999]:

Principal Component Analysis (PCA) algorithm is then applied on the recorded data from the haptic-rendered virtual environment, because the dimension of the recorded data from the haptic-rendered virtual environment is large, and the components of the data are highly redundant. It is useful in this situation to reduce the dimension of the input vectors. Principal Component Analysis (PCA) is an effective procedure for performing this operation.

The number of clusters must be specified before clustering. The higher the number of clusters, the finer will be the approximation of the nonlinearity. However, it will require the estimation of more parameters with higher variances. If no prior knowledge on the number of clusters is available, then automatic procedures can be applied.

A new approach for automatic determination of the number of clusters, based on a

flatness index of the cluster [Babuska 1995] and Principal Component Analysis (PCA), has been developed in this thesis. In this approach, the eigenvalues are sorted from the cluster covariance matrix $C_i$ in a descending order, $\lambda_{i,1} \geq \lambda_{i,2} \geq \cdots \geq \lambda_{i,n}$. When approximating the regression surface, the obtained hyperellipsoidal clusters are flat, that is, one of the axes is much shorter than the others. Consequently, the smallest eigenvalues $\lambda_{i,n}$ are significantly smaller than the remaining ones. In order to efficiently approximate the regression surface by hyperplanes, the clusters should be as flat as possible. The flatness index $t_i$ of a cluster is defined as a ratio between the smallest and the largest eigenvalues.

$$t_i = \lambda_{i,n} / \lambda_{i,1} \tag{4.15}$$

Where $\lambda_{i,n}$ and $\lambda_{i,1}$ are the smallest and the largest eigenvalues respectively.

An aggregate measure value $t_A$ called the average cluster flatness is defined in formula (4.16)

$$t_A = \frac{1}{c} \sum_{i=1}^{c} \frac{\lambda_{i,n}}{\lambda_{i,1}} \tag{4.16}$$

The data cluster sets obtained from the Fuzzy Gustafson-Kessel Model consist of fourteen different parameters, including force and position variables, and could be investigated further in the future research. In order to reduce the dimension of the data, Principal Component Analysis (PCA) is applied to each cluster. The compression level $p_{i,n}$ is determined according to the error produced in the compression process.

$$p_{i,n} = d_{i,n} / D_{i,n} \tag{4.17}$$

Where $d_{i,n}$ is the most significant eigenvalue number or the remaining dimension numbers after compression, and $D_{i,n}$ is the number of original eigenvalues.

The eigenvalue number $d_{i,n}$ is decided by an error percentage $e_i$.

$$e_i = \{\left\| X_i - \hat{X}_i \right\|^2\} \tag{4.18}$$

$e_i$ is the Frobenius norm between the compressed matrix $\hat{X}_i$ and original data matrix $X_i$. The error tolerance $E$ must be set before the calculation of $p_{i,n}$. The number of clusters and the compression rate of each cluster is calculated by the following function:

$$PT = \frac{1}{c}\sum_{i=1}^{c}\left(\frac{\lambda_{i,n}}{\lambda_{i,1}} \times p_i\right) \tag{4.19}$$

The optimal values of $\lambda_{i,n}$ and $d_{i,n}$, are found by minimising *PT.*

As a result, five clusters are determined. The compression rate $p$ and the error percentage $e$ for each cluster are shown in Table 4.3.

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 |
|---|---|---|---|---|---|
| $p$ ($e$ < 10%) | 28.6% | 21.4% | 35.7% | 35.7% | 28.6% |

Table 4.3 Error percentage below 10% and the maximum compression rate achieved

The state classification result along with peg position and force and torque values of one successful peg-in-hole insertion operation is depicted in Figure 4.10. The

generated state characteristics will be used to generate optimal state sequences by the Hidden Markov Model in section 4.7.

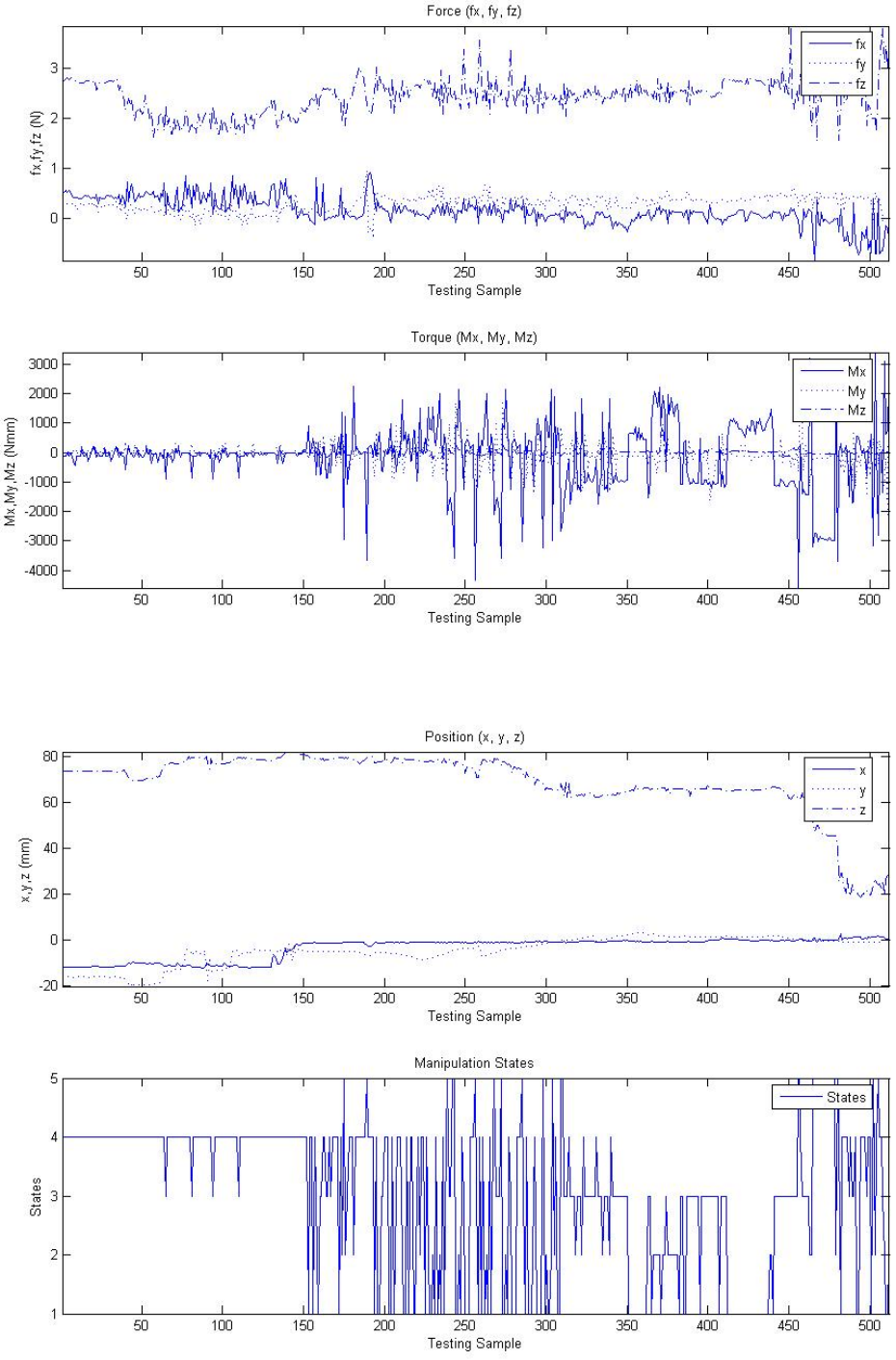



Figure 4.10 State classification result by FGK

#### 4.5.2.3 Competitive Agglomeration algorithm

The Competitive Agglomeration (CA) algorithm is applied to the offline data obtained from the haptic-rendered virtual environment to acquire the basic skills applied by the human operator during virtual manipulation. It has been necessary to remove the noise in the data before the analysis. Data was refined in the Manipulation Task Planner Module by the algorithm introduced in Section 4.4.

The algorithm classifies the trajectory into a number of clusters, each representing a state. In the first stage of the process, the constraints on the state variables are learned. These constraints determine the corresponding desired state and choose the best trajectory to the next state. The Competitive Agglomeration classifier, obtained by training the data from the haptic-rendered virtual environment, is used to estimate assembly states in online analysis. Actions that determine the best trajectory from the current state to the next state are computed using knowledge of the system dynamics, acquired through the trained Hidden Markov Model (HMM). Locally Weighted Regression (LWR) method will be encoded as the online nonlinear function approximator in each cluster for the trajectories.

Figure 4.11 illustrates the skill-acquisition procedure. The first two rows represent the offline training procedure. One hundred groups of data obtained from the haptic virtual environment are used in the Competitive Agglomeration classifier training. In this diagram, thin lines represent the flow of data in the training progress, the third row represents the online data analysis, thick lines represent the sensory data in the

physical assembly procedure from the experimental rig, and dotted lines represent the use of the previously trained module.

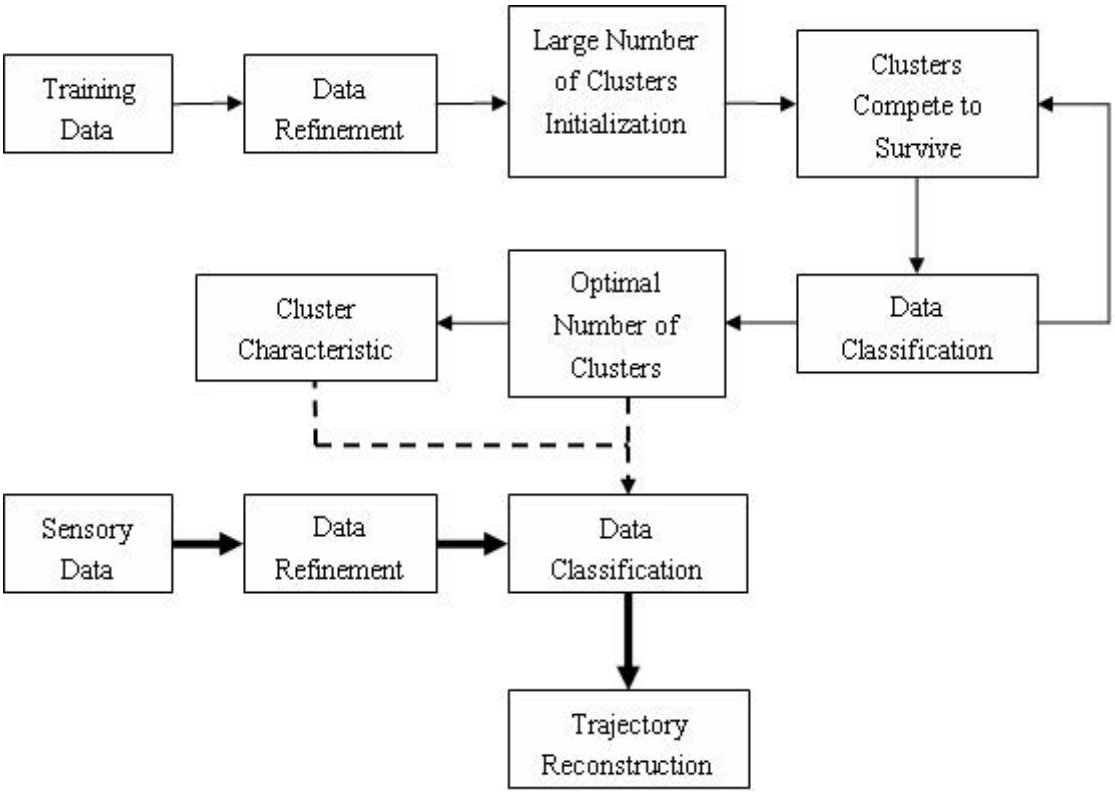



Figure 4.11 Skill acquisition procedure using CA

The Competitive Agglomeration algorithm has several advantages over other clustering algorithms:

(a) The clustering does not suffer from initialisation and the local minimum.

(b) The optimal number of clusters is determined after minimisation of the fuzzy prototype-based object function.

(c) The points are dynamic and can move from one cluster to another to minimise the fuzzy prototype-based object function.

(d) The algorithm can be used to find clusters of different sizes and shapes by using an appropriate distance measure in the fuzzy prototype-based object function.

The Competitive Agglomeration (CA) algorithm commences by classifying the data set into a large number of clusters. As the training proceeds, clusters compete for survival. Those with large cardinalities survive and clusters that lose the competition are removed. The training process gradually decreases the number of clusters. This will result in an optimal number of clusters when the fuzzy prototype-based object function is minimised.

Frigui and Krishnapuram [Frigui 1997] first introduced the Competitive Agglomeration (CA) algorithm as one of the clustering methods mostly for use in image segmentation.

The Competitive Agglomeration (CA) algorithm searches the optimal cluster prototypes in order to minimise the following prototype-based object function:

$$J(Z,U,V)=\sum_{i=1}^{M}\sum_{j=1}^{N}u_{ij}^{2}d^{2}(z_j,v_i)-\alpha\sum_{i=1}^{M}[\sum_{j=1}^{N}u_{ij}]^{2} \tag{4.20}$$

Subject to $\sum_{i=1}^{M}u_{ij}=1,$ for $j\in\{1,\ldots,N\}.$ (4.21)

where $Z=\{z_1,\ldots,z_N\}$ is a set of $N$ data objects represented by $n$-dimensional feature vectors. $V=[v_1,\ldots,v_M]$ represents $M$ cluster prototypes, each of which have to be determined, and $v_i$ is the centre of the cluster. $U=[u_{ij}]$ is the membership degree of the data vector $z_j$ in the $i$th cluster. $d^2(z_j,v_i)$, representing the distance from

feature vector $z_j$ to the cluster centre $v_i$.

The Fuzzy Maximum Likelihood Estimation (FMLE) algorithm [Gath 1989] is used to divide the given data sets into clusters of different sizes and different shapes. FMLE interprets the data set as a *p*-dimensional normal distribution. The distance of a datum to a cluster is inversely proportional to the posterior possibility (the probability of selecting the *i*th cluster given the *j*th feature vector) that a datum $z_j$ is the realisation of the *i*th normal distribution.

$$d^2(z_j, v_i) = \frac{(\det(F_i))^{\frac{1}{2}}}{P_i} \exp\left(\frac{(z_j - f_i)F_i^{-1}(z_j - f_i)^T}{2}\right) \tag{4.22}$$

$$F_i = \frac{1}{N_i}\sum_{j=1}^{N} u_{ij}^m (z_j - v_i)(z_j - v_i)^T \tag{4.23}$$

$$P_i = \frac{1}{N}\sum_{j=1}^{N} u_{ij} \tag{4.24}$$

The first part of the prototype-based object-function $J(Z,U,V)$, as the sum of squared distances to the prototypes weighted by constrained memberships, is used to control the shapes and sizes of the clusters and to obtain the compact clusters. The global minimum of the first component is achieved when the number of clusters *M* is equal to the number of samples *N*. The second part of prototype-based object function $J(Z,U,V)$, used to control the number of clusters, is the sum of squares of the cardinalities of the clusters. The global minimum of this part is achieved when all points are in one cluster. When both components are combined and $\alpha$ is chosen properly, the final partition will minimise the sum of cluster distances and divide the data set into the smallest possible number of clusters.

The parameter $\alpha$ is chosen by:

$$\alpha(\theta) = \eta_0 e^{\frac{-\theta}{\tau}} \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} u_{ij}^2 d^2(z_j, v_i)}{\sum_{i=1}^{M}[\sum_{j=1}^{N} u_{ij}]^2} \tag{4.25}$$

where $\theta$ is the iteration number, $\eta_0$ is the initial value and $\tau$ is a decay factor.

The equation for membership $u_{ij}$ update is given by:

$$u_{ij} = u^{FMLE} + u^{Bias} \tag{4.26}$$

$$u^{FMLE} = \frac{1/(2d^2(z_j, v_i))}{\sum_{i=1}^{M} 1/(2d^2(z_j, v_i))} \tag{4.27}$$

$$u^{Bias} = \frac{\alpha}{d^2(z_j, v_i)}(N_i - \frac{\sum_{i=1}^{M} N_i/(2d^2(z_j, v_i))}{\sum_{i=1}^{M} 1/(2d^2(z_j, v_i))}) \tag{4.28}$$

$u^{FMLE}$ is the membership term in FMLE algorithm which takes into account the relative distance of the feature point to all clusters,

$u^{Bias}$ is a signed bias term which depends on the difference between the cardinality of the clusters of interest, and the weighted average of cardinalities from the point of view of feature points.

The value of $\alpha$ is initially set high and then decreased slowly in each iteration to help the Competitive Agglomeration algorithm to seek an appropriate partition with an optimal number of clusters. As the algorithm proceeds, the second part of equation 7 enables the cluster to include as many points as possible. Through the process, a few clusters eventually survive and the rest disappear.

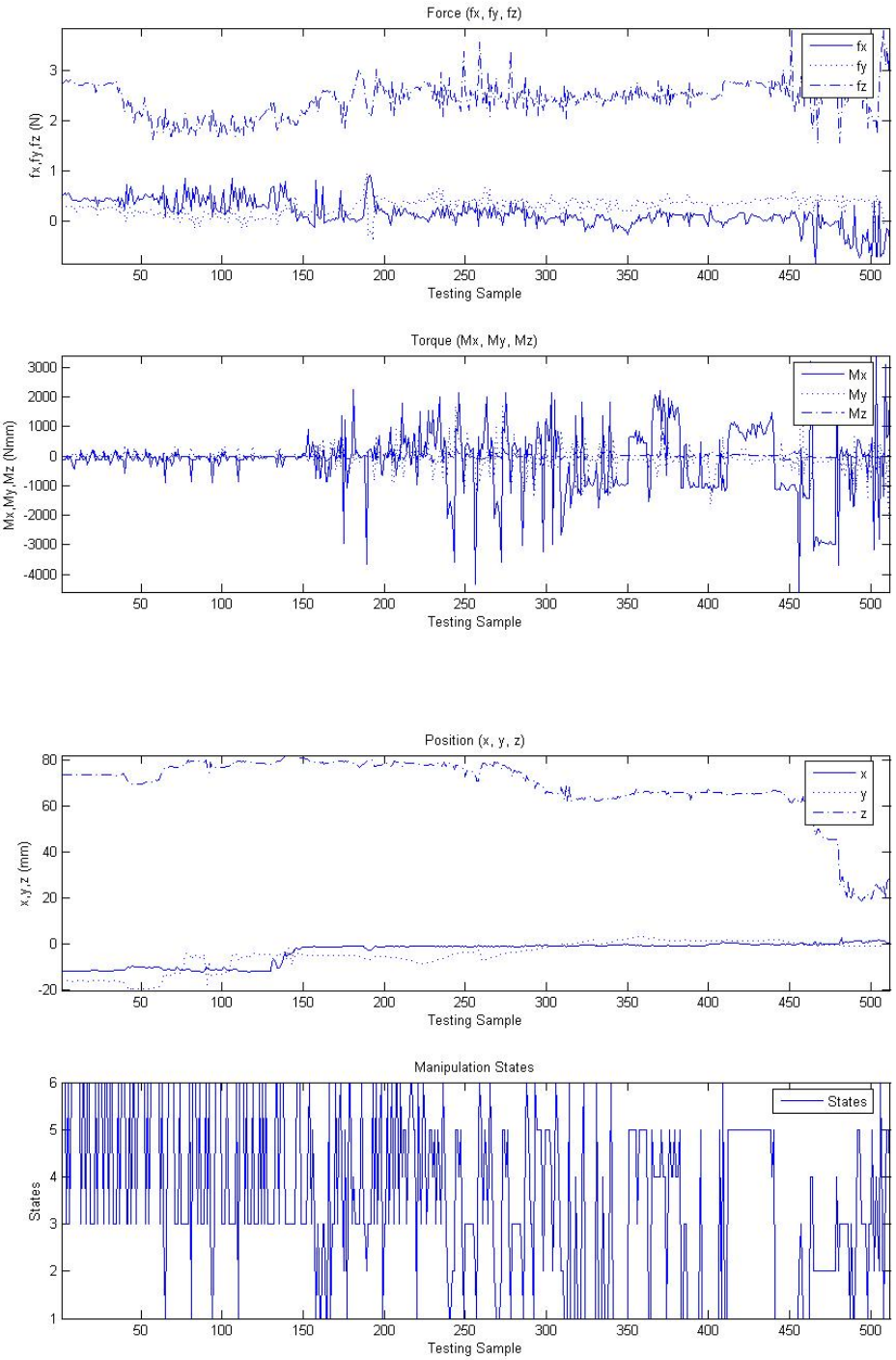



Figure 4.12 State classification result by CA

The state classification result along with peg position and force and torque values of one successful peg-in-hole insertion operation is depicted in Figure 4.12. The

generated state characteristics will be used to generate optimal state sequences by the Hidden Markov Model in section 4.7.

## *4.6 Manipulation States Learning by Hidden Markov Models*

After the manipulation states are identified by the above algorithms in the section 4.6, the Hidden Markov Models (HMM) is applied to the offline data obtained from the haptic-rendered virtual environment to acquire the basic skills applied by the human operator during virtual manipulation. Data is refined in the Manipulation Task Planner Module by the algorithm introduced in Section 4.4. The three optimal state sequences, employing different state classification methods respectively, will be estimated in this section.

### 4.6.1 The Concept of Hidden Markov Models

The contact states between the peg and the hole are identified using the Hidden Markov Models (HMM) which operates as a probability estimator defined by the structure $\lambda = (A, B, \pi)$. The three parameters of this structure are described as follows [Rabiner 1989]:

- Matrix A is the state transition probability distribution matrix defined by

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{4.29}$$

where $a_{ij} = P[S_j(t+1) \mid S_i(t)]$, $1 \le i, j \le N$.

$S$ denotes individual states where $S = [S_1, S_2, \cdots S_N]$.

$N$ is the number of states in HMM model.

$a_{ij}$ is the time independent probability of having state $S_j$ at time $t+1$ given that the state $S_i$ at time $t$. $\sum_j a_{ij} = 1$ for all $i$.

- Matrix B is the observation signal probability distribution matrix defined by

$$B = [b_{1k}, b_{2k}, \cdots b_{Nk}] \tag{4.30}$$

where $b_{jk} = P[v_k(t) \mid S_j(t)]$, $1 \le j \le N$, $1 \le k \le M$.

