INTERACTIVE CONTROL OF ARTICULATED STRUCTURES IN THE VIRTUAL SPACE

By

Kwok Lai Ho Victor

A Dissertation

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Computer Science and Engineering

The Chinese University of Hong Kong

June 1998

摘要

長久以來，機械手／臂模擬系統被局限在二維輸出／入裝置，如鍵盤、二維屏幕等．由於二維視窗環境的限制，機械手的三維模擬操作便依賴於把數個被限的一維或二維操縱空間結合起來。這方法每每都是由用者手動操控，其控制方法非常昂貴和不自然的。本篇論文主要發展一個實時三維互動控制系統，並應用在機械手／機械臂與物件之交介面上。關節結構如手或臂的移動控制將使用動力學和運動學技術，並討論和應用在機械手／臂的互動操控上。我們提議一個使用以上程序的混合方法以控制整體移動和局部微調。同時，我們結合了視覺修改和運動力計算以模擬機械手和物件的互動。我們的實驗顯示了虛議機械手／臂模擬是一個有效和靈活的操控介面。

Abstract

Robot simulation systems have been for a long time limited to 2D input/output devices, such as keyboard and 2D graphics screen. Due to the limitation of 2D windowing environments, the simulation of robot operations relies on the 3D integration of several reduced control spaces, which is usually done manually. The manual integration process by the operator is rather costly and unnatural. This thesis develops a real-time 3D interactive control system of a robot arm and hand manipulating 3D objects. Both kinematics and dynamics techniques for deriving the movement of an articulated structure like an arm or hand are discussed and experimented for interactive manipulations of robots. A hybrid approach using these algorithms for global movement and local refinement is proposed. Also, we combine the visual correction technique and dynamic force calculation to simulate the interaction between objects and hand. Our experiments certainly show the effective and flexible control interface for performing the virtual robotic simulations.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

For many years, robot systems have been studied and applied to many areas. Virtual reality has emerged as one of the hottest topics in computer field. Within the 3D immersive environment, the user can walk or fly through the design template, operate on surrounding objects, view and examine the structure of the models, etc. Although virtual reality appears promising, but the amount of applications are very limited. The main reason of the limitation is the high computation power and special hardwares required to perform the virtual operations, which are very expensive and not easily available. Since the technology of hardware has been improved rapidly in recent years, it is very likely that VR will become the most promising technology for developing the future computer applications.

To provide a highly interactive environment for articulated structure manipulations, we need to deal with representation, coordinate frames and control algorithms. These derive the motion to fulfill a certain goal, such as the end position and orientation of the robot arm/hand while grabbing a mechanical object. Even through motion control techniques have been applied extensively in robotics area for a long time, their use is mainly for off-line autonomous robot systems. The general use of these techniques in a common VR simulation platform is unknown. The first goal of the research is to find an efficient control approach in the virtual robotic manipulations. There are typical control techniques used in robotics,

which are kinematics and dynamics. In general, kinematics uses the geometrical properties and coordinate transformations to derive a motion; dynamics uses physical laws as building blocks to work out the forces and torques by the motion. These two control techniques will be investigated and experimented in performing the real-time robotic interactions.

The second goal of this research is to provide flexible, immersive VR environment as a common platform for robot simulations. For many years, robot simulations are mainly limited to 2D window-layout graphics environment. The motion of the robot control is displayed on 2D devices, like screen or control panel, which contains only visual information shown to guide the interactive control of the motion. The input of the control is performed through either 1D or 2D parameter interface one at a time. Obviously, there are several disadvantages in such control environment:

- The 2D display cannot provide the realistic view observing the robot interactions in the 3D working environment.
- Due to the reduced control space of each input device, extra efforts are required to control the 3D movement in the environment.
- The 2D interaction between the user and the environment cannot completely simulate the control environment of a robot system.
- Extra training is required for the operator to control the system, which will increase the cost of operation.

In contrast to 2D limitations, 3D interactive environment can provide a better environment for interactive robotic control. It allows the user to specify the robotic control directly in the task space. 3D devices can be used to record the required position and orientation of the goal, and be transmitted to the system. The user can also interactively modify the goal according to the result of the simulation. Direct control interface of robot manipulations makes the design process much easier, which reduces the cost and efficiency of control system development.