$M$ is the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the system being modelled.

$v_k(t)$ is the individual visible symbols in a particular visible sequence $V = [v_1(t), v_2(t), \cdots v_M(t)]$.

$b_{jk}$ is the probability of monitoring a particular visible state $v_k$ in any state $S_j(t)$. $\sum_k b_{jk} = 1$ for all $j$.

- $\pi$ is the observed initial signal:

$$\pi = [\pi_i] \tag{4.31}$$

where $\pi_i = P[S(1) = S_i]$, $1 \le i \le N$.

Given appropriate values of $A, B, \pi, M, N$, the HMM model can be used to observe sequence $O = [O_1, O_2, \cdots, O_T]$. Where $O_t$ is one of the symbols from $V$, and $T$ is the number of observations in the sequence.

There are three basic issues in Hidden Markov Model which must be solved in

real world [Rabiner 1989]:

(a) The Evaluation Problem: Gvien a HMM model with a complete probability estimator structure $\lambda = (A, B, \pi)$ and a particular sequence of visible states $O = [O_1, O_2, \cdots, O_T]$, how $P(V / \lambda)$, the probability of the sequence of visible states can be determined.

(b) The Decoding Problem: Given a HMM model with a complete probability estimator structure $\lambda = (A, B, \pi)$ and a particular sequence of visible states $O = [O_1, O_2, \cdots, O_T]$, how the most likely sequence of hidden states *S* that led to the sequence of visible states *V* can be determined.

(c) The Learning Problem: Given a set of training observations of visible symbols, the number of states and the number of visible states, how the probabilities $a_{ij}$ and $b_{jk}$ required for calculating the HMM model estimator structure $\lambda = (A, B, \pi)$ can be determined.

The solution of Evaluation Problem can be carried out by either HMM Forward algorithm or HMM Backward algorithm [Rabiner 1989].

The probability of a sequence $O = [O_1, O_2, \cdots, O_T]$ of visible states produced by HMM model is given by:

$$P(O) = \sum_{r=1}^{r_{\max}} P(O / S_r) P(S_r) \tag{4.32}$$

where *r* represents a particular sequence $S = [S_1, S_2, \cdots S_N]$ of T hidden states.

$$P(S_r) = \prod_{t=1}^{T} P(S(t) / S(t-1)) \tag{4.33}$$

$$P(O/S_r) = \prod_{t=1}^{T} P(O(t)/S(t)) \tag{4.34}$$

Hence, equation (4.32) can be expressed as:

$$P(O) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^{T} P(O(t)/S(t)) P(S(t)/S(t-1)) \tag{4.34}$$

Equation (4.34) can be further simplified by defining forward variable $\alpha_j(t)$

$$\alpha_j(t) = P(O_1, O_2, \cdots, O_T, S_j / \lambda) \tag{4.35}$$

Given the HMM model $\lambda$, the probability of the partial observation sequence $O_1, O_2, \cdots, O_T$ and state $S_j$ at time t is $\alpha_j(t)$.

$$\alpha_j(t) = \begin{cases} 0 & t = 0 \,\&\, j \neq initial\ \ state \\ 1 & t = 0 \,\&\, j = initial\ \ state \\ [\sum_{i=1}^{N} \alpha_i(t-1) a_{ij}] b_{jk} O(t) & otherhwise \end{cases} \tag{4.36}$$

The calculation of the Forward algorithm is given as follow:

BEGIN initialize $t \leftarrow 0$, $a_{ij}$, $b_{jk}$, $O = [O_1, O_2, \cdots, O_T]$

FOR $t \leftarrow t+1$

$\alpha_j(t) \leftarrow [\sum_{i=1}^{N} \alpha_i(t-1) a_{ij}] b_{jk} O(t)$

UNTIL $t = T$

RETURN $P(O) \leftarrow a_0(T)$ for the final state

END

Similarly, the Backward algorithm, the time-reversed version of the Forward algorithm, can be considered.

The backward variable $\beta_i(t)$ can be defined as:

$$\beta_i(i) = \begin{cases} 0 & t = T \,\&\, i \neq \textit{final state} \\ 1 & t = T \,\&\, i = \textit{final state} \\ [\sum_{j=1}^{N} \beta_j(t+1) a_{ij}] b_{jk} O(t+1) & \textit{otherhwise} \end{cases} \quad (4.37)$$

The calculation of the Backward algorithm is given as follow:

BEGIN initialize $t \leftarrow T$, $a_{ij}$, $b_{jk}$, $O = [O_1, O_2, \cdots, O_T]$

FOR $t \leftarrow t - 1$

$\beta_i(t) \leftarrow [\sum_{j=1}^{N} \beta_j(t+1) a_{ij}] b_{jk} O(t+1)$

UNTIL $t = 1$

RETURN $P(O) \leftarrow \beta_i(0)$ for the initial state

END

The solution of Decoding Problem can be carried out by Viterbi algorithm [Rabiner 1989]. The Viterbi algorithm is a formal technique for finding the best state sequence and is applied as a dynamic programming method. It is actually very similar to the forward algorithm. It is likely to consider every possible path and calculate the probability of the visible sequence observed. The HMM Decoding algorithm is given as follow:

BEGIN initialize $path \leftarrow \{\}$, $t \leftarrow 0$

FOR $t \leftarrow t + 1$, $j \leftarrow j + 1$

$k = 0$, $\alpha_0 = 0$

FOR $j \leftarrow j + 1$

$\alpha_j(t) \leftarrow b_{jk} O(t) \sum_{i=1}^{N} \alpha_i(t-1) a_{ij}$

UNTIL $j = N$

$j' \leftarrow \arg\max_j \alpha_j(t)$

append to *path* $S_{j'}$

UNTIL $t = T$

RETURN *path*

END

As to the solution of Learning Problem, the parameters of the Hidden Markov Model, including probability transition matrix *A* and probability density matrix *B*, are estimated based on the Baum-Welch algorithm [Rabiner 1989].

The reasonable re-estimation formulas employed to estimate parameters $a_{ij}$ and $b_{jk}$ in the Baum-Welch algorithm for *A* and *B* are:

- $$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (4.38)$$

  where $\sum_{t=1}^{T-1} \xi_t(i,j)$ is the expected number of transitions from state $S_i$ to state $S_j$ at any time in the sequence.

  $\sum_{t=1}^{T-1} \gamma_t(i)$ is the expected number of transitions from state $S_i$ at step *t*.

- $$\bar{b}_{jk} = \frac{\sum_{t=1 \; s.t. O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (4.39)$$

  which calculates the ratio between the frequency that any particular symbol $v_k$ is observed and that for any symbol.

The algorithm for training the Hidden Markov Model operates according to the pseudo code provided below:

BEGIN initialize $a_{ij}$, $b_{jk}$, training sequence *O* and convergence criterion $\theta$

DO $z \leftarrow z+1$

compute $\bar{a}(z)$ from $\bar{a}(z-1)$ and $\bar{b}(z-1)$

compute $\bar{b}(z)$ from $\bar{a}(z-1)$ and $\bar{b}(z-1)$

$a_{ij}(z) \leftarrow \bar{a}_{ij}(z-1)$

$b_{jk}(z) \leftarrow \bar{b}_{jk}(z-1)$

UNTIL $\max_{i,j,k}[a_{ij}(z)-a_{ij}(z-1), b_{jk}(z)-b_{jk}(z-1)] < \theta$

RETURN $a_{ij} \leftarrow \bar{a}_{ij}(z)$; $b_{jk} \leftarrow \bar{b}_{jk}(z)$

END

### 4.6.2 Skill Model Construction by HMM

The state transition probability distribution matrix A and the observation signal probability distribution matrix B are good representation of the inherent characteristics of the peg-in-hole manipulation skill. The Baum-Welch algorithm, discussed in the Learning problem section, will be employed to determine the matrix A and B. One hundred groups of data, which are generated by one hundred successful peg-in-hole assembling tasks and collected from the haptic-rendered virtual environment, are used as the training data.

The HMM model $\lambda = (A, B, \pi)$ is derived from the state and the observation sequences. In this project, the manipulation skills identified by the physical contact relations of the peg-in-hole assembly and by the fuzzy clustering algorithms of the peg-in-hole assembly is employed as the state sequence and the observation sequence

respectively, and vice versa. That is when the manipulation skills are observed by the physical contact relations of the peg-in-hole assembly as observation sequence, the manipulation skills identified by Fuzzy Gustafson-Kessel algorithm and the Competitive Agglomeration algorithm will be used as state sequences respectively; when the manipulation skills are observed by the Fuzzy Gustafson-Kessel algorithm and the Competitive Agglomeration algorithm as observation sequence respectively, the manipulation skills identified by physical contact relations of the peg-in-hole assembly will be used as the state sequences.

As the training result, the state transition probability distribution matrix A and the observation signal probability distribution matrix B of four HMM models are shown in Table 4.4 and Table 4.5.

| State sequence identified by | Observation sequence observed by | Result index |
|---|---|---|
| Physical contact relations | Fuzzy Gustafson-Kessel | Result 1 |
| Physical contact relations | Competitive Agglomeration | Result 2 |
| Fuzzy Gustafson-Kessel | Physical contact relations | Result 3 |
| Competitive Agglomeration | Physical contact relations | Result 4 |

Table 4.4 HMM model training results index

| Result index | Matrix A | Matrix B |
|---|---|---|
| Result 1 | $\begin{bmatrix} 0.834 & 0.012 & 0.095 & 0.057 & 0.002 \\ 0.003 & 0.619 & 0.378 & 0 & 0 \\ 0.069 & 0.001 & 0.703 & 0.024 & 0.203 \\ 0.009 & 0 & 0.110 & 0.697 & 0.184 \\ 0.028 & 0 & 0.597 & 0.375 & 0 \end{bmatrix}$ | $\begin{bmatrix} 0.955 & 0 & 0.043 & 0.002 & 0 \\ 0.253 & 0.533 & 0.202 & 0 & 0.012 \\ 0.079 & 0 & 0.843 & 0.012 & 0.066 \\ 0.010 & 0.054 & 0.043 & 0.873 & 0.020 \\ 0 & 0 & 0.113 & 0.011 & 0.876 \end{bmatrix}$ |
| Result 2 | $\begin{bmatrix} 0.792 & 0.015 & 0.088 & 0.069 & 0.036 \\ 0.006 & 0.659 & 0.335 & 0 & 0 \\ 0.049 & 0.001 & 0.704 & 0.031 & 0.215 \\ 0.005 & 0 & 0.133 & 0.646 & 0.216 \\ 0.033 & 0 & 0.637 & 0.330 & 0 \end{bmatrix}$ | $\begin{bmatrix} 0.254 & 0.329 & 0.332 & 0.034 & 0.035 & 0.011 \\ 0.001 & 0.433 & 0.397 & 0.022 & 0.103 & 0.044 \\ 0.255 & 0.111 & 0.360 & 0.233 & 0.002 & 0.039 \\ 0.110 & 0.002 & 0.021 & 0.474 & 0.013 & 0.380 \\ 0.234 & 0.110 & 0.339 & 0.233 & 0.010 & 0.074 \end{bmatrix}$ |
| Result 3 | $\begin{bmatrix} 0.693 & 0.035 & 0.074 & 0.001 & 0.197 \\ 0.036 & 0.795 & 0.135 & 0.016 & 0.018 \\ 0.079 & 0.001 & 0.839 & 0.047 & 0.034 \\ 0.007 & 0.011 & 0.137 & 0.731 & 0.114 \\ 0.041 & 0.009 & 0.029 & 0.133 & 0.788 \end{bmatrix}$ | $\begin{bmatrix} 0.833 & 0 & 0.122 & 0.011 & 0.034 \\ 0.101 & 0.711 & 0.139 & 0.038 & 0.011 \\ 0.013 & 0.201 & 0.670 & 0.104 & 0.012 \\ 0.156 & 0 & 0.039 & 0.793 & 0.012 \\ 0.211 & 0.002 & 0.005 & 0.013 & 0.769 \end{bmatrix}$ |
| Result 4 | $\begin{bmatrix} 0.813 & 0.012 & 0.113 & 0.062 & 0 & 0 \\ 0.110 & 0.763 & 0.024 & 0.101 & 0.002 & 0 \\ 0.003 & 0.133 & 0.772 & 0.010 & 0.052 & 0.030 \\ 0.058 & 0.023 & 0.015 & 0.682 & 0.221 & 0.001 \\ 0 & 0.033 & 0.059 & 0.233 & 0.663 & 0.012 \\ 0 & 0 & 0.049 & 0.050 & 0.122 & 0.779 \end{bmatrix}$ | $\begin{bmatrix} 0.343 & 0.119 & 0.093 & 0.237 & 0.208 \\ 0.410 & 0.002 & 0.311 & 0.094 & 0.183 \\ 0.197 & 0.276 & 0.077 & 0.341 & 0.109 \\ 0.334 & 0.117 & 0.074 & 0.039 & 0.436 \\ 0.099 & 0.337 & 0.075 & 0.131 & 0.358 \\ 0.073 & 0.078 & 0.411 & 0.343 & 0.095 \end{bmatrix}$ |

Table 4.5 HMM model training results

The first two state transition probability distribution matrices A of the physical contact relations of the peg-in-hole assembly have produced similar results, which demonstrate the inherent characteristics of the peg-in-hole manipulation skill. In this case, in the course of determining the optimal next state, either matrix A of Result 1 or 2 can be used. In this project, Result 1 is preferred, as the number of distinct

observation symbols per state, *M*, is 5, while in Result 2, *M* is 6, which takes a longer time to derive the parameters of the HMM model.

The state transition diagrams, illustrating manipulation skills classified by the physical contact relations of the peg-in-hole assembly, by Fuzzy Gustafson-Kessel algorithm and by Competitive Agglomeration algorithm, are represented by the Hidden Markov chain of Figure 4.13, Figure 4.14, and Figure 4.15 respectively.

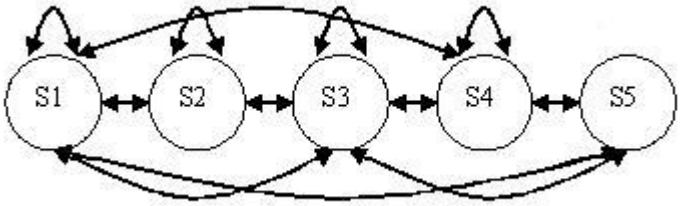



Figure 4.13 Manipulation states chain using physical contact state classification

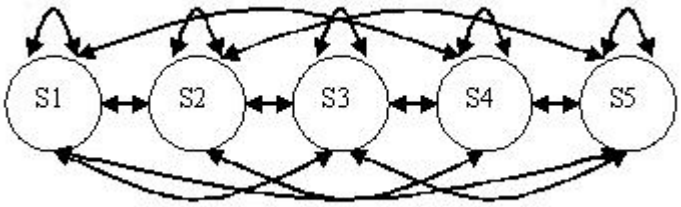



Figure 4.14 Manipulation states chain using Fuzzy Gustafson-Kessel algorithm

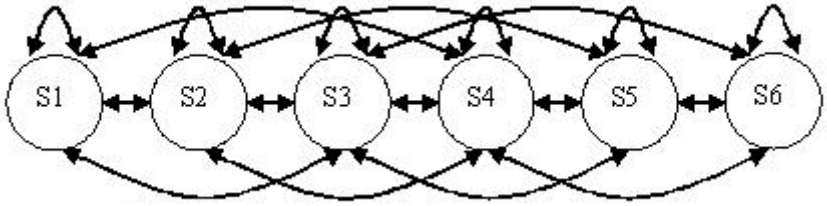



Figure 4.15 Manipulation states chain using Competitive Agglomeration algorithm

The validation of the four trained HMM models $\lambda = (A, B, \pi)$ is carried out by comparing the most likely sequences of hidden states derived from the state and the observation sequences with three different state sequences of one successful peg-in-hole assembly (showing in Figure 4.11, Figure 4.12, and Figure 4.13). The Viterbi algorithm, discussed in the decoding problem section, will be employed. Results are shown in Figure 4.16 (a), (b), and (c). It is obvious that the reconstructed state sequences are similar to the original state sequences, which has proved the four HMM models derived successfully.

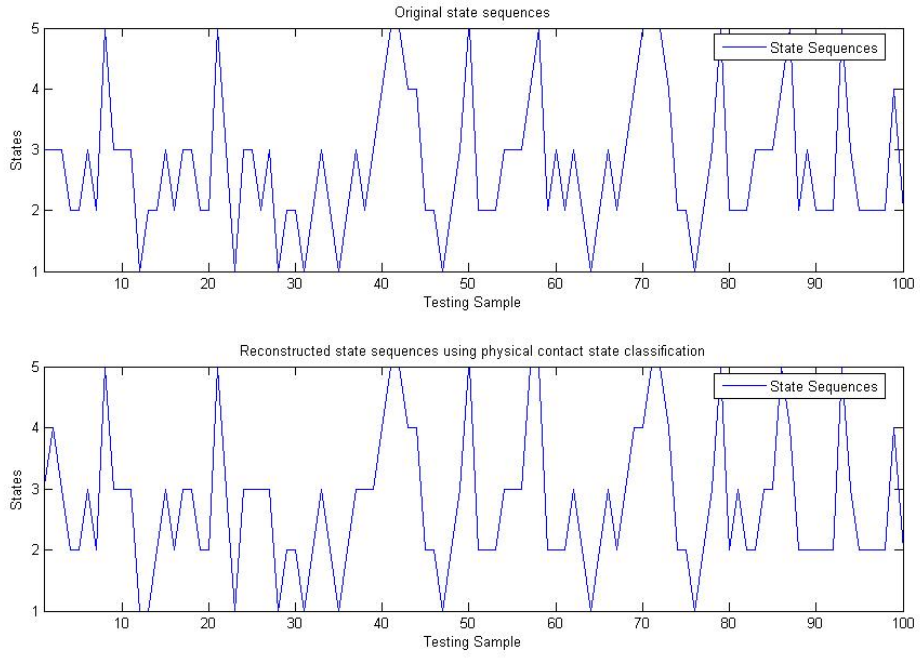



Figure 4.16 (a) Comparing the reconstructed state sequences using physical contact state with the original state sequences

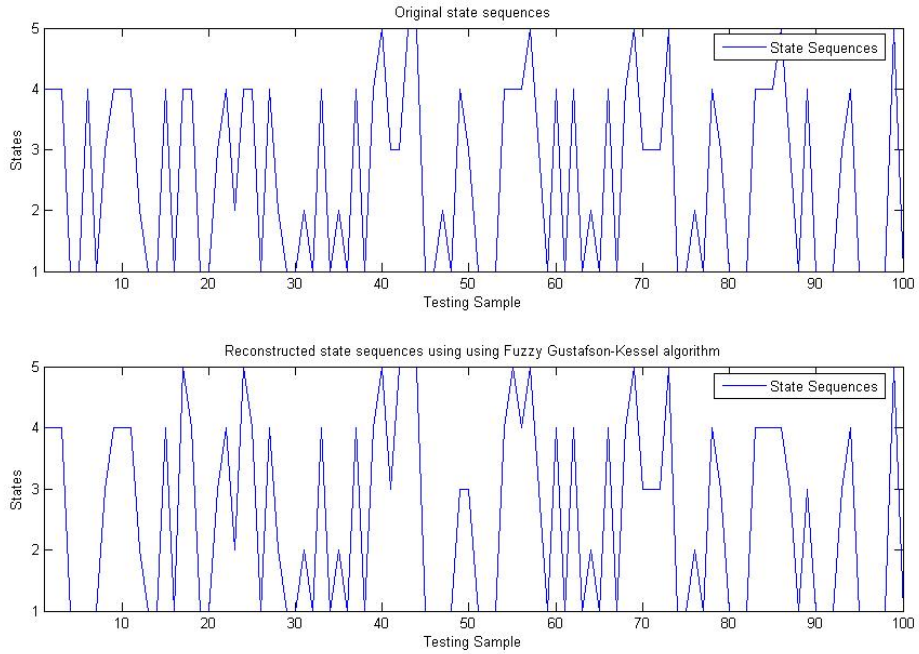



Figure 4.16 (b) Comparing the reconstructed state sequences using Fuzzy Gustafson-Kessel algorithm with the original state sequences

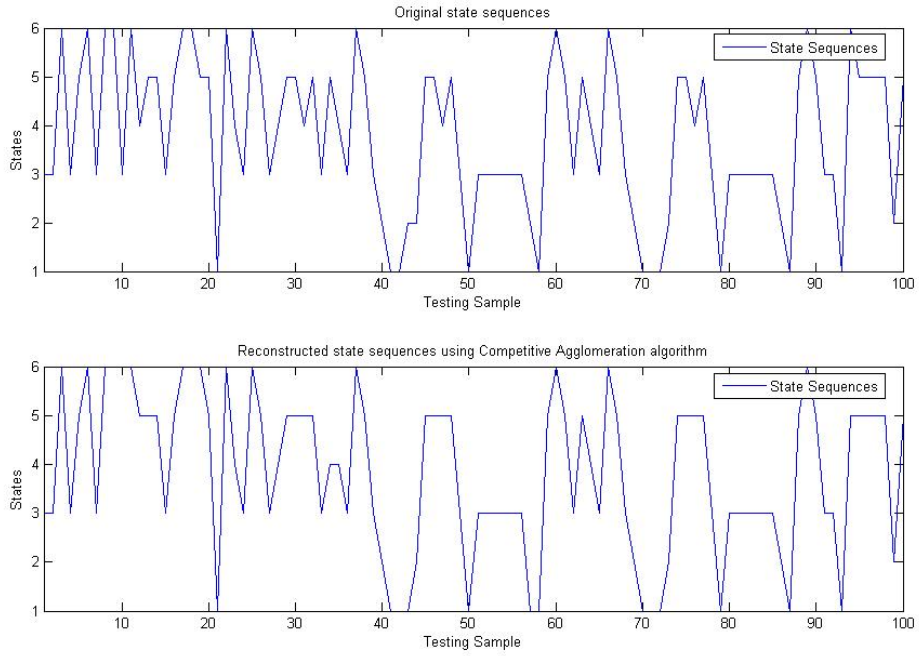



Figure 4.16 (c) Comparing the reconstructed state sequences using Competitive Agglomeration algorithm with the original state sequences

## 4.7 Trajectory Learning in Physical Experimental Rig

The Locally Weighted Regression (LWR) method is encoded as the approximator for the trajectories in each manipulation state during physical manipulation [Christopher 1997]. In most learning methods, a single global model is used to fit all the training data, while local models attempt to fit the training data only in a region around the location of the query point. Locally weighted regression uses a distance-weighted regression to fit nearby points, giving them a high relevance. Locally weighted regression is a form of lazy and memory-based learning, since it stores the training data in memory, performs a regression around a point of interest using only training data that is local to that point and finds relevant data from the database to answer a particular query point [Bratko 1995]. When a locally weighted linear model is computed, the stored data points are weighted according to the distance from the query point.

The following issues are considered when locally weighted regression learning is applied [Christopher 1997]:

1. Distance function: A typical distance function $d_m(x_q, x)$, the diagonally weighted Euclidean distance, is used to measure the relevance between the query point $x_q$ and each data point input vector $x$.

$$d_m(x_q, x) = \sqrt{\sum_j (m_j(x_{q_j} - x_j))^2} = \sqrt{(x_q - x)^T M^T M (x_q - x)} = d_E(M_x, M_q) \qquad (27)$$

$m_j$ is the feature scaling factor for the $j$th dimension.

$M$ is a diagonal matrix with $M_{jj}=m_j$.

2. Separable criterion: a general global model can be trained to minimise the weighted training criterion $E(x_q)$.

$$E(x_q) = \frac{1}{2}\sum_{x \in x_q} (f(x) - \overline{f}(x))^2 K(d(x_q, x)) \quad (28)$$

$f(x)$ is the output value corresponding to the input vector *x*.

$\overline{f}(x)$ is the global general nonlinear model.

$K(d_m(x_q, x))$ is the Gaussian kernel function, which is used to determine the weight of each training example.

$$K(d_m(x_q, x)) = \exp(-d^2{}_m(x_q, x)) \quad (29)$$

3. Enough data: sufficient data are needed to satisfy the statistical requirements.

4. Labelled data: each training data point should be linked to a specific output.

5. Representation: fixed length vectors are produced for a list of specified features.

The following LWR training algorithm is performed when a prediction is needed for a query point *q*:

BEGIN initialize $\overline{f}(x) = \omega_0 + \omega_1 a_1(x) + \cdots \omega_n a_n(x)$, $i = 0$, convergence criterion $\theta$

DO $i \leftarrow i + 1$

$$\Delta\omega_i = \eta \sum_{x \in x_q} K(d(x_q, x))(f(x) - \overline{f}(x)) a_j(x)$$

$$\omega_i(i) \leftarrow \omega_i(i-1) - \Delta\omega_i$$

compute $\overline{f}(x)$

UNTIL $E(x_q) = \frac{1}{2}\sum_{x \in x_q} (f(x) - \overline{f}(x))^2 K(d(x_q, x)) < \theta$

RETURN $\overline{f}(x)$

END

## 4.8 Summary

The focus of this chapter has been on the acquisition of the manipulation skills. An overview of classical learning methods has been provided. Offline learning methods used in the work to develop skill models including Behavioural Cloning method, Fuzzy Clustering Algorithms, and Hidden Markov Model have been introduced. The results indicate that the Fuzzy Clustering Algorithms are rather effective in obtaining manipulation skills by classifying these skills into different states, while the Hidden Markov Model is more efficient in acquiring the optimal state sequence and estimating the next optimal state. On line learning algorithm, the Locally Weighted Regression algorithm, employed in trajectory learning in each manipulation state during the physical manipulation procedure, is also introduced. Validation of this method will be further discussed in the following chapter.

# CHAPTER 5 VALIDATION

## *5.1 Introduction*

The methodologies developed in this study are validated through experimental work and the results are reported in this chapter. The physical peg-in-hole insertion rig representing a typical assembly process is explained first. Various experimental set ups designed to validate the algorithm are then described. The experimental results obtained by performing the physical peg-in-hole insertion are presented and a critical analysis of the results is finally carried out.

## *5.2 Proposed Learning Procedure*

The paradigm developed in this work to create a skill model and used in the physical experimental rig is illustrated in Figure 5.1. The peg-in-hole insertion procedure is classified into several states by the state classifier according to the peg and hole's contact relations. Different states have different online LWR learning module. Manipulation states are generalized offline by the peg and hole's physical contact relations or Fuzzy Clustering algorithms, according to overall generated manipulation data characteristics. Hidden Markov Model is used to estimate the next optimal state. Data generated from the haptic-rendered virtual environment are first applied to the one of the three offline training algorithms. Then the online state classifier is created. The more training data applied to the algorithm, the more accurate will be the state

classifier. Specific indexes are assigned to each state. The index speeds up the search process in the physical manipulation by looking for the proper LWR online learning module associated with a particular state in a sub-training data set, rather than the whole database.

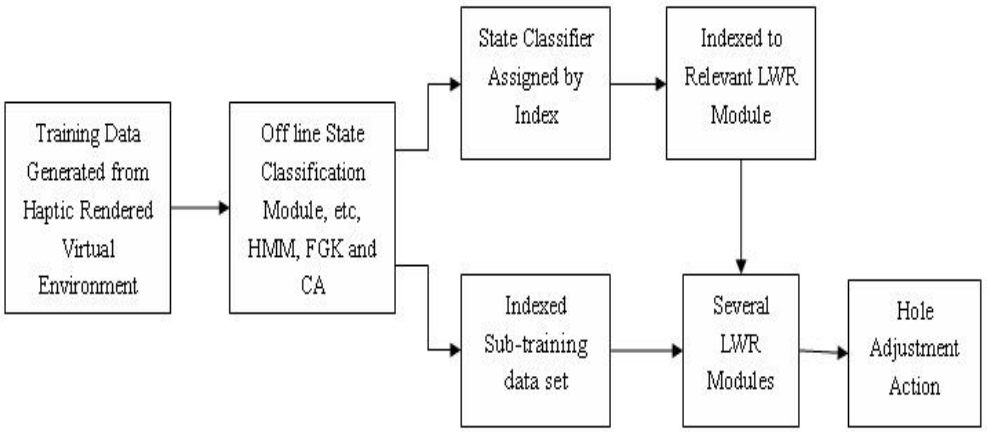



Figure 5.1 Skill model used in the physical experimental rig

## *5.3 Experimental Rig*

The physical experimental rig is the peg-in-hole insertion task which represents a typical assembly process. The algorithms of the Fuzzy Gustafson-Kessel (FGK), Competitive Agglomeration (CA), Hidden Markov Model (HMM), and Locally Weighted Regression (LWR) are applied to the experimental rig, which consists of a one degree of freedom (1 DOF) peg (the translation along the axis of the peg) and a two degrees of freedom (2 DOFs) hole (the pitch and yaw angles). This provides three degrees of freedom (3 DOFs) altogether, quite adequate to study the insertion phase of

the assembly process associated with the rig.

The peg with one degree of freedom (1 DOF) is controlled by a Baldor MTE Series DC servo motor [Baldor (online) a] and driven by a Baldor TSD Series DC servo controller [Baldor (online) b]. The DC servo motor and the controller provide precise motion for the peg. The position of the peg is represented by pulses which are generated by a 14-pin Baldor DC servo motor encoder [Baldor (online) a] when an actuator is driving the peg. A magnitude of 50 pulses of the encoder is equivalent to a displacement of 0.1 mm. The position of the peg is between 0 to 50000 pulses. The number of pulses increases with deeper penetration of the peg into the hole.

The hole with two degrees of freedom (2 DOFs) is controlled by two high-torque Sanyo Denki phase stepper motors [Sanyo (online)] and driven by two Gecko micro-step drives [Gecko (online)]. High-torque phase stepper motors are used to avoid backlash, while micro-step drives are used to control the micro-tilt motion instead of using a gearbox. The stepper motors and step drives together provide accurate tilt motion for the hole. The tilt motion is represented by steps of the two stepper motors controlling the hole. The maximum resolution of the stepper motor is 0.18 degree. The maximum absolute value of steps used in this experimental rig is 25. Hence, this gives a maximum initial angle of 4.5 degrees (25×0. 18=4. 5).

The radius of the peg and the hole are l0 mm and 10.05 mm, respectively. This gives a clearance of 0.05 mm between the peg and the hole. This peg-in-hole assembly experimental rig is used as a platform to study the concept of typical

assembly problem which concisely models a constrained motion-force-sensitive manufacturing task with all the attendant issues of jamming, tight clearances, and the need for quick assembly times, reliability, etc.

The peg is fitted with a 6 DOF Lord force/torque sensor [Lord 1986] which consists of a transducer unit and a force/torque sensing system controller unit. The force unit of this Lord force/torque sensor is uf (one uf equals to 0.056448N). The PHANToM 1.5 haptic device can produce a maximum exertable force of up to 8.5 N at a nominal position, and a continuous exertable force of 3 N at a nominal position. Hence, in this task, when a force signal exceeds 55 uf (3.1 N) during the operation, the peg is stopped and moved back 0.4 mm, waiting for the hole to be aligned with the peg by the system, and then continues. The peg-in-hole insertion is stopped when the maximum force reaches 150 uf (8.5N); the force limitation of the PHANToM 1.5. In this situation, the operation is assumed to have failed. The Lord Force/Torque sensing controller unit is connected to the Host PC through a RS-232 serial port, which is configured for a 9600 baud rate and 8N1 (8 bits, no parity, 1 stop bit) mode. Force sensory signals are read into the Host PC through Com 1 port.

The DAQ-802 data acquisition board, which is a cost-effective high-speed data-acquisition board for IBM-compatible ISA bus applications, is used to read the decoded signal generated by the decoder HCTL-2016 [Agilent (online)] connected to the encoder of the DC servo motor, send signals to the DC controller controlling the servo motor driving the peg and the stepper motors driving the hole. All of these input

and output signals are read into the Host PC by a DAQ-802 I/O interface card through a Main I/O connector and an Auxiliary connector.

The experimental rig must be initialised first. The angles of the axes of the stepper motors fitted on the hole are initialised by aligning the axis of the peg with that of the hole. This alignment can be achieved by a successful insertion performance. The input to the Baldor TSD Series DC servo controller, controlling the Baldor MTE Series DC servo motor driving the peg, should be initialised by a zero value to ensure that the data from a previous experiment is cleared. This will prevent the peg from moving while the power is being turned on. The force/torque sensing system is initialised by inputting the Bias Sensor command “BS” in order to derive force/torque information from the force/torque sensor. The “OR” command which selects the Output Record mode is issued to begin the transmission of one record of force/torque data over the serial port. The “out of sequence” errors produced during the programming, that cannot input force/torque data to the serial port during the system running, can be overcome by the initialisation of the force/torque sensor.

The block diagram of the purpose-built peg-in-hole physical experimental rig is depicted in Figure 5.2. Initially, the peg is being driven down to the hole. The position of the peg is measured by pulses sent from the encoder/decoder to the DAQ-802 data acquisition board at the same time. At 26000 pulses, the position of the peg is just above the hole, while the force/torque sensing system controller starts to monitor the force/torque by the transducer fitted on the peg and communicating with the Host PC

through the Com1 port at a 9600 baud rate. Hence, the peg can be driven faster when the position is smaller than 26000 pulses. During the peg-in-hole insertion process, if the maximum force/torque is below the threshold, the process is continued to completion. If the maximum force/torque is reached, a certain learning algorithm will be applied to help either the peg or the hole to continue performance. In this case, the peg may be moved back 0.4 mm by sending a negative signal to the Baldor TSD Series DC servo controller or the hole may be turned a certain angle to adjust its pitch angle or yaw angle. If the position of the peg is increased to more than 47000, the peg-in-hole insertion is considered to be completed successfully.

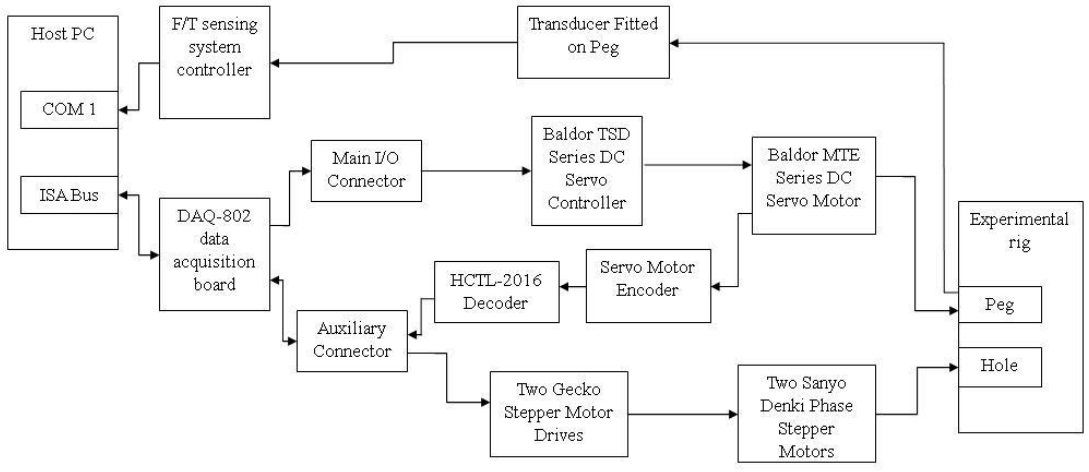



Figure 5.2 Block diagram of the purpose-built peg-in-hole physical experimental rig

## *5.4 Experimental Results*

The manipulator's behaviours are cloned offline and classified into different operation states by peg and hole's physical contact relations or Fuzzy Clustering algorithms,

according to overall generated manipulation data characteristics. Hidden Markov Model is used to estimate the next optimal state, as discussed in Chapter 4. Three skill models，based on the algorithms above, and developed to estimate different operation states, are used respectively in each experiment for the purpose of comparing the physical peg-in-hole insertion performance. The Locally Weighted Regression (LWR) algorithm described in Chapter 4 is applied to estimate online the required corrective actions in the form of the rotation angles of the hole, according to different operation states.

### 5.4.1 Typical Performance

Three typical performances of three successful peg-in-hole insertion examples, employing peg and hole's physical contact relations, Fuzzy Gustafson-Kessel (FGK) or Competitive Agglomeration (CA) algorithm as manipulation skill state classification method respectively and Hidden Markov Model (HMM) as optimal next state estimator, are illustrated in Figures 5.3, 5.4 and 5.5. In order to compare the results, the same initial angles consisting of *x*-axis and y-axis misalignments of 2.7 degrees were chosen for all the experiments.

The variation of seven normalised series of force, torque, position and state changes for the three examples are illustrated in Figures 5.2, 5.3 and 5.4. The variables *fx*, *fy*, and *fz* represent forces along *X*, *Y*, and *Z*-axis whereas *Mx* and *My* are the torques around *X* and *Y* axes. The translation along Z-axis is represented by Z.

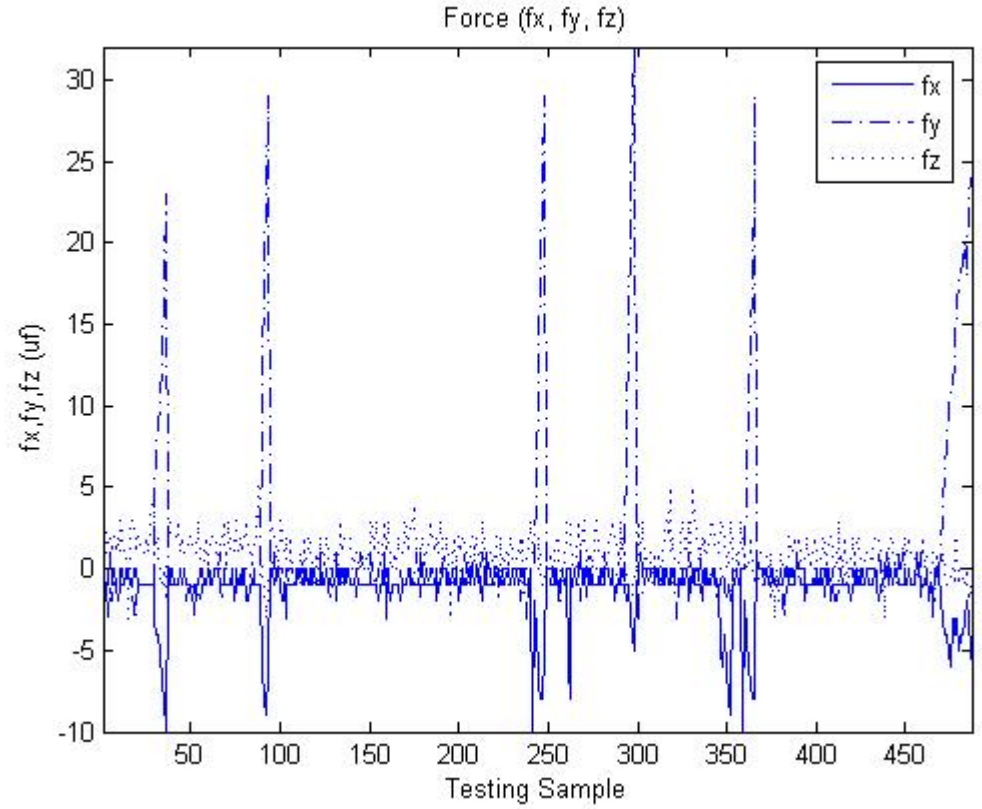
The contact state is the state change from the initial state to the goal state.

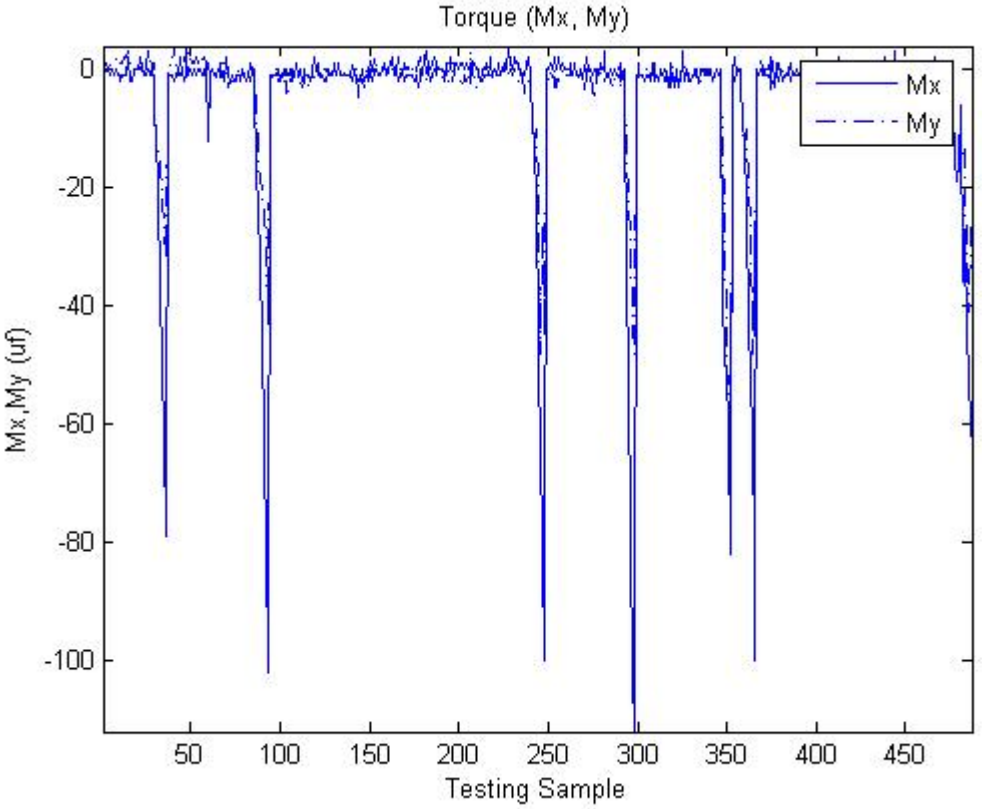
The monitoring of the peg-in-hole insertion process starts when the peg reaches the surface of the hole, which is the initial position (position z = $0.7608\times10^{4}$mm), and stops at the goal position (position z = $2.9508\times10^{4}$mm). The force/torque data are to be monitored and the position is recorded every pulse time when the position reaches the initial position, and data-acquisition interface card starts to read the signal generated by the sensor and the decoder at the same time.

The last diagram in each study shows the contact state changes from the initial state to the goal state. The peg cannot move along the X and Y axes due to the limitation on the number of degrees of freedom of the rig. Since the recording of data commences when the peg is close to the surface of the hole, some of the contact states defined in the haptic-rendered virtual environment (as described in Chapter 5) could not be monitored in the physical experimental rig.

The 6 DOF virtual peg-in-hole insertion assembly process changes its rotation angles about *X*, *Y* or *Z*-axis frequently depending on the input signals received. The trained LWR model calculates the corresponding rotation angles (steps of the two stepper motors). Similar to that in the haptic-rendered virtual environment, the LWR model generates the signal driving the two stepper motors to perform the tilting adjustment of the hole. The 3 DOF physical experimental rig is relatively simpler than that of the system in the haptic-rendered virtual environment. Hence, in the physical system, a correction signal is generated to turn the hole when the peg is encountered

with a large force or torque. Normally this occurs in the jamming state. All the experiments provided in this section have a jamming state.