Special training of the user is not necessary, which make possible of the introduction of robotic control into our daily life. The real-time 3D simulation we proposed in this research will provide common function modules and framework that can be used in various robot applications.

In addition to 3D interactive environment, feedback systems will be used to improve the ease of control. Recently, people start to understand that a good feedback system can improve the efficiency of the job. For instance, the user can feel the force from the controller and knows that the robot he controlling has hit the virtual table when it walks through the room. It reminds the user to move the robot backwards and go around the table. Moreover, feedback systems is necessary in the operations which requires accurate simulation. The user has to receive more information before he can determine the next action he takes. Currently two main kinds of feedback systems are commonly used, which are the force/touch feedback and visual feedback system. In this research, we will discuss and compare the advantage and disadvantage of these two systems. In the following chapters, we will show that enhanced visual feedback is more suitable to be used in our research. A detail analysis will be done on how enhanced visual feedback system can be used to improve our system. An approach called visual correction will be used as the base of interaction between virtual hand and virtual object. Finally, we will combine the motion control techniques with the feedback system in our object manipulation system.

In the following, Chapter 2 outlines the development of robot systems. New approach of robot simulation in virtual reality will also be discussed Chapter 3 presents the objectives of this research. Chapter 4 describes the robotic structures and representation. Chapter 5 presents two virtual manipulator, which are the examples of robotic structures discussed in chapter 4. Chapter 6 discusses the two typical control techniques, kinematics and dynamics, and their use in both forward and inverse modes in robot applications. The use of constraints and optimization in robot simulation is also discussed. Chapter 7 describes some previous works in force/touch feedback systems and discuss the advantage and

disadvantage of them. Chapter 8 presents how visual correction is used in our research. Experiments base on the above discussion will be presented in chapter 9 and chapter 10 concludes the research.

Chapter 2

Background

For decades after the invention of the first robot, robotic systems has been applied to many areas. It replaces human labors in tedious, repetitive task and allow us to focus on design and creative activities. In this section, the development of robotic systems is presented, which can be divided into three major stages: autonomous systems, 2D windowing simulators, and virtual robot simulators.

2.1 History of Robotics

The term **robot** was first introduced into our vocabulary by the Czech playwright Karel Capek in his 1920 play *Rossum's Universal Robots*, the word *robota* being the Czech word for work. Afterwards, the term was applied to describe a great variety of mechanical devices, such as autonomous land rovers, teleoperators, underwater vehicles, etc. Nowadays, every kinds of machines which operates with some degree of autonomy, usually under computer control, can be called a robot.

In our context, the term robot means a computer controlled industrial manipulator. An official definition of such a robot comes form the **Robot Institute of America** (RIA): *A robot is a re-programmable multi-functional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.*

Robotic system has its own attraction to the industrial environment. It is

commonly know that the introduction of robots can decrease the labor costs, increase precision and productivity, increased flexibility compared with specialized machines. Also, working conditions such as dull, repetitive, or hazardous jobs are more suitable to be performed by robots.

The robot, as we have defined it, was born out of the marriage of two earlier technologies: that of **teleoperators** and **numerically controlled milling machines**. Teleoperators were developed during the second world war to handle radioactive materials. Computer numerical control(CNC) was developed because of the high precision required in the machining of certain items, such as components of high performance aircraft. The first robots essentially combined the mechanical linkages of the teleoperator with the autonomy and programmability of CNC machines. The followings are several milestones on the road to present day robot technology:

1947 the development of first servoed electric powered teleoperator

1948 a teleoperator with force feedback is developed

1954 the first programmable robot designed, by George Devol

1961 the first Unimate robot is installed in a Trenton, New Jersey plant of General Motors

1963 the development of first robot vision system

1971 the development of Stanford Arm at Stanford University

1973 the development of first robot programming language (WAVE) at Stanford

1974 the introduction of T^3 robot with computer control by Cincinnati Milacron

1978 the introduction of PUMA robot by Unimation, which based on designs from a General Motors study

1976 the development of Remote Center Compliance (RCC) device for part insertion in assembly at Draper Labs in Boston

1981 the development of first direct-drive robot at Carnegie-Mellon University

It should be pointed out that the important applications of robots are not limited to those jobs which the robot is replacing the human worker. There are also many other applications which the use of humans is impractical or undesirable. Among these are undersea and planetary exploration, satellite retrieval and repair, the defusing of explosive devices, and work in radioactive environments.

2.2 Autonomous Robot Systems

Robotics system has been researched extensively for decades. Various robotic systems have been developed and applied to applications [32], including heavy machines in construction site, space robots for exploring unknown universe, and tiny robots for performing surgery inside human body [29]. Very often, robotic systems are introduced to improve the efficient of the task. For example, if the car parts are assembled by human workers, it is often a slow and tedious task. With the use of robotic system, however, the production cycle can be speed up, and the cost of operation is reduced. It is not a surprise to see more and more applications of Robotic system.

In the earliest stage, the control of robotic system is done by direct manipulation of each joint. User specifies the desire configurations by controlling the rotation or extension of each joint. A typical example of such system is the mechanical excavator in the construction site. The worker controls its motion by switches and levers.

Although autonomous system has been commonly used in many areas, it still has some problems which limit its usage. One major problem with autonomous system is the lack of flexibility. In most autonomous systems, the operations that can performed are rather rigid and limited. One cannot request the system to do other tasks which the system is not designed for. Even for the same task, if the conditions or environment is slightly changed, the whole system needs to be

rebuilt from the very beginning, which usually requires a lot of man power and testing cycles, as well as the development cost.

2.3 2D Windowing Simulators

Simulation is the idea introduced to reduce the costs required for building autonomous robotic systems. It can be used to directly operate a system in a remote site. (e.g. operating the robot on Mars). The fundamental idea is to use computer system to predict and reproduce the behavior of a real system. For a long time, robot simulation environment has been limited to 2D windowing layout. Using 2D graphical tools, such as part display or control panel, the user can monitor the system and perform the task by moving the part or manipulator from one place to another.

However, this kind of operation is rather unnatural and inefficient. The main cause of the problem is 2D limitation. In the windowing simulation, one or more cameras are used to observe the robot behavior. The movement is then caught and projected to 2D output devices. To control the robotic system in 3D environment, the user has to divide the 3D control space into several separate 1D or 2D control planes and integrate the separated control effects manually. For instance, to move a robot to a goal position or orientation the user needs to specify the coordinates (x,y,z), one at a time from keyboard or slider. This usually involves special training of the operator and thus will increases the cost of operation in a reduced windowing control environment.

2.4 Robot Simulation in VR

Virtual reality (VR) is a newly emerged three-dimensional environment [38, 6]. It directly places the user in computer simulated worlds, in which the user can view and manipulate the surrounding objects with the movements of his/her hand(s) and body [41]. 3D input and output devices are usually used to provide a nature

interface between user and computer. Data glove devices are one of the three dimensional interactive devices introduced in virtual reality [36, 37]. Equipped with sensors to track both static hand shapes and dynamic hand movements, data glove devices have the advantages in creating simple human-computer interfaces than most conventional input devices, such as keyboard and mouse, for their highly coupled degrees of freedom, familiar sign language, and body reference coordinates.

The use of virtual reality in robot simulation is in its infancy. It is mainly due to the special VR hardware with high computation power, which is not commonly available and affordable until recent years. Research in VR is not abundant, but is increasing. Takahashi and Sakai [39] have proposed a virtual workspace to simulate the actual robot workspace, in which the user's movement is translated to manipulator commands that control the robot to perform the same task. A virtual environment for teaching robotic assembly operation is presented in [22]. Another research trend in the virtual robot simulation is teleoperation of robots [26]. Operating a robot in a remote site may result in time delays between the input commands and the robot reactions. These delays make real-time robot operations very difficult. To address the problem, Brunner *et al.* [4] has developed a telesensor-programming concept that uses sensory perception to locally control the robot. Virtual environments have also used in the development of robotic teleoperation for NASA's Space Station Freedom [2].