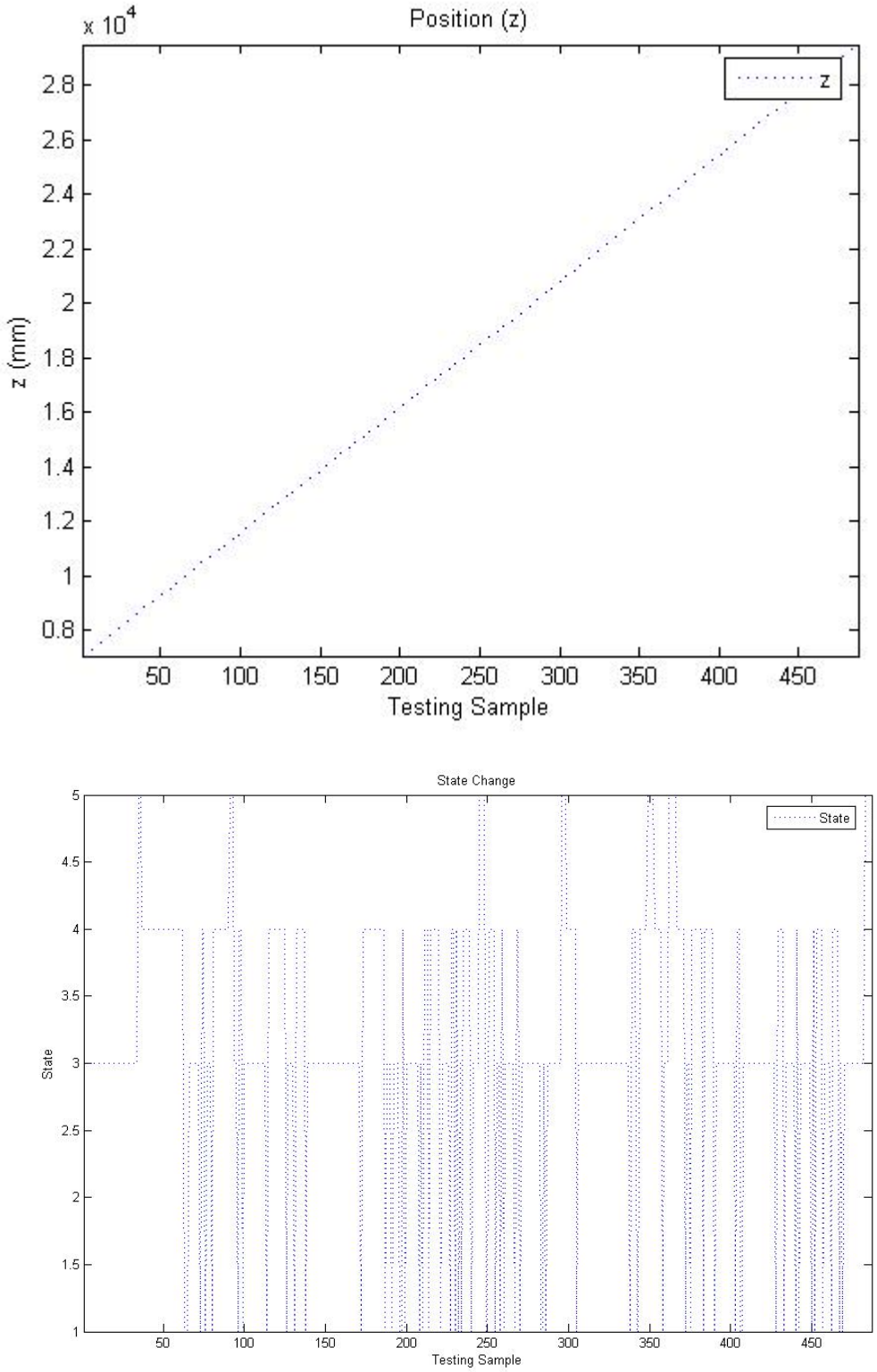
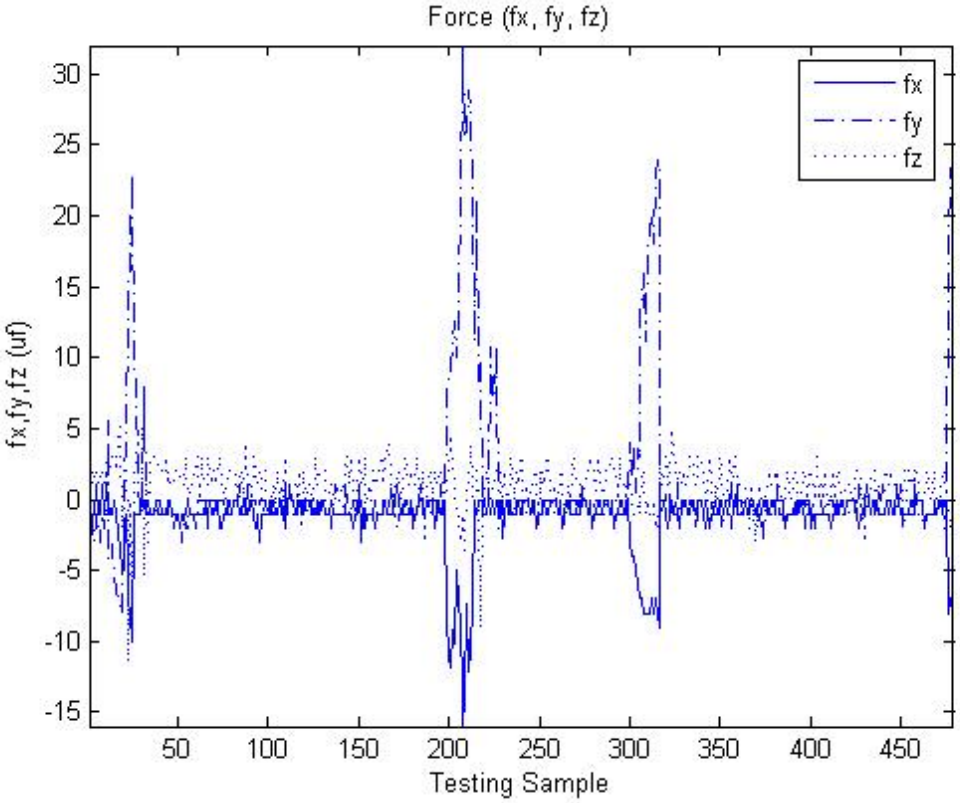
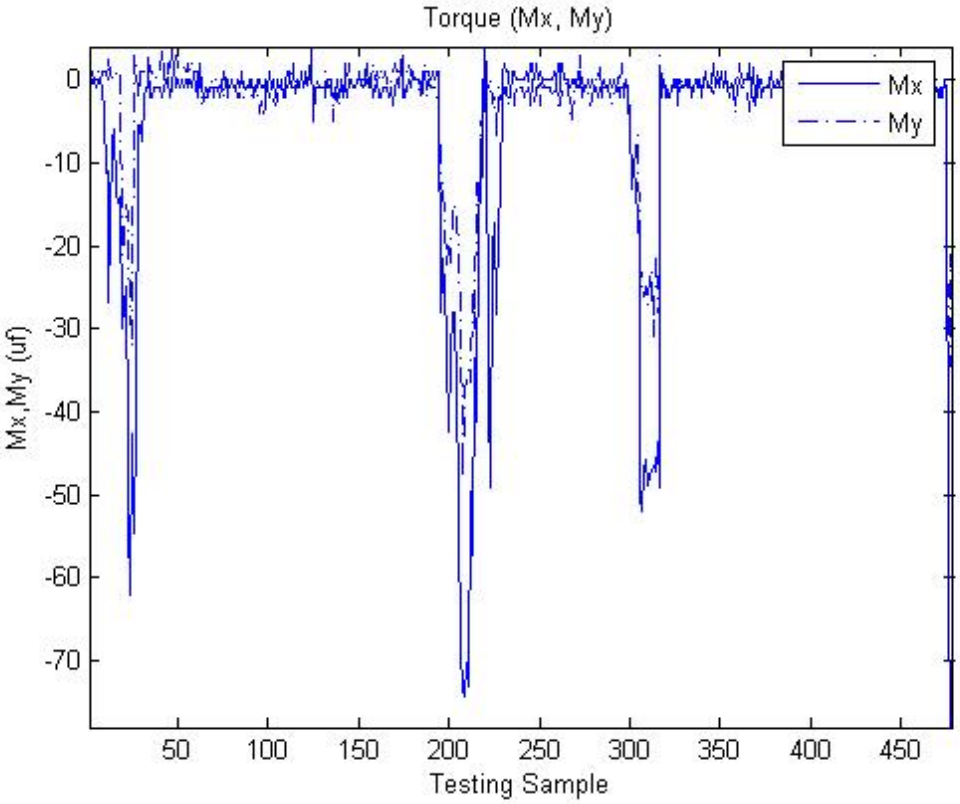



Figure 5.3 Experimental results using peg and hole's physical contact relations as state classifier


Force (fx, fy, fz)
fx
fy
fz
fx,fy,fz (uf)
30
25
20
15
10
5
0
-5
-10
-15
50
100
150
200
250
300
350
400
450
Testing Sample


Torque (Mx, My)
Mx
My
Mx,My (uf)
0
-10
-20
-30
-40
-50
-60
-70
50
100
150
200
250
300
350
400
450
Testing Sample

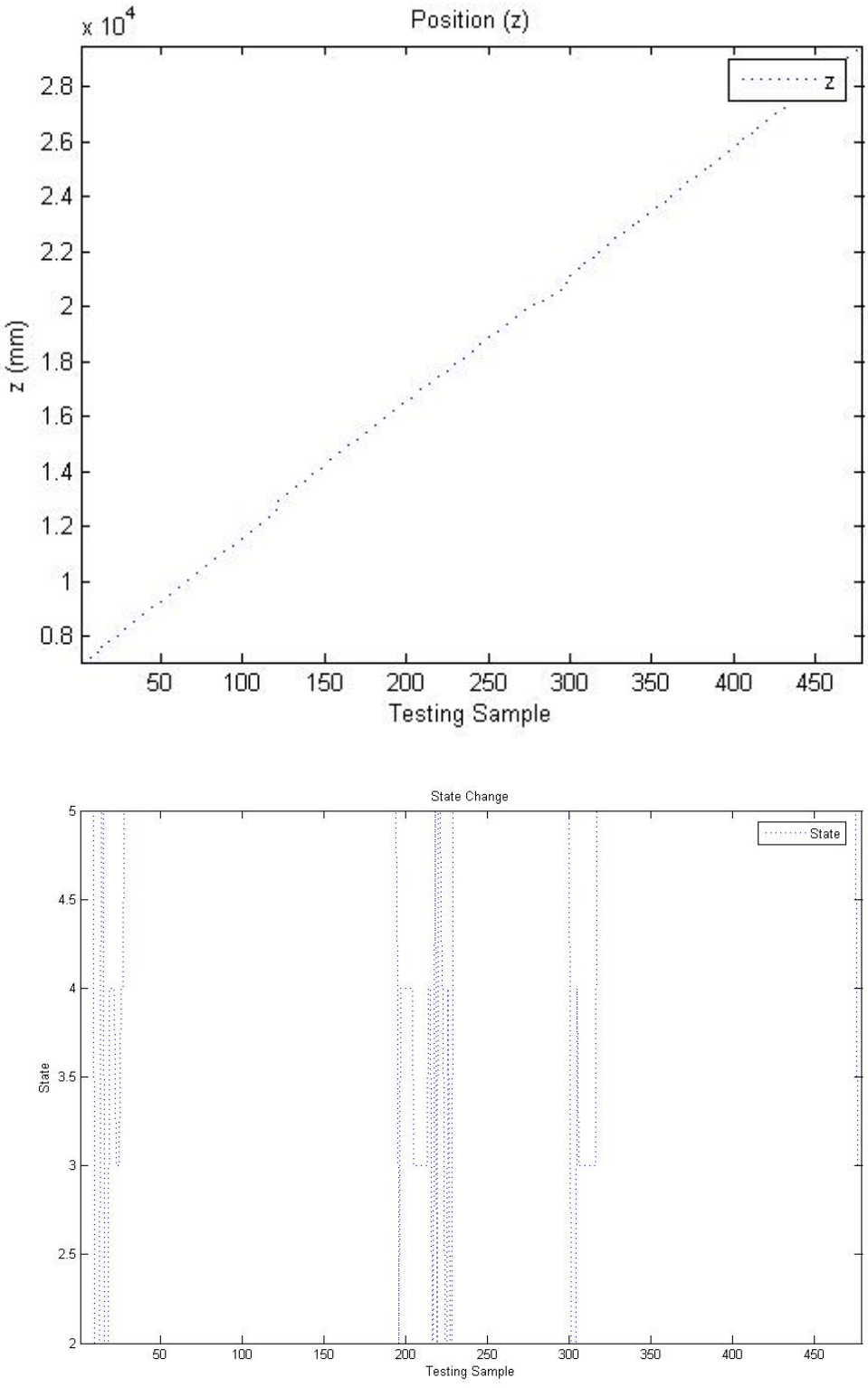
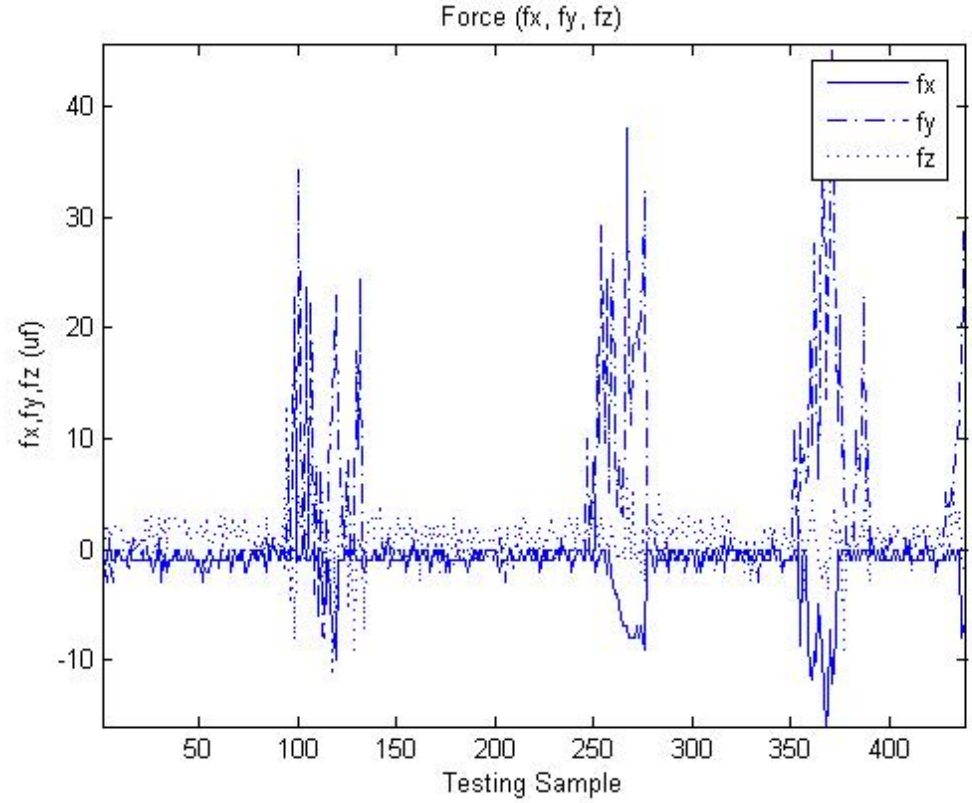
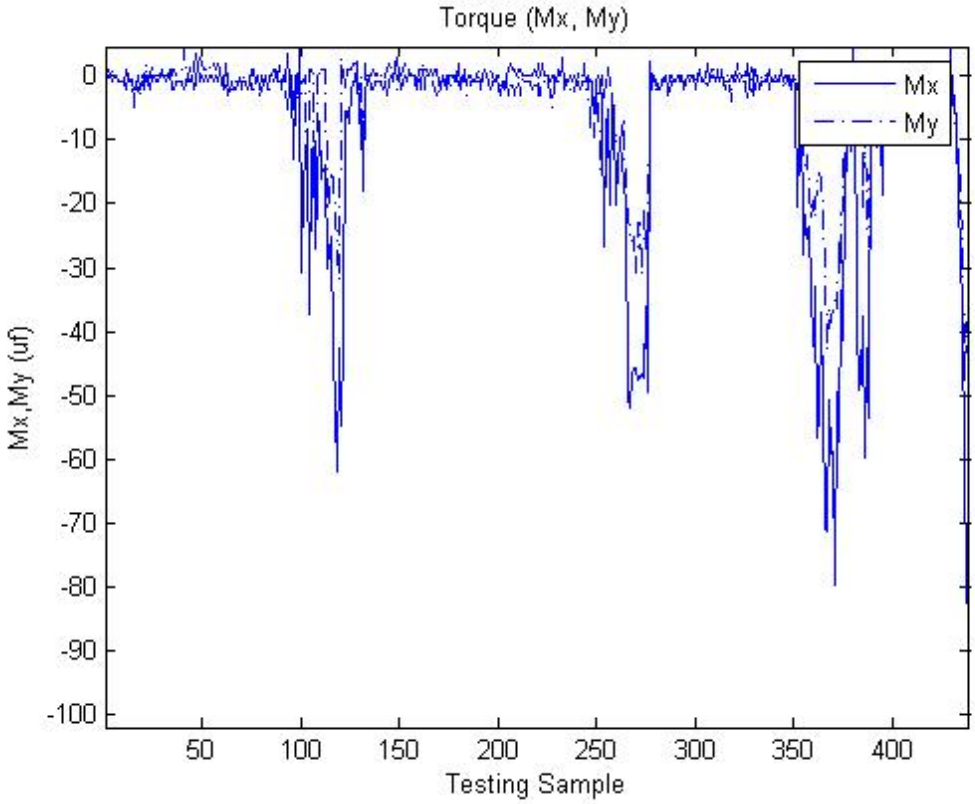



Figure 5.4 Experimental results using Fuzzy Gustafson-Kessel algorithm as state classifier


Force (fx, fy, fz)
fx
fy
fz
40
30
20
10
0
-10
fx,fy,fz (uf)
50
100
150
200
250
300
350
400
Testing Sample


Torque (Mx, My)
Mx
My
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Mx,My (uf)
50
100
150
200
250
300
350
400
Testing Sample

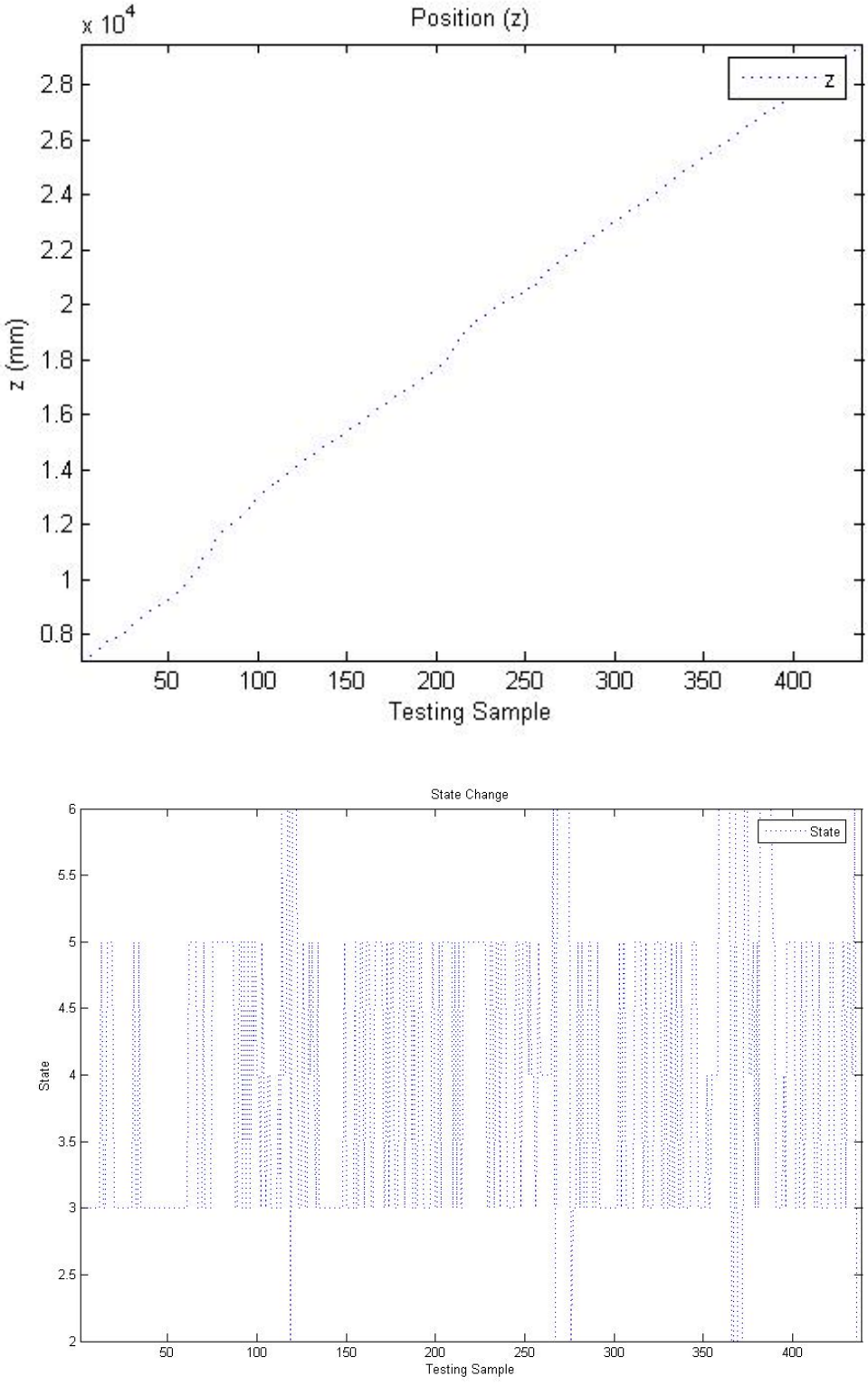



Figure 5.5 Experimental results using Competitive Agglomeration algorithm as state classifier

Observing the results obtained in these experiments, the following conclusions can be drawn:

(a) The physical experimental rig requires less time to complete the peg-in-hole insertion task when Competitive Agglomeration Algorithm is used.

(b) The physical experimental rig also has less jamming state and the hole is aligned with the peg faster when Competitive Agglomeration Algorithm is used.

(c) The peg and hole's physical contact relationships and Fuzzy Gustafson-Kessel algorithm classifies the peg-in-hole insertion process into five states, while the Competitive Agglomeration algorithm classifies the insertion process into six states. The state numbers are the optimal results when the three algorithms are used respectively.

(d) The peg-in-hole insertion process changes its contact states frequently when the peg and hole's physical contact relationships or Competitive Agglomeration algorithm is used. In this case, the online state classifier plays a less important role than that of the LWR online learning model, while the peg-in-hole insertion process changes its contact states infrequently when the Fuzzy Gustafson-Kessel algorithm is used. In this case, the online state classifier is more important than the online learning model. If the wrong state is identified, a different online learning model is chosen, and the final result could be worse.