Generally speaking, VR environment provides the user a 3D vision of the system, which helps the user simulating the robot operations with the available 3D information. For example, the sense of distance is important in most robot applications. To better control the movement of a robot, we need to know the distance between the robot arm/hand and the mechanical model to be manipulated. This information is difficult to represent in a 2D window-layer environment. Moreover, 3D input devices, such as data glove devices and 3D mouse, provide direct and natural manipulation interface which allows the user perform the same robotic tasks in the simulation space. Usually, direct manipulation interface requires less

user training time when comparing to traditional 2D system. This can reduce the operating cost. Also, the control is much easier because the operator can control the robot in a similar way as the robot does.

Real-time interactive manipulation is another important feature in robotic simulation system. The system should allow the user to control the system and display the simulated results instantly to the user. To archive this, small latency is required or the user will have difficulty in controlling the system. A frame-rate of 15-30 frames/second is necessary for performing real time operations. This requirement may not be archived easily without using efficient algorithms and suitable user interface. In order to develop the common simulation framework across different platforms, different precision-control levels should be supported.

Chapter 3

Objective

The primary objective of this research is to create a real-time 3D interactive control system as a common simulator for performing the robotic manipulations. The following features are proposed:

- The same working environment should be modeled and experimental through the 3D input/output interface.
- Our system should allow the user to interactively specify the goal of a manipulator (i.e., the end position and orientation of a robot arm).
- The system should have instant response and small latency. Otherwise, it will create extra difficulty operating the virtual simulation mode.
- To satisfy both the requirements of high response speed and precise control, our system should provide mechanisms which adopt control algorithms to different precision levels of a manipulation task.
- Combine the strength of both kinematics and dynamics approaches and apply them to the same simulation system.
- Our system should provide basic manipulation functions and structural framework that can be commonly used in various robot applications.

- Feedback system is required to improve the efficiency and accuracy of the operation.
- Physical interaction, such as force and torque, between the manipulator and object is simulated.

Chapter 4

Articulated Structures

In this chapter, the articulated structures and representations used in this research is described.

4.1 Joints and links

To allow easy modification, a simple representation method of articulated structure is required. An articulated structure can be represented by a collection of **links** connected together by **joints**. Usually, the links are rigid objects, which cannot be deformed. Although some researchers is working on the control of robots with deformable links, but the word *link* mentioned in this thesis refer only to non-deformable links.

The joints in an articulated structure can be classified into three types: **prismatic**(translation) and **revolute**(rotational) or the combination of two. Figure 4.1 and 4.2 show an example of the first two types. A prismatic joint allows two links to move linearly relative to each other. It is usually used to extend or move certain parts of the structure. The drawer of a desk is a good example of such joint. A revolute joint allows rotation between two links. It can usually be seen in the structures which require the change of shape. Most of the joints in our body require such ability, which are good examples of revolute joint.

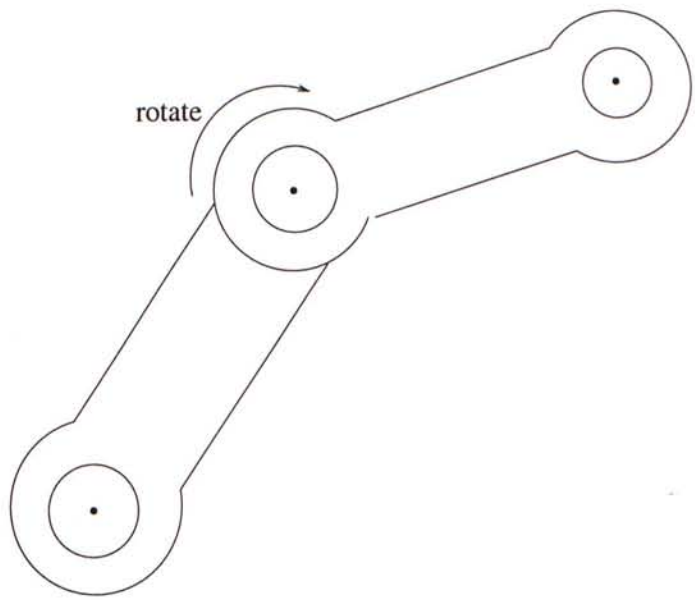

Figure 4.1: A revolute joint

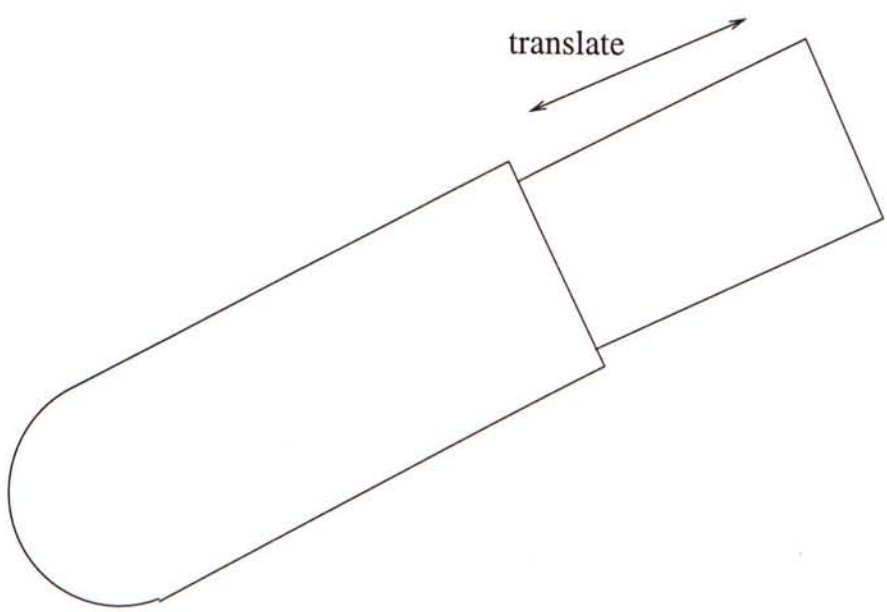

Figure 4.2: A prismatic joint

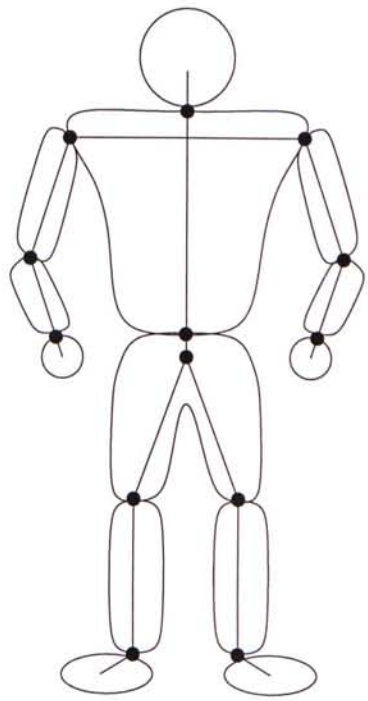

Figure 4.3: A tree-form articulated structure : body

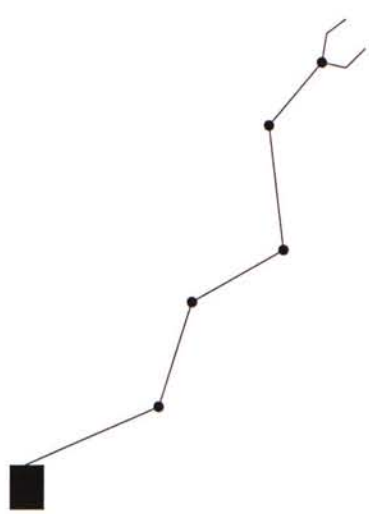

Figure 4.4: An N-link articulated structure

Articulated structures can be divided into two types: n-link structure and tree-form structure. Figure 4.4 is an example of n-linked structure. A general n-link structure consists of links which connected together by joints in series. This kind of structure can be used to model articulated body or parts, such as a robot arm and leg. Another type of structure is tree-form structure, which is the general extension of n-link structure. It consists of several n-link structures and each of them connects to a common base. An example of tree structure is human hand, as shown in figure 4.3. The arms, legs and head act as the 5 n-links structures, which are connected by the body. Although the motion of each link is independent, the links can be coordinated to collectively perform the same tasks, such as walking.