### 5.4.2 State Classifier Comparison

In order to compare the performance of the physical experimental rig using different state classifiers and starting at different initial angles in the physical insertion, a series of studies was carried out and the results are shown in Table 5.1. The maximum resolution of the stepper motor is 0.18 degree. Hence, the initial angle of the hole can be increased by 0.18 degree. In this case, the initial angle of the hole is increased by 0.9 degree.

| Initial Angle | Algorithm | Average Testing Sample | Standard. Deviation Testing Sample | Mean Tilting Adjustment | Mean Jamming Times | Successful Rate |
|---|---|---|---|---|---|---|
| (0,0) | Physical | 478 | 5.67 | 6 | 6 | 70.89% |
| | GFK | 469 | 5.74 | 6 | 5 | 92.82% |
| | CA | 440 | 5.96 | 5 | 5 | 67.81% |
| (0,0.9) | Physical | 481 | 6.17 | 6 | 6 | 75.76% |
| | GFK | 470 | 5.62 | 7 | 5 | 86.50% |
| | CA | 440 | 6.03 | 5 | 4 | 69.25% |
| (0,1.8) | Physical | 481 | 6.14 | 8 | 5 | 71.46% |
| | GFK | 471 | 5.62 | 6 | 4 | 83.57% |
| | CA | 441 | 5.93 | 6 | 6 | 58.56% |
| (0,2.7) | Physical | 482 | 5.71 | 5 | 5 | 65.78% |
| | GFK | 472 | 5.85 | 6 | 5 | 92.92% |
| | CA | 443 | 5.80 | 5 | 4 | 58.22% |
| (0,3.6) | Physical | 482 | 5.57 | 6 | 6 | 68.12% |
| | GFK | 473 | 5.77 | 6 | 6 | 86.62% |
| | CA | 444 | 6.32 | 6 | 4 | 61.13% |
| (0,4.5) | Physical | 485 | 5.79 | 7 | 6 | 71.20% |
| | GFK | 474 | 5.91 | 7 | 5 | 91.80% |
| | CA | 443 | 6.34 | 6 | 4 | 58.25% |
| (0.9,0.9) | Physical | 479 | 5.88 | 7 | 5 | 68.90% |
| | GFK | 470 | 5.74 | 5 | 5 | 96.44% |
| | CA | 439 | 6.18 | 4 | 4 | 57.18% |
| (0.9,1.8) | Physical | 481 | 5.80 | 7 | 5 | 66.13% |

| | | | | | | |
|---|---|---|---|---|---|---|
| | GFK | 469 | 5.74 | 5 | 4 | 94.20% |
| | CA | 441 | 5.49 | 4 | 4 | 61.24% |
| (0.9,2.7) | Physical | 482 | 5.65 | 5 | 5 | 72.19% |
| | GFK | 471 | 5.91 | 6 | 4 | 93.95% |
| | CA | 442 | 6.26 | 4 | 4 | 62.02% |
| (0.9,3.6) | Physical | 483 | 5.93 | 6 | 5 | 79.60% |
| | GFK | 472 | 5.59 | 5 | 5 | 90.01% |
| | CA | 443 | 5.81 | 5 | 5 | 55.24% |
| (0.9,4.5) | Physical | 484 | 6.07 | 8 | 7 | 71.83% |
| | GFK | 474 | 5.92 | 7 | 6 | 88.82% |
| | CA | 444 | 5.75 | 4 | 4 | 61.92% |
| (1.8,1.8) | Physical | 481 | 5.70 | 7 | 6 | 71.92% |
| | GFK | 470 | 5.60 | 5 | 5 | 93.17% |
| | CA | 440 | 5.57 | 6 | 5 | 67.59% |
| (1.8,2.7) | Physical | 482 | 5.89 | 6 | 6 | 65.38% |
| | GFK | 471 | 5.45 | 7 | 5 | 94.05% |
| | CA | 441 | 5.77 | 5 | 5 | 60.76% |
| (1.8,3.6) | Physical | 482 | 5.88 | 5 | 5 | 75.02% |
| | GFK | 472 | 5.69 | 5 | 5 | 98.85% |
| | CA | 442 | 5.91 | 5 | 4 | 67.54% |
| (1.8,4.5) | Physical | 483 | 5.56 | 6 | 6 | 73.79% |
| | GFK | 474 | 6.32 | 6 | 5 | 98.59% |
| | CA | 443 | 5.46 | 5 | 5 | 58.81% |
| (2.7,2.7) | Physical | 481 | 5.63 | 5 | 5 | 74.47% |
| | GFK | 470 | 5.93 | 5 | 5 | 97.41% |
| | CA | 440 | 5.85 | 5 | 5 | 62.79% |
| (2.7,3.6) | Physical | 481 | 5.66 | 6 | 6 | 66.94% |
| | GFK | 472 | 5.51 | 7 | 6 | 91.14% |
| | CA | 442 | 5.75 | 4 | 4 | 59.14% |
| (2.7,4.5) | Physical | 484 | 5.54 | 5 | 5 | 70.57% |
| | GFK | 472 | 5.45 | 7 | 5 | 88.92% |
| | CA | 442 | 5.58 | 4 | 4 | 56.17% |
| (3.6,3.6) | Physical | 482 | 5.60 | 5 | 5 | 71.44% |
| | GFK | 471 | 5.39 | 7 | 5 | 98.15% |
| | CA | 440 | 5.59 | 6 | 4 | 67.39% |
| (3.6,4.5) | Physical | 482 | 5.82 | 6 | 6 | 66.13% |
| | GFK | 471 | 5.71 | 5 | 5 | 95.62% |
| | CA | 442 | 5.93 | 6 | 6 | 60.30% |
| (4.5,4.5) | Physical | 482 | 5.71 | 7 | 6 | 70.59% |
| | GFK | 472 | 6.04 | 6 | 4 | 90.19% |

| | | | | | | |
|---|---|---|---|---|---|---|
| | CA | 441 | 5.87 | 5 | 4 | 67.24% |

Table 5.1 Statistical results of the physical insertion performance

Comparisons of the results produced by the three algorithms are shown in a series of diagrams. The average runtimes for different initial states and their standard deviations are shown in Figures 5.6 and 5.7, respectively. Figure 5.8 shows the comparison of the average tilting adjustment times in different initial states. Figure 5.9 presents the comparison of the average jamming times. Comparison of the successful rates of the peg-in-hole insertion performance using one of the three algorithms is illustrated in Figure 5.10.

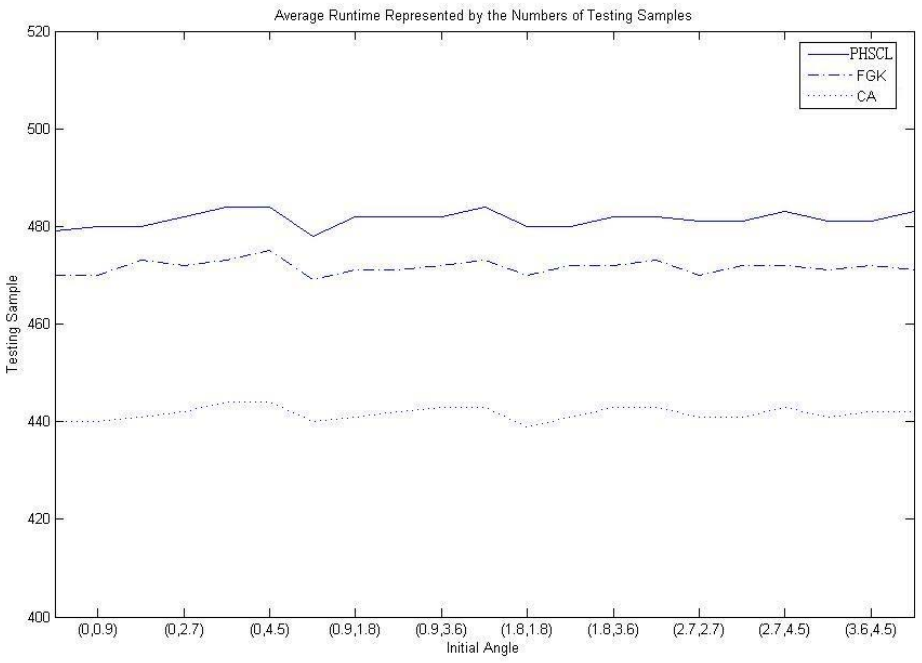



Figure 5.6 Average runtimes in different initial states

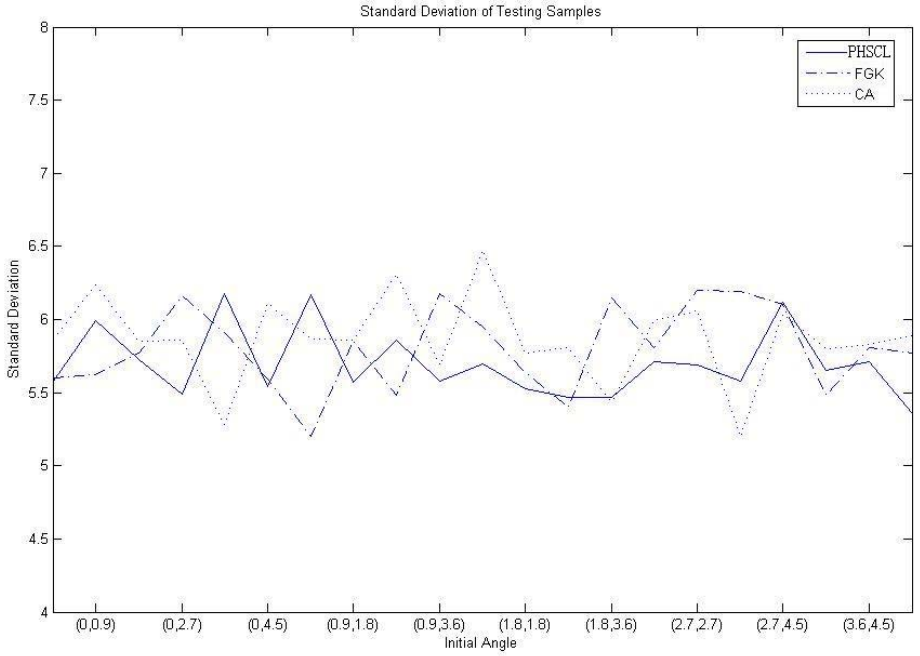



Figure 5.7 Standard deviation of the average runtimes

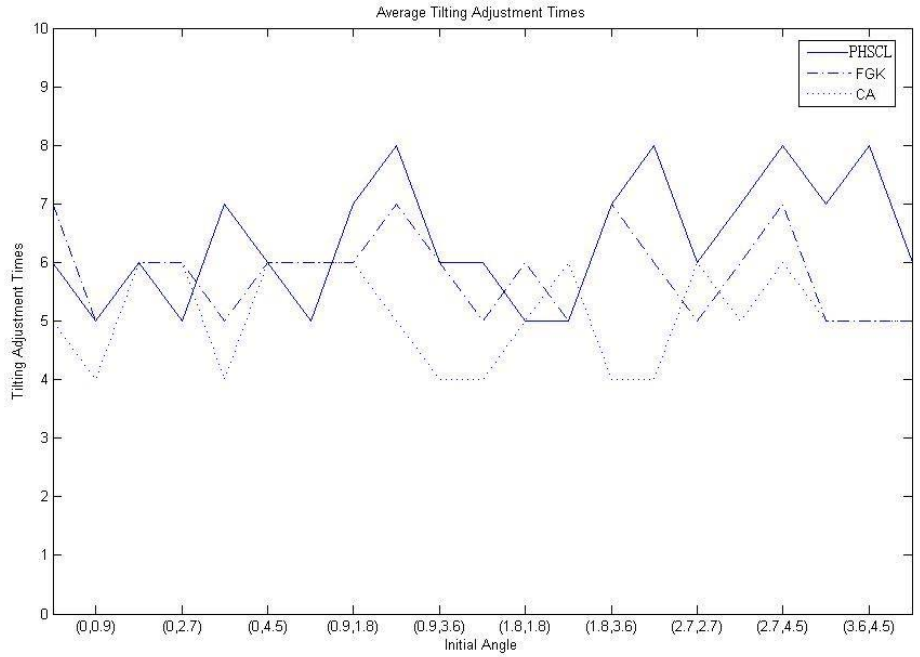



Figure 5.8 Average tilting adjustment times in different initial states

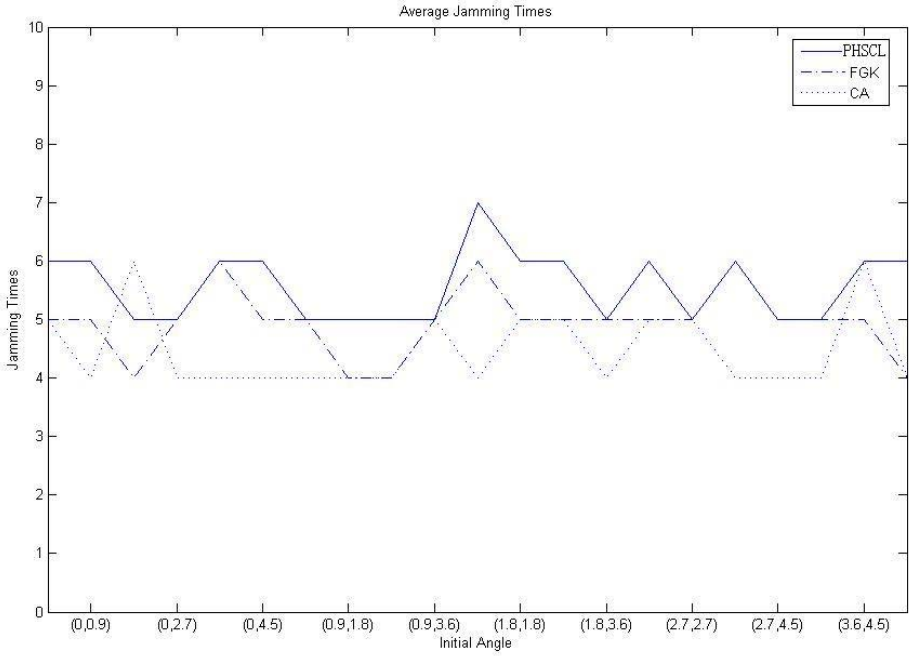



Figure 5.9 Average jamming times in different initial states

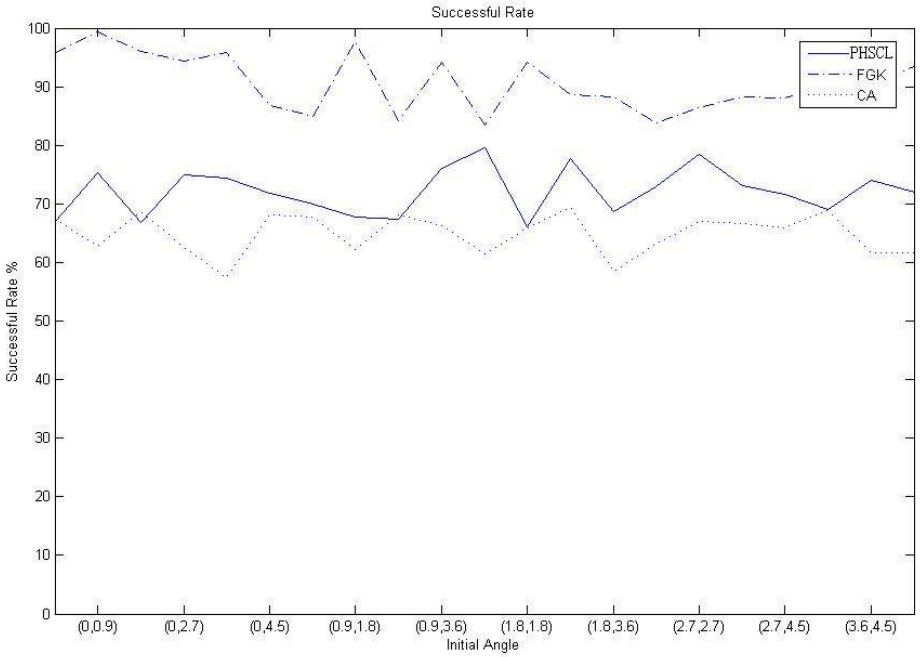



Figure 5.10 Successful rates of the peg-in-hole insertion performance

The results shown above have confirmed the conclusions drawn in Section 5.4.2. In addition, further observations can be made:

(a) The actual insertion without the deployment of skills obtained from the haptic-rendered virtual environment has been discussed in Y. Chen's Ph.D. thesis [Chen 2005]. The results showed that the average runtimes in different initial states are 4 time longer than that of deployment of skills obtained from the haptic-rendered virtual environment because of very high average tilting adjustment times and average jamming times. In this case, the success rate of the peg-in-hole insertion performance was also low.

(b) Considering the success rate of the peg-in-hole insertion performance, the experiments using the Fuzzy Gustafson-Kessel (FGK) algorithm as the state classifier are the most satisfactory.

(c) Considering the insertion time, using the Competitive Agglomeration (CA) algorithm as the state classifier results in the fastest performance. The success rate of the experiments is however relatively low.

(d) As described in Chapter 4, the peg and hole's physical contact relations classifier is determined by contact relationship between the peg and the hole, and the state number is predefined. While the Fuzzy Gustafson-Kessel (FGK) or Competitive Agglomeration (CA) algorithm can classify the overall performance into every contact state, a specific approach should be used to automatically determine the number of clusters. In this experiment, the fuzzy

clustering algorithms, which automatically determine the optimal number of clusters, performs better.

(e) The overall performance of the peg-in-hole insertion, in terms of all the manipulation processes, depends very much on which algorithms is used. In complicated manipulation tasks, correctly identifying the right state in order to choose the correct online learning is more important. This will expend less running time. However, a considerable amount of offline training must be performed.

(f) In addition, choosing the correct offline state training algorithm and online learning algorithm will improve the overall performance and avoid expending considerable amount of time.

## *5.5 Summary*

The overall physical peg-in-hole insertion experimental rig has been described in detail in this chapter. The performance of the physical experimental rig has been described. The experimental validation by employing various algorithms has been provided by illustrating them in tables and figures. Significant discussions have followed.

# CHAPTER 6 CONCLUSION AND FURTHER RESEARCH

## *6.1 Overview*

Teaching manipulation skills to a robot by cloning human manipulation habits from a haptic-rendered virtual environment has been reported. The research conducted to explore the feasibility of such approach has been explained. The offline and online learning algorithms proposed to carry out the training and learning process have been presented. The peg-in-hole insertion which represents a typical assembly process has been used as a case in the study. The effectiveness of the developed strategy has been demonstrated and validated through a series of experiments carried out on the peg-in-hole insertion rig. The results obtained are quite encouraging and clearly highlight the strengths and weaknesses of the approach.

## *6.2 Contributions of the Research*

The work conducted in this thesis has made a number of contributions as described in the following sections.

### 6.2.1 Virtual Manipulation through 6 DOFs

In the first implementation of the virtual manipulation environment, PHANToM Premium 1.0, a three degrees of freedom (3 DOFs) haptic device was used. The touch-enabled applications were developed based on GHOST SDK (General Haptic

Open Software Toolkit), a powerful C++ software tool kit, which accompanies PHANToM.

The graphical model of the assembly [Chen 2002a] was constructed using OpenGL, whereas its physical model and the force/torque vectors generated in the virtual manipulation environment were modelled based on the two different approaches of PointShell and TriPolyMesh [Chen 2002b]. The developed system proved quite stable when the peg-in-hole insertion was performed using 3DOF PHANToM.

In order to improve the performance of the developed system, the haptic device was upgraded to a six-degree of freedom PHANToM Premium 1.5. In complex applications in which simulation of an arbitrary object to object interaction is required, a six degrees of freedom (6DOF) haptic device can be a more effective tool. Applying the 6DOF haptic device to the developed haptic rendered virtual environment resulted in strong oscillation occurring during virtual peg-in-hole insertion, and preventing a successful insertion.

Further investigation of the problem revealed that the PointShell and TriPolyMesh algorithms used in the model were not sufficiently accurate for operation with a 6DOF haptic device.

In order to stabilise the virtual peg-in-hole insertion with tight fit for a 6DOF haptic device, three new more precise haptic-rendered models were developed. They included a modified PointShell algorithm, modified TriPolyMesh algorithm and

dual-gstCylinder algorithm. They stabilised the virtual insertion process and removed the oscillation observed in the system when the 3DOF haptic device was used.

### 6.2.2 Proposed Physical Model

The peg-in-hole assembly physical model based on three approaches of Modified TriPolyMesh Method, Dual-gstCylinder Method and PointShell Method suffered from a number of shortcomings. As the point on the peg's penetrated into the surface of the hole, internal forces were generated to halt the peg advance through the hole's surface. However, due to insufficient internal volume modelled for the surface, the generated reactive force was inadequate. Moreover, the point on the peg penetrating one side of the hole-surface was too close to the other side of the hole surface. As a result, the generated force would eventually push the point off the other side of the hole.

The virtual proxy method was used to overcome these shortcomings [Petersik 2002]. The virtual rigid peg was defined as a virtual proxy and was controlled by the physical ReachIn probe in the haptic-rendered virtual environment. The position of the virtual proxy was changed according to alteration in the probe's position. The force and torque reacting to the peg were transferred to PHANToM Premium 1.5 through the spring damper system. The hole was static in the environment while the peg could be translated and rotated.

This approach overcame the challenges faced in developing the physical model of the virtual environment. It also proved to be a simple and adaptable method,

sufficiently generic for modelling other applications in the virtual environment.

### 6.2.3 Concurrent Haptic and Geometric Modelling

In the first virtual model of the peg-in-hole insertion process for the 6 DOFs device, the graphic and haptic models were created using OpenGL and GHOST SDK respectively. The process proved to be too complex and elaborate.

The second implementation of the model was carried out based on ReachIn hardware platform and application programming interface from ReachIn Technologies [ReachIn online]. CrystalEyes shutter glasses from the Stereo Graphics Corporation [Stereo online] were connected to a stereo-output graphics card to provide stereo vision. By using this hand-immersive hardware platform, the peg-in-hole scene graph, haptic system and communication system between graphics and haptics were integrated in a consistent, seamless way.

This approach presented a new platform for concurrent haptic and geometric modelling of the virtual environment and significantly simplified the model and the development process.

### 6.2.4 Skill-acquisition Update

Three behavioural cloning methods including Fuzzy Gustafson-Kessel (FGK) algorithm, Competitive Agglomeration (CA) algorithm, and Hidden Markov Model (HMM) were explored in the study as possible skill acquisition methods.

The peg-in-hole insertion procedure was classified into several states by the state classifier, according to the peg and hole's contact relations. Different states had different online LWR learning modules. The acquisition of the peg-in-hole insertion skills was primarily based on behavioural cloning methods. Manipulation states were generalized offline by the peg and hole's physical contact relations or statistical methods such as Fuzzy Gustafson-Kessel (FGK) algorithm, and Competitive Agglomeration (CA) algorithm. Hidden Markov Model was used to estimate the next optimal state and the optimal state sequence. Data generated from haptic-rendered virtual environment were first applied to one of the three offline training algorithms. Then the online state classifier was created by the Locally Weighted Regression (LWR) method. The accuracy of the state classified proved to improve with deployment of more training data in the development of the algorithm. The inclusion of larger number of training data in the algorithm resulted in more accurate state classifier. Specific indexes were assigned to each state. This increased the search speed during physical manipulation by looking for the proper LWR online learning module associated with a particular state in a sub-training data set, rather than the whole database.

## *6.3 Further Research*

The research results described in this thesis can be further extended in a number of directions. Details are described in the following sections.

### 6.3.1 Physical Experimental Rig

The haptic-rendered virtual peg-in-hole environment has six degrees of freedom, which is designed to simulate real situation, while the current peg-in-hole physical experimental rig has three degrees of freedom, which can only provide a one-degree of freedom peg (the translation along the axis of the peg) and a two-degrees of freedom hole (the pitch and yaw angles). More degrees of freedom are suggested to be added to the physical experimental rig in further research.

The robot SCORBOT-ER 4u or its latest version, manufactured by Intelitek, is a good hardware experimental platform [Intelitek (online)]. The robot SCORBOT-ER 4u has five degrees of freedom (5DOF) (five rotational axes + gripper), in which it doesn't have yaw angle freedom. However, the hole has yaw angle freedom which can provide an additional one degree of freedom. The robot can be programmed to perform the task of picking up the peg and inserting it in the hole fixed on the work bench.

The current study was carried on based on cylindrical peg and hole in which the roll angle around the axis of the peg is void. This physical experimental rig can be replaced by cuboids peg and hole to which will require the control of the roll angle as well.

### 6.3.2 Haptic-rendered Virtual Environment

The current virtual manipulation environment is implemented by the ReachIn hardware platform and application programming interface. One PHANToM 1.5 haptic device is used in the research. The ReachIn hardware can provide complex, high-quality haptic feedback through one PHANToM haptic device or two PHANToM haptic devices' cooperation. Hence, another PHANToM haptic device can be used in a future research study to simulate two hands on one robot or two robots cooperating. The PHANToM 1.5 haptic device can produce a maximum exertable force of up to 8.5 N at a nominal position, and a continuous exertable force of 3 N at a nominal position. Hence, the maximum excertable force in two PHANToM haptic devices' cooperation can be doubled.

The construction of cuboid peg-in-hole haptic-rendered virtual environment is suggested to be the next research step, because it is a more generic model, and involves the freedom of the roll angle. The current haptic-rendering algorithm, the virtual proxy algorithm, can still be used as the major rendering algorithm. As to more complex haptic models, Voxmap PointShell$^{TM}$ (VPS$^{TM}$) software can be used to solve certain difficult geometry-related computing problems faster and more efficiently..

### 6.3.3 Machine Learning Algorithm

The skill-acquisition models employed in this study are based on the idea of cloning human behaviour and manipulation trajectories. Traditional behaviour cloning, such

as the decision tree method, has been used in many research projects but has proven to have low efficiency. Statistical analysis algorithms, such as the Hidden Markov Model, Fuzzy Clustering, Competitive Agglomeration and Locally Weighted Regression algorithms have been used in this project and have performed well. The Hidden Markov Model, Fuzzy Clustering and Competitive Agglomeration have been used to train highly efficiency state classifiers, while Locally Weighted Regression has been used to clone trajectories of human habits in each manipulation state.

In the future research, the feasibility of designing of a more systematic approach for classification of the contact states obtained from the haptic rendered virtual manipulation model and estimation of the assembly states during physical manipulation should be explored. In the current approach, the fuzzy state classification algorithms classify the training data into different clusters. Depending on different classification algorithm, different cluster shape, cluster size, and cluster number are produced. A large amount of training data was used in this project. The more training data employed, the more accurate the training result. However, the training time will be much longer. 14 dimensional spaces, including the force, torque, and position data, were recorded in this project. In the future research, the more complex the assembly task perform the higher dimension may be recorded. This will result of positive or negative dimension of the training data are recorded. Further analysis of the high dimension data would be, for example, leaving the positive dimension unchanged, doubling or halving the negative dimension weights to give the

training results less mistake. In this case, an effective algorithm employed to remove unhelpful dimensions should be developed in the next step in the further research.

Hidden Markov Model is used to estimate the next optimal state and the optimal state sequence. A fully connected HMM chain will be constructed, no matter how complex the assembly process is, but arbitrary state transitions may be recorded in the more complex task in the further research project. Some similar algorithm, derived from Hidden Markov Model can be employed. For example, Layered hidden Markov model (LHHM), it is often useful to constrain the model by not allowing arbitrary state transitions. In the same way it can be beneficial to embed the HMM in a layered structure which, theoretically, may not be able to solve any problems the basic HMM cannot, but can solve some problems more efficiently because less training data is needed. It is sometimes useful to use HMMs in specific structures in order to facilitate learning and generalization [Oliver 2004].

### 6.3.4 Generalisation

The work at this stage is focused on the peg-in-hole insertion process. The algorithms and methodologies developed for this application should be expanded to include different constraint motion manipulations in the next stage of the project. This will assist in developing a more generic approach.

# Reference


[Adms 1971] J. Adams, “A Closed-loop Theory of Motor Learning,” Journal of Motor Behaviour, vol. 3, no. 2, pp.111－149, 1971.

[Agilent (online)] Agilent Technologies Innovating the HP Way, “Quadratre Decoder/Counter Interface ICs,” URL: http://groups.csail.mit.edu/drl/courses/cs54-2001s/pdf/hctl2000.pdf.

[Aleotti 2004] J. Aleotti, A. Skoglund and T. Duckett, “Teaching Position Control of a Robot Arm by Demonstration with a Wearable Input Device,” Proceedings of International Conference on Intelligent Manipulation and Grasping, Genoa, Italy, July 1－2, 2004.

[Anderson 1982] I. R. Anderson, “Acquisition of Cognitive Skill,” Psychological Review, vol. 89, pp. 369－406, 1982.

[Avila 1996] R. S. Avila and L. M. Sobierajski, “A Haptic Interaction Method for Volume Visualization,” Proceedings of the 7th IEEE Conference on Visualization, pp. 197－204, San Francisco, USA, 1996.

[Babuska 1995] R. Babuska and H. B. Verbruggen, “New Approach to Constructing Fuzzy Relational Models From Data”, Proceedings of the 3rd European Congress on Intelligent Techniques and Soft Computing EUFIT’95, pp. 583－587, Aschen, Germany, 1995.

[Babuska 1998] R. Babuska, Fuzzy Modeling for Control, Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.

[Bain 1999] M. Bain and C. Sammut, A framework for behavioral cloning, Machine Intelligence, vol. 15, K. Furukawa, D. Michie, and S. Muggleton, Eds, Oxford, U.K.: Oxford Univ. Press, 1999.

[Baldor (online) a] Baldor MTE DC Servo Moytor, URL: http://www.baldor.com/products/servomotors/dc_servomotor/dc_servo.asp.

[Baldor (online) b] Baldor TSD DC Servo Controller, URL: http://www.baldor.com/products/motioncontrol/tsd.asp.

[Bertsekas 1987] D. P. Bertsekas, Dynamic Programming: Deterministic and Stochastic Models, Prentic Hall, Englewood Cliffs, 1987.

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

[Billard 1998] A. Billard, K. Dautenhahn, and G. Hayes, “Experiments on Human－robot Communication with Robota, an Imitative Learning and Communication Doll Robot,” Workshop, Socially

Situated Intelligence at SAB98 conference, Zurich, Technical Report of Centre for Policy Modelling, Manchester Metropolitan University, 1998.

[Billard 1999] A. Billard and G. Hayes, “DRAMA, a Connectionist Architecture for Control and Learning in Autonomous Robots,” Adaptive Behavior Journal, Vol. 7, no. 1, pp. 35－64, January 1999.

[Billard 2001a] A. Billard and M. Mataric, “Learning Human Arm Movements by Imitation: Evaluation of a Biologically－inspired Connectionist Architecture,” Robotics & Autonomous Systems 941, pp. 1－16, 2001.

[Billard 2001b] A. Billard and S. Schaal, “Robust Learning of Arm Trajectories through Human Demonstration,” Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 734－739, Maui, Hawaii, USA, Oct. 29 － Nov.2 2001.

[Blume 1983] C. Blume and W. Jakob, Programmiersprachen für Industrieroboter. Vogel-Verlag, 1983.

[Bratko 1995] I. Bratko, T. Urbancic, and C. Sammut, “Behavioural Cloning: Phenomena, Results and Problems,” Proceedings of 5th IFAC Symposium on Automated Systems Based on Human Skill. Berlin, Germany, 1995.

[Bratko 1998] I. Bratko, T. Urbancic, and C. Sammut, “Behavioral Cloning of Control Skill,” Journal of Machine Learning and Data Mining: Methods and Applications, R. S. Michalski, I. Bratko, and M. Kubat, Eds, Chicester, U.K.: Wiley, 1998, pp. 335－351.

[Burdea 2000] G. C. Burdea, “Haptics Issues in Virtual Environments,” Proceedings of the International Conference on Computer Graphics, pp. 295－302, Geneva, Switzerland, June 19－24, 2000.

[Caine 1989] M. E. Caine, T. Lozanno-Perez, and W. P. Seering, “Assembly Strategies for chamferless parts”, Proceedings of IEEE International Conference on Robotics and Automation, pp. 472－477, Scottsdale AZ, May 1989.

[Camacho 1995] R. Camacho, “Using Machine Learning to Extract Models of Human Control Skill,” Proceedings of AIT’ 95, Czech Republic, 1995.

[Chen 1998] J. R. Chen and B. J. McCarragher, “Robot Programming by Demonstration － Selecting Optimal Event Paths,” Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 518－523, May 16－20, 1998.

[Chen 2000a] J. R. Chen and B. J. McCarragher “Programming by Demonstration － Constructing Task Level Plans in Hybrid Dynamic Framework,” Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, pp. 1402－1407, April 24－28, 2000.

[Chen 2000b] S. Chen and J. Weng “State-based SHOSLIF for Indoor Visual Navigation,” IEEE Transactions on Neural Networks, vol. 11, Issue 6, pp. 1300－1314 Nov. 2000.

[Chen 2001] J. R. Chen, A. Zelinsky, “Programming by Demonstration: Removing Sub-optimal Actions in a Partially Known Configuration Space,” Proceedings of IEEE International Conference on Robotics and Automation, vol. 4, pp. 4096－4103, 2001.

[Chen 2002a] Y. Chen and F. Naghdy, “Teaching Manipulation Skills to a Robot through a Haptic Rendered Virtual. Environment,” Journal of Advanced Manufacturing Systems (IJAMS), vol. 1, no. 1, pp. 89－106, June 2002.

[Chen 2002b] Y. Chen, and F. Naghdy, “Skill Acquisition in Transfer of Manipulation Skills from Human to Machine through a Haptic Virtual Environment,” Proceedings of IEEE International Conference on Industrial Technology, Productivity Reincarnation through Robotics & Automation , 11－14, pp. 337－342, Bangkok, Thailand, Dec. 2002.

[Christopher 1997] G. A. Christopher, W. M. Andrew, and S. Stefan, Locally Weighted Learning, Artificial Intelligence Review, vol. 11, no. 1－5, pp. 11－73, 1997.

[Data (online)] Data Ming Tools See5 and C5.0, URL: http://www.rulequest.com/see5-info.html.

[Derby 1982] S. J. Derby, “General robot arm simulation program (GRASP); Parts 1 and 2,” Proceedings of ASME Conference on Computer Engineering, pp. 139－154, San Diego, 1982.

[Dixon 2003] W. E. Dixon, “Teach by Zooming: a Camera Independent Alternative to Teach by Showing Visual Servo Control,” Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, no. 27－31, pp. 749－754, Oct. 2003.

[Dong 2003] S. Dong, F. Naghdy, and Y. Chen “Six d.o.f Haptic Rendered Simulation of the Peg-in-Hole Assembly,” Proceedings of International Conference Manufacturing Excellence, Melbourne Australia, Oct. 2003.

[Dufay 1984] B. Dufay and J. C. Latombe, “An Approach to Automatic Robot Programming Based on Inductive Learning,”

International Journal of Robotics Research, vol. 3, no. 4, pp. 3－20, 1984.

[Durlach 1995] N. I. Durlach and A. S. Mavor, Eds, National Research Council, Virtual Reality: Scientific and Technological Challenges, National Academy Press, Washington, D.C., pp. 188－204, 306－317, 1995.

[Ehrenmann 2002] M. Ehrenmann, R. Zollner, O. Rogalla, and R. Dillmann, “Programming Service Tasks in Household Environments by Human Demonstration,” Proceedings of 11th IEEE International Workshop on Robot and Human Interactive Communication, pp. 460－467, Sep. 25－27 2002.

[Esmaili 1995] N. Esmaili, C. Sammut, and G. M. Shirazi, “Behavioral Cloning in Control of a Dynamic System,” Proceedings of IEEE International Conference on Systems, Man and Cybernetics Intelligent Systems for the 21st Century, Vancouver, Canada, pp. 2904－2909, Oct. 22－25, 1995.

[ESmith 1966] E. J. Smith, “The Classification of Education Objectives: Psychomotor Domain”, University of Illinois Research Project No. OE 5, pp. 85－104, 1966.

[Frigui 1997] H. Frigui and R. Krishnapuram, “Clustering by Competitive Agglomeration,” Pattern Recognition, vol. 30, no. 7, pp. 1223－1232, 1997.

[Friedrich 1998] H. Friedrich, J. Holle, and R. Dillmann, “Interactive Generation of Flexible Robot Programs,” Proceedings of IEEE International Conference on Robotics and Automation, Vol 1, pp. 538－543, May 16－20, 1998.

[Fogel 1994] D. B. Fogel, “An introduction to simulated evolutionary optimisation,” IEEE Transactions on Neural Networks, vol. 5, no. 1, pp. 3－14, Jan. 1994.

[Gath 1989] I. Gath and A. Geva, “Unsupervised Optimal Fuzzy Clustering,” IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI), vol. 11, pp. 773－781, 1989.

[Gecko (online)] Gecko Drive G201 Step Motor, URL: http://www.geckodrive.com/product.cfm?pid=9.

[Grudic 1996] G. Z. Grudic and P. D. Lawrence, “Human-to-robot Skill Transfer Using the SPORE Approximation,” Proceedings of IEEE International Conference on Robotics and Automation, pp. 2962－2967, Minneapolis, Minnesota, April, 1996.

[Grunwald 2001] G. Grunwald, G. Schreiber, A. Albu-Schaffer, and G. Hirzinger, “Touch: The Direct Type of Human Interaction with a Redundant Service Robot,” Proceedings of 10th IEEE

International Workshop on Robot and Human Interactive Communication, pp. 347－352, Sep. 18－21, 2001.

[Handleman 1996] D. A. Handleman, and S. H. Lane, “Human-to-machine Skill Transfer through Cooperative Learning,” Intelligent Control Systems, Theory and Applications, M.M. Gupta and N.K. Sinha, Eds, pp. 187－205, 1996.

[Harrow 1972] A. J. Harrow, “A Taxonomy of the Psychomotor Domain,” David McKay Company, p. 16, Inc, New York, 1972.

[Hass 1991] N. Hass, “Learning by Ostentation for Robotics Assembly,” Proceedings of SPIE conference, 1991.

[Holding 1989] D. H. Holding, Human Skills, 2nd ed, Wiley, Chichester, 1989.

[Hoppner 1999] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler, Fuzzy Cluster Analysis, Chichester, J. Wiley & Sons, 1999.

[Iba 2005] S. Iba, C. J. J. Paredis, and P. K. Khosla, “Interactive Multimodal Robot Programming,” The International Journal of Robotics Research, vol. 24, no. 1, pp. 83－104, 2005.

[Ikeuchi 1991] K. Ikeuchi and T. Suehiro, “Towards an Assembly Plan from Observation,” Tech. Rep. CMU-CS-91-167, School of Computer Science, Carnegie Mellon Univ, Pittsburgh, PA, 1991.

[Intelitek (online)] Intelitek, URL: http://www.intelitek.com/products/menu.asp?cid=1&pid=5.

[Isaac 2003] A. Isaac and C. Sammut, “Goal-directed Learning to Fly,” Proceedings of the 20th International Conference on Machine Learning (ICML－2003), Washington DC, 2003.

[Jayaram 1997] S. Jayaram, H. Connacher, and K. Lyons, “Virtual Assembly using Virtual Reality Techniques,” Computer-Aided Design, vol. 29, no. 8, Aug., 1997.

[Jeannerod 1988] M. Jeannerod, “The Neural and Behavioural Organization of Goal-Directed Movements,” Clarendon, Oxford University Press, UK, 1988.

[Kahl 2002] B. Kahl and D. Henrich, “Virtual Robot Programming for Deformable Linear Objects: System Concept and Prototype Implementation,” Proceedings of 12th International Symposium on Measurement and Control in Robotics (ISMCR02), Bourges France, Jun. 20－21, 2002.

[Kaiser 1996] M. Kaiser and R. Dillmann, “Building Elementary Robot Skills from Human Demonstration,” Proceeding of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2700－2705, 1996.

[Kerr 1994] R. M. Kerr and D. Kibira, “Intelligent Reactive Scheduling by Human Learning and Machine Induction,” Proceedings of

IFAC Conference on Intelligent Manufacturing Systems, Vienna, Austria, 1994.

[KSmith 1962] K. U. Smith and W. H. Smith, “Perception and Motor,” W. B. Saunders, 1962.

[Kuniyoshi 1994] Y. Kuniyoshi, and M. Inbana, “Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance,” IEEE Transactions on Robotics and Automation, vol. 10, no. 6, Dec. 1994.

[Laschi 2002] C. Laschi, M. Gonzalo-Tasis, J. F. Codes, and P. Dario, “Recognizing Hand Posture by Vision: Applications in Humanoid Personal Robotics,” Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, no. 11－15, pp. 1439－1444, May 2002.

[Lauria 2002] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, “Mobile Robot Programming using Natural Language,” Journal of Robotics and Autonomous Systems, vol. 38, no. 3－4, pp. 171－181, March 2002.

[Light-Weight (online)] Light-Weight Robots Project. URL: http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-417/557_read-987/.

[Lord 1986] Lord Corporation. “Installation and Operation Manual for F/T Series Force Torque Sensing System,” Lord Corporation, Industrial Automation Division, 407 Gregson Drive, Cary, N.C. 27511, 1986.

[Mark 1996] W. R. Mark, S. C. Randolph, M. Finch, J. M. Van Verth, and R.M. Taylor III, “Adding Force Feedback to Graphics Systems: Issues and Solutions,” Proceedings of ACM SIGGRAPH 96, pp. 447－452, New Orleans, Aug. 1996.

[Martens 2001] C. Martens, N. Ruchel, O. Lang, O. Ivlev, and A. Graeser, “A FRIEND for Assisting Handicapped People,” issue of IEEE Robotics and Automation Magazine, Mar. 2001.

[Mathews 1989] R. C. Mathews, R. R. Buss, W. B. Stanely, F. Blanchard-Fields, J. R. Cho, and B. Druhan, “Role of Implicit and Explicit Processes in Learning from Examples: A Synergistic Effect,” Journal of Experimental Psychology: Learning, Memory and Cognition, vol. 15, no. 6, pp. 1083－1100 1989.

[Matsubara 1985] H. Matsubara, A. Okano, and H. Inoue, “Design and Implementation of a Task Level Robot Language,” Journal of Robotics Society, Japan, vol. 3, no. 3, 1985.

[Matsumoto 2003] A. Matsumoto, G. Yoshita, and I. Kihana, “Teaching by Showing Few Images for the Navigation of Mobile Robots,”

Proceedings of the IEEE International Symposium on Assembly and Task Planning, pp. 270－275, Jul. 10－11, 2003.

[McGuire 2002] P. McGuire, J. Fritsch, J. J. Steil, F. Rothling, G. A. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter, “Multi-modal Human-machine Communication for Instructing Robot Grasping Tasks,” Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System, vol. 2, pp. 1082－1088, Sep. 30 － Oct. 5, 2002.

[McNeely 1999] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, “Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling,” Proceedings of ACM SIGGRAPH 99, pp. 401－408, Los Angeles, California, Aug. 1999.

[Michie 1993] D. Michie, “Knowledge, Learning and Machine Intelligence,” Intelligent Systems, L. S. Sterling, Ed, New York: Plenum, 1993.

[Montgomery 1998] J. F. Montgomery and G. A. Bekey, “Learning Helicopter Control through ‘Teaching by Showing’,” Proceedings of the 37th IEEE Conference on Decision and Control, vol. 4, pp. 3647－3652, Dec. 16－18 1998.

[Muggleton 1992] S. Muggleton, Inductive Logic Programming－Logic Based Approach to ML, ser. 38, London, U.K.: Academic, 1992.

[Myers 2001] D. R. Myers, M. J. Pritchard, and M. D. J. Brown, “Automated Programming of an Industrial Robot through Teach by showing,” Proceedings of IEEE International Conference on Robotics and Automation (ICRA’ 2001), vol. 4, pp. 4078－4083, 2001.

[Nakamura 1996] A. Nakamura, T. Oqasawara, T. Suehiro and H. Tsukune, “Skill-based back-projection for fine motion planning”, Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems ’96, IROS 96, vol. 2, pp. 526－553, 1996.

[Naylor 1987] A. Naylor, I. Shao, R. Volz, R. Jungelas, P. Bixel, and K. Lloyd, “PROGRESS－a Graphical Robot Programming System,” Proceedings of IEEE International Conference on Robotics Automat, pp. 1282－1291, 1987.

[Nicolescu 2001] M. Nicolescu and M. J. Mataric, “Learning and Interacting in Human－Robot Domains,” Special Issue of IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans , vol. 31, no. 5, pp. 419－430, Chelsea C. White and Kerstin Dautenhahn Eds, Sep. 2001.

[Nicolescu 2002] M. Nicolescu and M. J. Mataric, “A Hierarchical Architecture for Behavior-based Robots,” Proceedings of First International

Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 227－233, Bologna, ITALY, Jul. 15－19, 2002.

[Nicolescu 2003] M. Nicolescu and M. J. Mataric, “Natural Methods for Learning and Generalization in Human－Robot Domains,” Proceeding of Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, AUSTRALIA, Jul. 14－18, 2003.

[Ogasawara 1998] T. Ogasawara, H. Hirukawa, K. Kitagaki, H. Onda, A. Nakamura, and H. Tsukune, “A Telerobotics System for Maintenance Tasks Integrating Planning Functions Based on Manipulation Skills,” Proceedings of 1998 IEEE International Conference on Robotics and Automation, vol. 4, pp. 2870－2876, May 16－20, 1998.

[Ogawara 2002] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, “Generation of a Task Model by Integrating Multiple Observations of Human Demonstrations,” Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, pp. 1545–1550, May 11－15, 2002.

[Onda 1995] H. Onda, H. Hirukawa, and K. Takase, “Assembly Motion Teaching System Using Position/Force Simulator－Extracting a Sequence of Contact State Transition”, Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction and Cooperative Robots, vol. 1, pp 9－16, 1995.

[Onda 2001] H. Onda, “Teaching of Assembly Motion by Demonstration: Artificial Constrained Motion Primitives and Its Implementation using Virtual Small Faces,” Proceedings of the IEEE International Symposium on Assembly and Task Planning, pp. 142－147, May 28－29, 2001.

[Onda 2002a] H. Onda, T. Suehiro, and K. Kitagaki, “Teaching by Demonstration of Assembly Motion in VR－detection of Nondeterministic Search-type Motion and Developing of Its Skillful Motion Primitive,” Proceedings of IEEE 28th Annual conference of the Industrial Electronics Society, vol. 4, Nov. 5－8 pp. 2640－2645, 2002.

[Onda 2002b] H. Onda, T. Suehiro, and K. Kitagaki, “Teaching by Demonstration of Assembly Motion in VR － non-deterministic Search-type Motion in the Teaching Stage,” Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System, vol. 3, pp. 3066－3072, Sep. 30 － Oct. 5, 2002.

[OpenGL (online)] OpenGL, URL: http://www.opengl.org/.

[Oliver 2004] N. Oliver, A. Garg and E. Horvitz, “Layered Representations for Learning and Inferring Office Activity from Multiple Sensory Channels,” Computer Vision and Image Understanding, vol. 96, pp. 163-180, 2004.

[Pearce 1999] A. Pearce, C. Sammut, and S. Goss, “Simulation as an Environment for the Knowledge Acquisition of Procedural Expertise,” Proceedings of the 4th International SimTecT Conference, Eds, Grant Tudor, Melbourne Australia, SimTecT99 Organising and Technical Committee, Melbourne, pp. 255－260. Feb 29 – Mar. 1, 1999.

[Perez 1977] T. Lozano Perez and P. H. Winston, “LAMA: A Language for Automatic Mechanical Assembly,” Proceedings of International Joint Conference on Artificial Intelligent, pp. 321－333, 1977.

[Petersik 2002] A. Petersik, B. Pflesser, U. Tiede, K. H. Hohne, and R. Leuwer, “Haptic Volume Interaction with Anatomic Models at Sub-Voxel Resolution”, Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environments & Teleoperator Systems, pp. 66－72, Orlando, USA, Mar. 2002.

[Pomerleau 1993] D. A. Pomerleau, Neural Network Perception for Mobile Robot Guidance, Kluwer Academic Publishers, 1993.

[Potts 2000] P. Aimee, “Phantom-Based Haptic Interaction,” Proceedings of the Computer Science Discipline Seminar Conference (CSCI 3901), URL: http://mrs.umn.edu/~lopezdr/seminar/spring2000/potts.pdf.

[Python (online)] Python, URL: http://www.python.org.

[Rabiner 1989] L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” Proceedings of the IEEE, vol, 77, no. 2, pp. 257－286, 1989.

[Ránky 1985] P. G. Ránky, and C. Y. Ho, Robot Modelling: Control and Applications with Software. IFS Ltd. Springer, 1985.

[Reachin (online)] Reachin Technologies, URL: http://www.reachin.se.

[Renz 2001] M. Renz, C. Preusche, M. Pötke, H. P. Kriegel, and G. Hirzinger, “Stable Haptic Interaction with Virtual Environments Using an Adapted Voxmap－PointShell Algorithm,” Proceedings of the Eurohaptics Conference, Birmingham, UK, 2001.

[Rosenbaum 1991] D. A. Rosenbaum, Human Motor Control, pp. 16, New York, Academic, 1991.

[Ruchel 1999] N. Ruchel, O. Lang, and A. Gräser, “Service Robot Programming by Demonstration with Integration of Sensor Information,” Proceedings of the 1999 IEEE

Hongkong　Symposium on Robotics and Control. vol. 1, pp. 226－231, Hongkong, Jul. 2－3, 1999.

[Ruspini 1997] Ruspini, Diego, K. Kolarov, and O. Khatib, “The Haptic Display of Complex Graphical Environments,” Proceedings of SIGGRAPH 97, pp. 345－352, Los Angeles, CA, Aug. 3－8, 1997.

[Salisbury 1995] J. K. Salisbury, D. Brocki, T. Massiet, N. Swarupf, and C. Zillest, “Haptic Rendering: Programming Touch with Virtual Objects,” Proceedings of the 1995 Symposium on Interactive 3D Graphics, pp. 123－130, Monterey, CA, 1995.

[Sammut 1992] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, “Learning to Fly,” Proceedings of the 9th International Workshop Machine on Learning. pp. 385－393, San Mateo, CA: Morgan Kaufmann, 1992.

[Sanyo (online)] Sanyo Denki 2 Phase Stepping System, URL: http://www.sanyo-denki.com/Products/Stepping%20Motors/menu1.htm#sups.

[Sato 1997] D. Sato, S. Yamada, and M. Uchiyama, “Human Skill Analysis Based on Multi-sensory Data,” Proceedings of IEEE International workshop on Robot and Human Communication, pp. 278－283, 1997.

[Schoppers 1987] M. Schoppers, “Universal plans for reactive robots in unpredictable environments,” Proceedings of the 10th International Joint Conference on Artificial Intelligence, vol. 11, pp. 1039－1046, 1987.

[SensAble (online)] SensAble Technologies, URL: http://www.sensable.com.

[She 2003] H. She, C. Martens, and A. Graeser “Application of Programming by Demonstration in the Rehabilitation Robotic System FRIEND,” The 8th International Conference on Rehabilitation Robotics (ICORR 2003), Daejeon, South Korea, Apr. 23－25, 2003.

[Simpson 1966] J. S. Simpson, “The Classification of Educational Objectives: Psychomotor Domain,” University of Illinois Research Project No. OE 5, pp. 85－104, 1966.

[Skoglund 2005] A. Skoglund, R. Palm and T. Duckett, “Towards a Supervised Dyna-Q Application on a Robotic Manipulator,” Proceedings of SAIS-SSLS 2005, 3rd Joint Workshop of the Swedish AI and Learning Systems Societies Mälardalen, Sweden, Apr. 12－14, 2005.

[Smart 2002] W. D. Smart and L. P. Kaelbling, “Effective Reinforcement Learning for Mobile Robots,” Proceedings of IEEE

International Conference on Robotics and Automation, vol. 3, pp. 3404－3410, May 11－15, 2002.

[Smith 2002] L.I. Smith, “A tutorial on Principle Component Analysis,” Technical Report, University of Otago, New Zealand, 2002.

[Steil 2001] J. J. Steil, G. Heidemann, J. Jockusch, R. Rae, N. Jungclaus, and H. Ritter, “Guiding Attention for Grasping Tasks by Gestural Instruction: the GRAVIS-robot Architecture,” Proceedings of 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1570－1577, Oct. 29 － Nov. 3, 2001.

[Stereo (online)] Stereo Graphics Corporation. URL: http://www.stereographics.com.

[Strobel 2002] M. Strobel, J. Illmann, B. Kluge, and F. Marrone, “Using Spatial Context Knowledge in Gesture Recognition for Commanding a Domestic Service Robot,” Proceedings of 11th IEEE International Workshop on Robot and Human Interactive Communication, pp. 468－473, Sep. 25－27, 2002.

[Suc 1997] D. Suc and I. Bratko, “Skill Reconstruction as Induction of LQ Controllers with Subgoals,” Proceedings of 15th International Joint Conference on Artificial Intelligent, vol. 2, pp. 914－920, C. S. Mellish, Ed, Japan, Aug. 1997.

[Suc 1998] D. Suc and I. Bratko, “Skill Modeling through Symbolic and Qualitative Reconstruction of Operator’s Trajectories,” Technique Report, Artificial Intelligent Lab, Faculty of Computer Information Science, University of Ljubljana, Slovenia, 1998.

[Suc 2000] D. Suc and I, Bratko, “Problem Decomposition for Behavioral Cloning,” Proceedings of the European Conference on Machine Learning, pp. 382－391, New York: Springer-Verlag, 2000.

[Summers 1984] P. D. Summers and D. D. Grossman, “XPROBE: An Experimental System for Programming Robots by Example,” International Journal of Robotics Research, vol. 3, no. 1, pp. 25－39, 1984.

[Takamatsu 1999] J. Takamatsu, H. Kirnura, and K. Ikeuchi, “Classifying Contact States for Recognizing Human Assembly Task,” Proceedings of 1999 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFT ’99, pp. 177－182, 1999.

[Takamatsu 2000] J. Takamatsu, H. Tominaga, K. Ogawara, H. Kimura, and K. Ikeuchi, “Extracting Manipulation Skills from Observation,” Proceedings of 2000 IEEE/RSJ International Conference on

Intelligent Robots and Systems (IROS 2000), vol. 1, pp. 584－589, Oct. 31 － Nov. 5, 2000.

[Takamatsu 2002] J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, “Correcting Observation Errors for Assembly Task Recognition,” Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System, vol. 1, pp. 232－237, Sep. 30 － Oct. 5, 2002.

[Ude 1994] A. Ude and R. Dillmann, “Vision-Based Robot Path Planning,” Journal of Advances in Robot Kinematics and Computational Geometry, J. Lenarcic and B. Ravani, Eds, pp. 505－512, Kluwer, Dordrecht, 1994.

[Urbancic 1994] T. Urbancic and I. Bratko, “Reconstructing Human Skill with Machine Learning,” Proceedings of the 11th European Conference on Artificial Intelligence, A. Cohn, Ed, New York: Wiley, pp. 498－502, 1994.

[Voyles 1997] R. M. Voyles, “Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation,” doctoral dissertation, tech. report CMU-RI-TR-97-36, Robotics Institute, Carnegie Mellon University, Aug. 1997.

[Voyles 1999a] R. M. Voyles, J. D. Morrow, and P. K. Khosla, “Gesture-based Programming for Robotics: Human-augmented Software Adaptation,” Journal of IEEE Intelligent Systems and Their Applications, vol. 14, Issue 6, pp. 22－29, Nov.－Dec. 1999.

[Voyles 1999b] R. M. Voyles, and P.K. Khosla, “Gesture-based Programming: a Preliminary Demonstration,” Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 708－713, May 10－15, 1999.

[VRML (online)] VRML. URL: http://www.web3d.org/vrml/vrml.htm.

[Weng 2000] J. Weng, W. Hwang, Y. Zhang, C. Yang, and R. Smith, “Developmental Humanoids: Humanoids that Develop Skills Automatically,” Proceedings of The 1st IEEE－RAS International Conference on Humanoid Robots, Boston, MA, Sep. 7－8, 2000.

[Yano 2003] Y. Yano, “An Intuitive Teaching Method,” http://www-cv.mech.eng.osaka-u.ac.jp/~yano/research/study1-e/study1.html, 2003.

[Yokokohji 2005] Y. Yokokohji, Y. Kitaoka, and T. Yoshikawa, “Motion Capture from Demonstrator's Viewpoint and its Application to Robot Teaching,” Journal of Robotic Systems, vol. 22, Issue 2, pp. 87－97, Jan. 10, 2005.

[Yuan 2000] X. Yuan, “Interactive Assembly Planning in Virtual Environments,” Proceedings of 2000 IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS 2000), vol. 2, pp. 1462－1467, Oct. 31 － Nov. 5, 2000.

[Yuan 2004] X. Yuan and S. X. Yang, “Interactive Assembly Planning with Automatic Path Generation,” Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), vol. 4, pp. 3965－3970, Sep. 28 － Oct. 2, 2004.

[Zhang 2001] Y. Zhang and J. Weng, “Grounded auditory development of a developmental robot,” Proceedings of INNS－IEEE International Joint Conference on Neural Networks, pp. 1059－1064, Washington, DC, Jul. 14－19, 2001.

[Zhang 2002] Y. Zhang and J. Weng; “Chained Action Learning through Real-time Interactions,” Proceedings of the 2002 International Joint Conference on Neural Networks, vol. 3, pp. 2012－2017, May 12－17, 2002.

[Zhang 2005] Y. Zhang; J. Weng, and W. Hwang, “Auditory Learning: A Developmental Method,” IEEE Transactions on Neural Networks, vol. 16, Issue 3, pp. 601－616, May 2005.

[Zollner 2002] R. Zollner, O. Rogalla, R. Dillmann, and M. Zollner, “Understanding Users Intention: Programming Fine Manipulation Tasks by Demonstration,” Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System, vol. 2, pp. 1114－1119, Sep. 30 － Oct. 5, 2002.

# APPENDIX A Experimental Rig Hardware Introduction

## *A.1 Introduction*

The hardware employed in this project, including the stepping motors, micro-step drives, DC servo motor, DC servo motor encoder, DC servo controller, DAQ-802 data acquisition board, HCTL-2016 decoder, and Lord force/torque sensor, are introduced as follow.

## *A.2 Sanyo Denki 2 Phase Stepping Motor*

Two Sanyo Denki 2 phase 60mm square stepping motors, model no. 103H7822-0440, were employed in the physical experimental rig to control the hole and provide two degrees of freedom (2 DOFs) (Figure A.1) [Sanyo (online)].

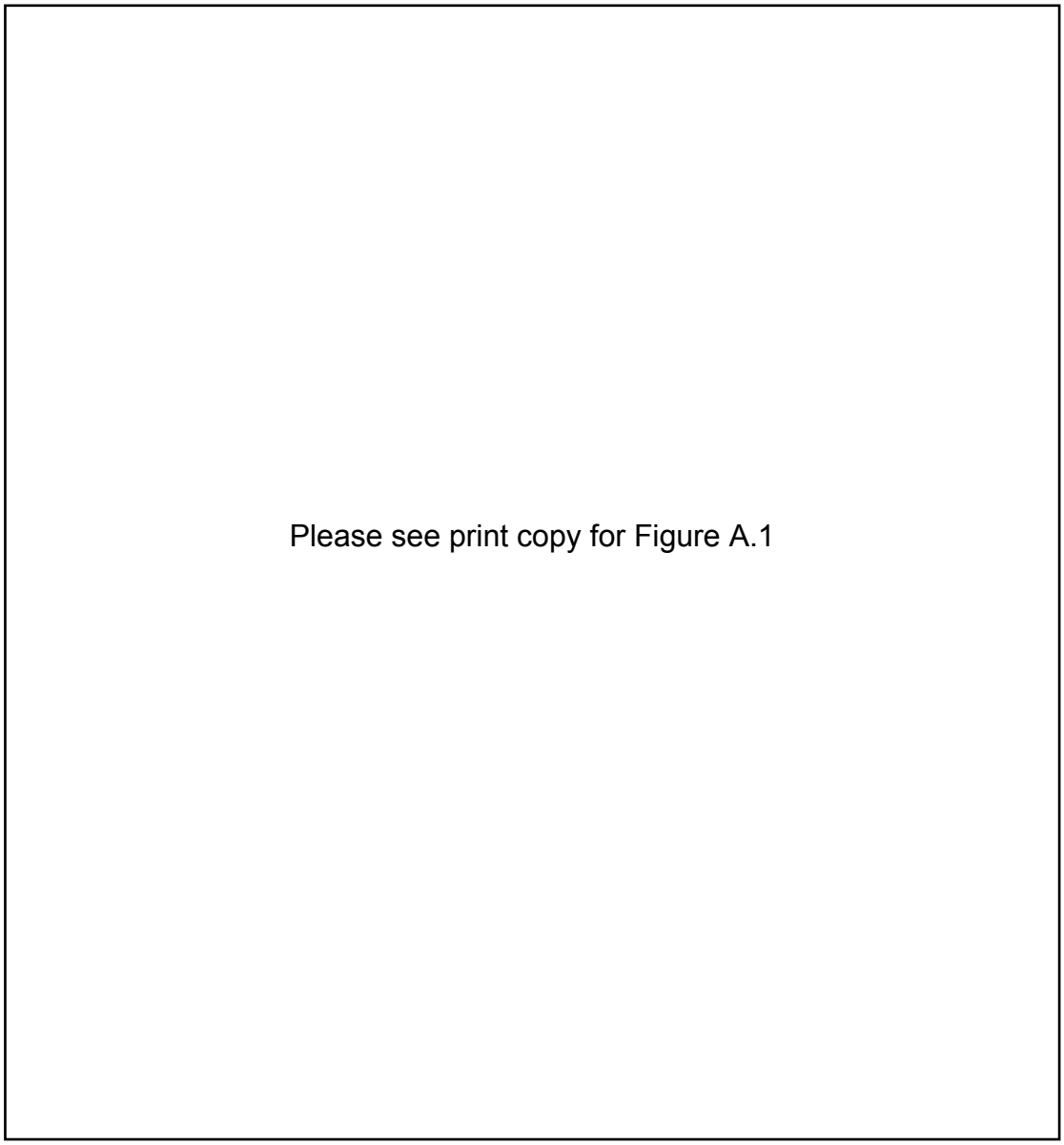



Figure A.1 Sanyo Denki 2 Phase Stepping Motor [Sanyo (online)]

The stepping motor has the flowing major features that suitable for the project [Sanyo (online)]:

1. Compact size: H=82mm, W=60mm, D=53.88mm.
2. High torque: 1170mNm. 2 phase hybrid rare earth magnet technology offering 15% to 20% more torque than standard hybrid types.

3. Low noise: lower noise has been realized by an optimum structure design employed for the motors.
4. Simple wiring: the motor equipped with a connector promises easier system design (Figure A.2).

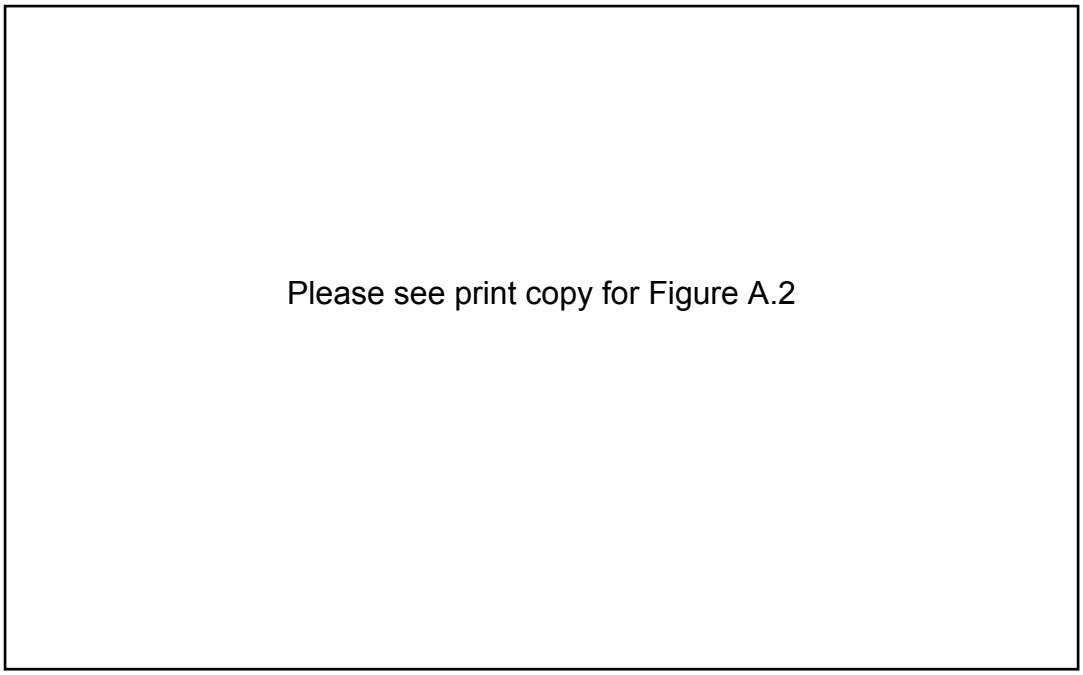



Figure A.2 Sanyo Denki Motors wiring connections [Sanyo (online)]

Other features include [Sanyo (online)]:

1. Very high positional accuracy, designed for micro-steeping.
2. The motor can be unipolar or bipolar driven.
3. The motor can be available with integrated connector for ease of assembly, crimps and matching socket supplied.
4. Supply Voltage: 4V

   Current: 2A/phase

Resistance: 2 Ω /phase

Inductance: 3.6mH/phase

Weight: 770g

## *A.3 Gecko Micro-step Drive*

The Gecko micro-step drives used in the project can provide a minimum of 0.09 degree resolution for turning the hole and enhance the holding torque through a special circuit. It doesn't need external circuits to generate and enlarge the clock and direction signals. The Gecko micro-step drive is illustrated in Figure A.3 [Gecko (online)].

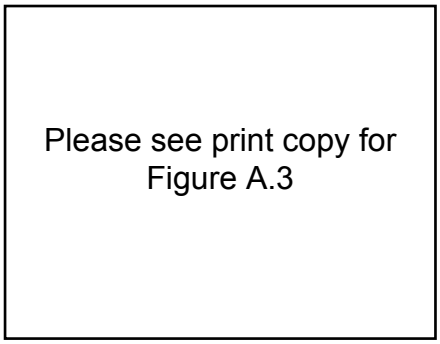



Figure A.3 Gecko micro-step drive [Gecko (online)]

The Gecko micro-step drive has the following specifications [Gecko (online)]:

1. Supply Voltage: 24 to 80 VDC
2. Phase Current: 1 to 7 Amps and 0.3 to 2 Amps (2 ranges)
3. Auto Current Reduction: 33% of set current, 1 second after last Step Pulse
4. Size: 2.5"W, 2.5"D, .85"H (63.5mm, 63.5mm, 21.5mm)

5. Mounting Pattern: 4 6-32 screws, 1.75” by 2.375” (44.5 mm, 60 mm)
6. Quiescent Current: 15 Ma or less
7. Weight: 3.6 oz. (100 gm)
8. Step Frequency: 0 to 200 kHz
9. Step Pulse “0” Time: 0.5 uS min (Step on falling edge)
10. Temp: 0 to 70 C
11. Step Pulse “1” Time: 4 uS min
12. Humidity: 0 to 95 % (non-condensing)
13. Direction Setup: 1 uS min (20 uS min hold time after Step edge)
14. Power Dissipation: 1 to 18 W (1 to 7 Amps)

The connection diagram of the Gecko micro-step drive is illustrated in Figure A.4

[Gecko (online)]

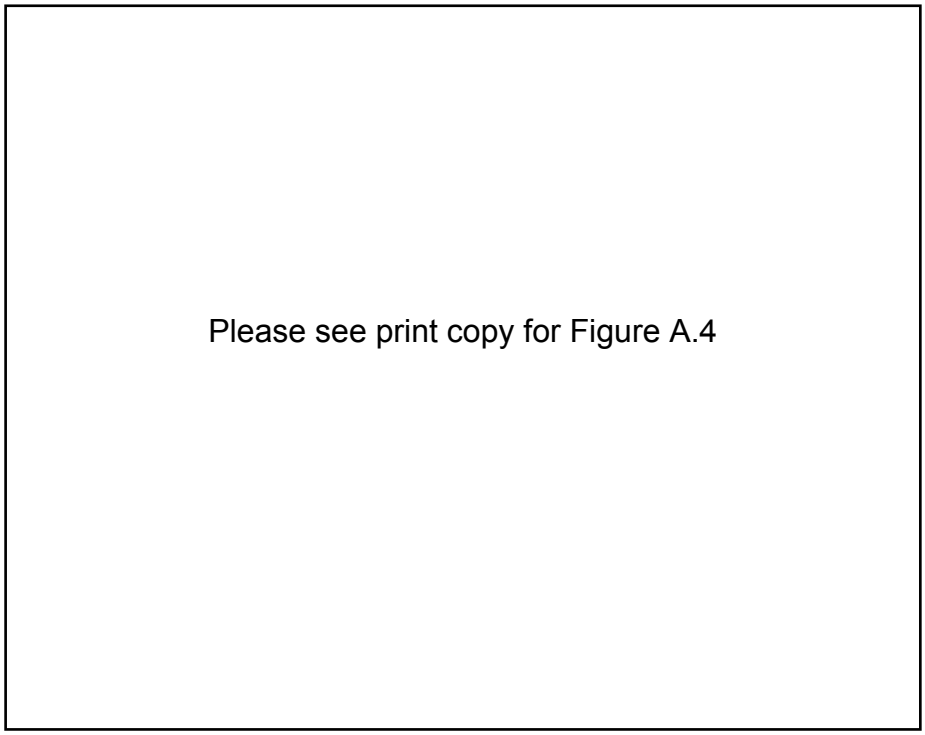



Figure A.4 Gecko micro-step drive connection diagram [Gecko (online)]

## *A.4 Baldor DC Servo Motor*

Baldor MTE Series DC servo motor, model no. MTE-2250-AMACN, is employed in this project, which is used to control the peg and provide one degree of freedom (1 DOF). It is environmentally rugged that can provide reliability and long life in industrial applications. The Baldor's DC Servo Motors are shown in Figure A.5 [Baldor (online) a]

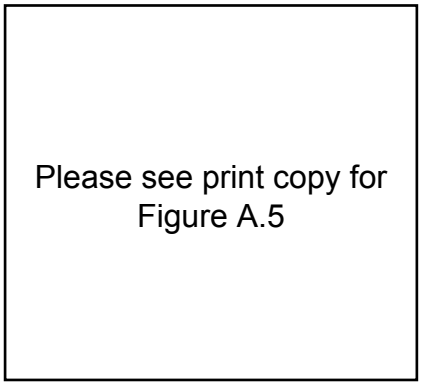



Figure A.5 The Baldor's DC Servo Motors [Baldor (online) a]

The features of the Baldor's DC Servo Motors include [Baldor (online) a]:

1. Provide continuous torques from 1.9 lb.-in to 58 lb.-in.
2. high continuous duty 155° C rotor temperature and premium moisture resistant.
3. A rugged industrial housing and can be supplied with many electrical and mechanical options.
4. Stock and custom design available.
5. Superior performance down to zero speed.
6. Ideal for applications such as X-Y tables, coil winders, machine tool, robotics, factory automation, labeling equipment, assembly equipment, textile, packaging, converting equipment, and laboratory equipment.

## *A.5 Baldor DC Servo Motor Encoder*

14-pin Baldor DC Servo Motor Encoder, model no. MS3102E-20-27P is used in the project that provides a physical link between DC motor and DC motor controller.

The connection diagram is shown in Figure A.6 [Baldor (online) a].

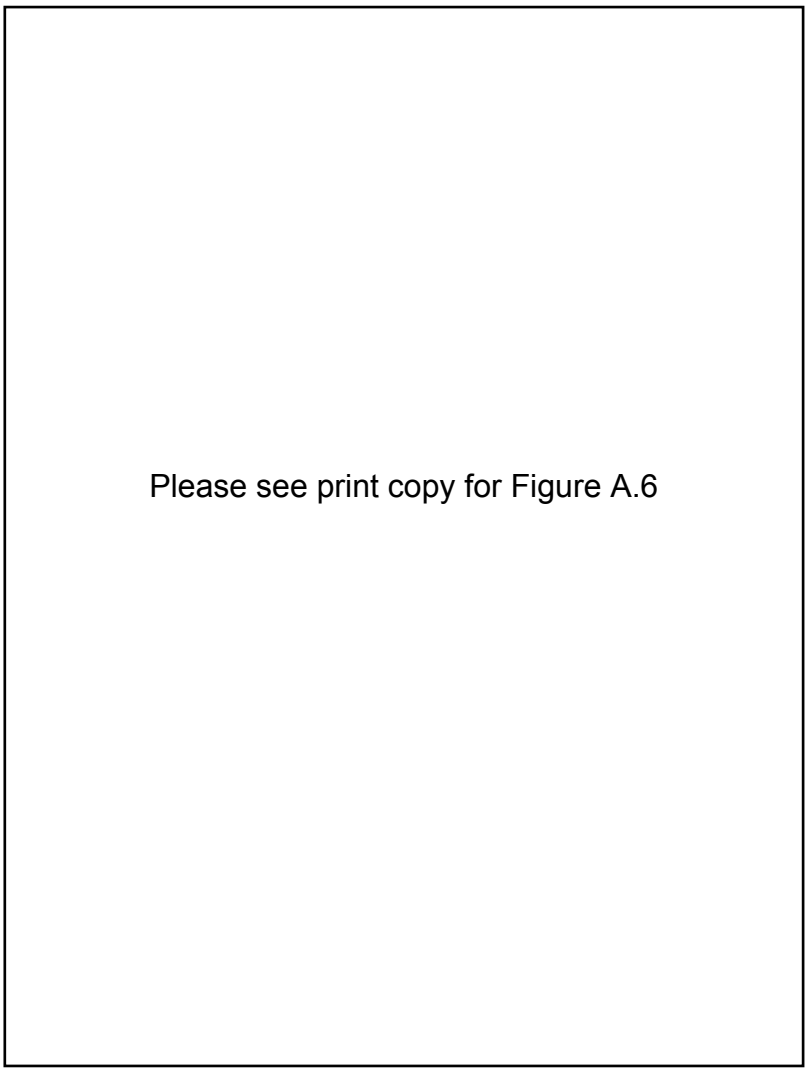



Figure A.6 14-pin Baldor DC Servo Motor Encoder connection diagram [Baldor (online) a]

## *A.6 Baldor DC Servo Motor Controller*

The Baldor TSD (Twin Servo Drive) series motor controller, model no. TSD-050-05-1-I (Figure A.7) is employed in the project to control the one degree

freedom peg. The TSD series motor controller is totally enclosed, stand-alone one or two axes brush-type PWM servo control, utilizing the latest in FET/IGBT transistors for efficiency and reliability. This DC Servo Control is fully protected. It contains a front panel on/off switch and operates directly from 115 VAC. The TSD will power DC servo motors providing up to 5 amps continuous, 10 amps peak. [Baldor (online) b].

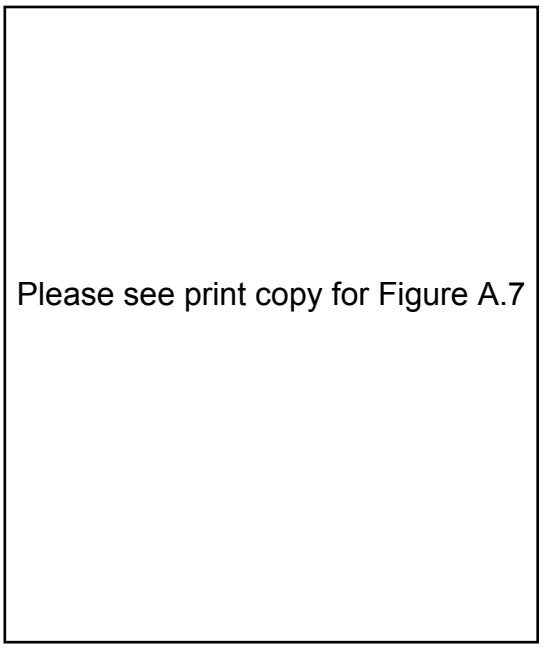



Figure A.7 The Baldor TSD motor controller [Baldor (online) b]

The Baldor TSD series motor controller has the following features [Baldor (online) b]:

1. Easily set up for velocity or torque (current) control applications.
2. Form factor 1.01 or better.
3. Zero deadband performance
4. Adjustable current limits: Peaks and Continuous.
5. Detachable screw terminal inputs (no special tools).

6. Panel mount enclosure ensures there are no exposed electronics.
7. Simplified ‘start up’ as all connections are defined right on the exterior of the enclosure.
8. ON/OFF main toggle switch.
9. No audible noise with 20 kHz switching.
10. No additional inductors required.
11. Protection features, with LED indicators for
    i. Voltage Error
    ii. Surge Current
    iii. Over Temperature

Typical connections of the Baldor TSD series motor controller is shown in Figure A.8

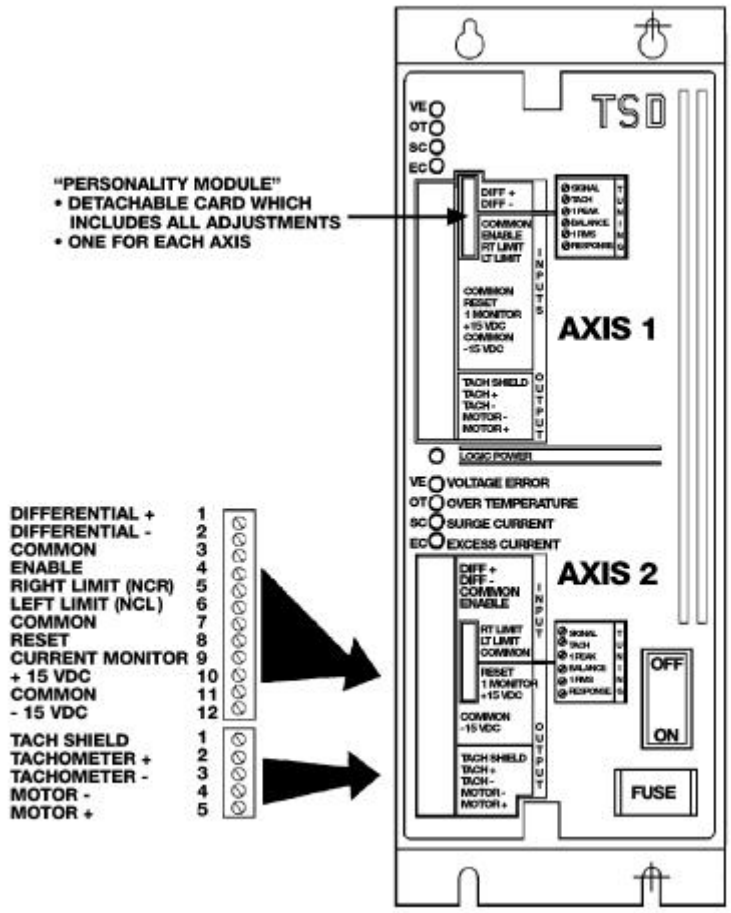



Figure A.8 The Baldor TSD series motor controller typical connections

## *A.7 Quatech DAQ-802 Data Acquisition Board*

The Quatech's DAQ-802 data acquisition board is used to read the decoded signal generated by the decoder HCTL-2016 [Agilent (online)] connected to the encoder of the DC servo motor, send signals to the DC controller controlling the servo motor driving the peg and the stepper motors driving the hole. All of these input and output signals are read into the Host PC by a DAQ-802 I/O interface card through a Main I/O connector and an Auxiliary connector.

Quatech's DAQ-802 is a low-cost data acquisition board with 12-bit resolution,

eight analog input channels, two analog output channels, 32 digital I/O channels, and three 16-bit programmable counter/timers (Figure A.9). The maximum sampling rate is 40 kS/sec for A/D with internal or external triggering [DAQ (online)].

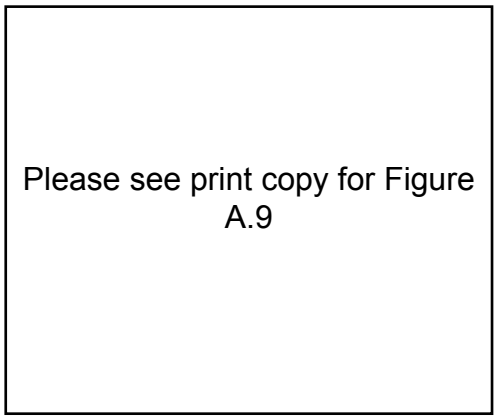



Figure A.9 Quatech DAQ-802 Data Acquisition Board [DAQ (online)]

The DAQ-802 provides eight differential analog inputs. It has a bipolar input range with software programmable gains of 1, 2, 4, and 8 (DAQ-802), and has auto zeroing and a self-calibrating facility for A/D conversion. The main D-37 connector is for analog I/O, control lines, and eight digital I/O channels. A second auxiliary D-37 connector, requiring an additional slot in the PC, is provided for an additional 24 digital I/O lines from an 8255 programmable peripheral interface chip (Figure A.10) [DAQ (online)].

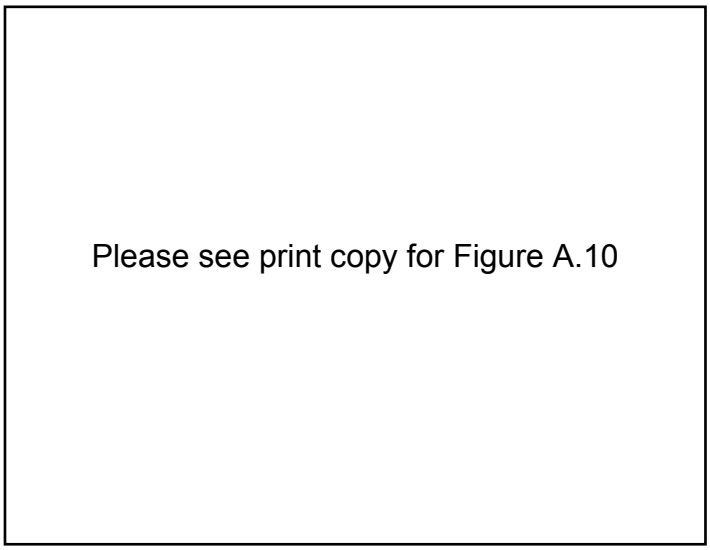



Figure A.10 DAQ-802 mainD-37 connector and auxiliary D-37 connector [DAQ (online)]

A 2K Data FIFO is available that provides a cushion for the data stream coming from the output of the A/D converter. This protects data integrity when using an interrupt routine under Windows or other operating systems. Scan List capability is also provided for scanning the input channels with their corresponding gains. Sequential scanning between any two channels can be programmed. The DAQ-802 can be installed in any available I/O base address location without conflict with presently installed devices. The board can be enabled or disabled through software manipulation. The interrupt levels are register selectable through software from IRQ 2-7, 10-12, 14, and 15 [DAQ (online)].

Features of the DAQ-802 Data Acquisition Board include [DAQ (online)]:

1. 40 kS/sec sampling, 12-bit analog input resolution.
2. Eight differential analog inputs.

3. Two 12-bit D/A channels, three 16-bit timer/ counters and 32 channels digital I/O.
4. Self calibrating and auto zeroing.
5. Channel scan capability with various gain for each channel.
6. 2K Data FIFO buffer.
7. Programmable gains of 1, 2, 4, and 8.
8. Interrupt handling capability.
9. Drivers supplied for third-party software.

## *A.8 HCTL-2016 decoder*

The HCTL-2016 decoder [Agilent (online)] provides a physical link between the Baldor DC Servo Motor Encoder and the DAQ-802 data acquisition board in order to obtain the position information. The pinouts are shown in Figure A.11. CHA and CHB read signals from the DC servo motor through the motor encoder. The 12/1A-bit position latch is read through an 8-bit output port (D0-D7) in 2 sequential bytes. At first, both OE and SEL pins are set low via DAQ-802, and read in high byte of position. Then, SEL pin is set high via DAQ-802 and read in low byte of position. Finally, high and low bytes are combined to produce a 16 bit word.

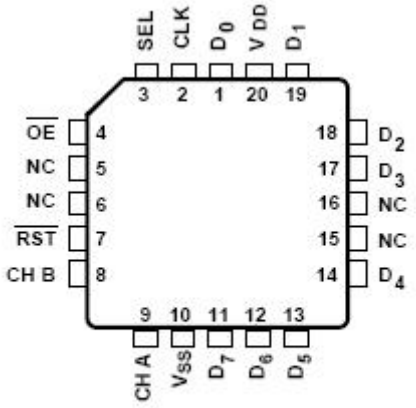



Figure A.11 HCTL-2016 pinouts

## *A.9 Lord force/torque sensor*

The force/torque data is obtained by Lord Force/Torque (F/T) sensing system, which consists of a transducer unit and a system controller unit.

In this project, the F/T system is connected to the computer through RS232 serial port, which is configured at 9600 baud rate and 8N1 (8bits, noparity, 1 stop bit) mode. The commands issued through the serial port are as follow:

- **FT**: Select resolved force/torque data

  In response to this commend, forces and torques applied on the transducer unit are resolved into three Cartesian force components *fx, fy, fz* and three Cartesian torque components *tx, ty, tz*. The force/torque components are represented as a six-element vector, *F=(fx, fy, fz, tx, ty, tz)*.

- **OA**: Output ASCII on the serial port

  In response to this command, the F/T system will output continuous data in

ASCII mode. If FT type is selected, F/T system records are transmitted as strings of 37 characters. The first character is ‘0’ or ‘1’ representing the strain gauge saturation flag. Following are the six Cartesian force/torque components, each is expressed as a decimal number, right-hand justified in a six character field.

- **OR**: Output one data record in ASCII mode

  In response to this command, output record is in the same format as OA command, but only one data record is issued (one force/torque components vector).

- **BS:** Remove bias

  Normally, force/torque data output is biased by gravitational loading due to the weight of the end effector, work piece and any attached cables or hoses. For task where these effects are constant (the orientation of the end effector remains fixed with respect to the gravity vector), BS command can be used to remove this bias. This command establishes the current Transducer Unit load as bias to be subtracted from all subsequent force/torque output.

# APPENDIX B Related Publications

## *Journal Paper*

1. Shen Dong, Fazel Naghdy, “Reproduction of Human Manipulation Skills in a Robot”, Journal of Robotics and Computer Integrated Manufacturing (RCIM), 2005

## *Conference Paper*

2. Shen Dong, Fazel Naghdy, “Application of Competitive Clustering to Acquisition of Human Manipulation Skills”, IEEE International Conference on Fuzzy Systems, 2006 pp. 78-83
3. Shen Dong, Fazel Naghdy, “Manipulation Skills Acquisition and Classification from Haptic Rendered Virtual Environment”, The 2005 World Congress in Applied Computing, June 20-23, 2005, Las Vegas, Nevada, USA. pp. 103-108
4. Shen Dong, Fazel Naghdy, “Manipulation Skills Acquisition through State Classification and Dimension Decrease”, The 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 14th-16th, November, 2005 Hong Kong. pp. 392-396
5. Shen Dong, Fazel Naghdy, “Competitive Clustering Application on Acquisition of Human Manipulation Skills”, in International Conference on Computational Intelligence for Modelling, Control & Automation CIMCA2005, 2005, pp.

1092-1097.

6. Shen Dong, Fazel Naghdy, “Manipulation Skills Acquisition by HMMs from Haptic Rendered Virtual Environment”, World Haptics Conference, March 18-20, 2005, Pisa, Italy.
7. Shen Dong, Fazel Naghdy, “Reproduction of Human Manipulation Skills in a Robot”, International Manufacturing Leaders Forum (IMLF-2005), February 2005, Adelaide, Australia.
8. Shen Dong, Fazel Naghdy, “Haptic Rendered Virtual Modelling of Gear Assembly”, 3rd IFAC Symposium on Mechatronic Systems, September 2004, Sydney, Australia.
9. Shen Dong, Fazel Naghdy, Yuxin Chen, “Six d.o.f Haptic Rendered Simulation of the Peg-in-Hole Assembly”, 9th International Conference on Manufacturing Excellence 2003 (ICME 2003), October 2003, Melbourne, Australia.