4.2 Degrees of Freedom

Theoretically, all joints can be rotated in 3 orthogonal directions and translated in 3 planar direction. The planar translation control the relative position of the connected links, while the orthogonal rotation control the relative orientation. However, it is rarely that a joint can rotate or translate in all 6 direction. It is because the movement of the joint is limited by constraint of the system and the allowable range of each joint. We called the allowable moving directions as the degrees of freedom (DOF) of the joint. For example, the legs of a human body can only rotate about 180 in 1 direction. Therefore, the degree of freedom is 1. If the articulated contains more than 1 joint, its DOF is calculated as the the total DOF of all joints in the body.

The degree of freedom is important for the movement control of a body. The higher the DOF, the more difficult to control the movement. It is because the movement of the joints usually depend on each other. Moving a joint required us to adjust other joints to balance the whole system, which is extremely complex in a body with lots of DOFs.

4.3 Denavit-Hartenberg Notation

To simplify the representation, each joint with more than 1 degree of freedom will be decomposed into several 1-D joints, which are either translational or rotational. Then, Denavit-Hartenberg(DH) notation [9] is used to describes the kinematics of each link relative to its neighbours by attaching a coordinate frame to each link. Following is the procedure to assign the coordinate frames to the links:

Algorithm 4.1: D-H Representation

1. Number the joints from 1 to n starting with the base and ending with the tool yaw, pitch, and roll, in that order.
2. Assign a right-handed orthonormal coordinate frame L_0 to the robot base, making sure that z^0 aligns with the axis of joint 1. Set k = 1.
3. Align z^k with the axis of joint k + 1.
4. Select x^k to be orthogonal to both z^k and z^{k-1} are parallel, point x^k away from z^{k-1}.
5. Select y^k to form a right-handed orthonormal coordinate frame L_k
6. Set k = k + 1. If k ¡ n, go to step 2; else, continue.
7. Set the origin of L_n at the tool tip. Align z^n with the sliding vector, and x^n with the normal vector of the tool. Set k = 1.
8. Locate point b^k at the intersection of the x^k and z^{k-1} axes. If they do not intersect, use the intersection of x^k with a common normal between x^k and z^{k-1}.
9. Compute θ_k as the angle of rotation from x^{k-1} to x^k measured about z^{k-1}.
10. Compute d_k as the distance from the origin of frame L_{k-1} to point b^k measured along z^{k-1}.
11. Compute a_k as the distance from point b^k to the origin of frame L_k measured along x^k.
12. Computer α_k as the angle of rotation from z^{k-1} to z^k measured about x^k.
13. Set k = k + 1. If k $\leq$ n, go to step 8; else, stop.

Four parameters are used to define a linear transformation matrix between consecutive coordinate systems attached to each joint. The four parameters are described below (see figure 4.5 and figure 4.6 for the correspondence):

- a_k is the distance from $\mathbf{z_k}$ to $\mathbf{z_{k+1}}$ measured along $\mathbf{x_k}$ - the length of the link.
- α is the angle between $\mathbf{z_k}$ and $\mathbf{z_{k+1}}$ measured about $\mathbf{x_k}$. This is the twist of the link.
- d_k is the distance between the $\mathbf{x_{k-1}}$ and $\mathbf{x_k}$ axes measured along $\mathbf{z_k}$ - the distance between links.
- θ is the angle between $\mathbf{x_{k-1}}$ and $\mathbf{x_k}$ measured about $\mathbf{z_k}$.

The k-th coordinate frame is therefore characterized by the four Denavit-Hartenberg kinematic link parameters. For a rotational joint, α_k, d_k and a_k are constant while θ_k is changed along the z axis. For a translational joint, d_k, α_k, θ_k remains constant while a_k is changed along the x-axis. These information can then be used to calculate homogeneous transformations between link coordinate frames. To change from k-1th frame to kth frame, the following four steps are required :

- rotate about $\mathbf{z_{k-1}}$ an angle θ_k
- translate along $\mathbf{z_{k-1}}$ a distance d_k
- translate along rotated $\mathbf{x_{k-1}} = \mathbf{x_k}$ a length a_k
- rotate about $\mathbf{x_k}$ an angle α_k

The four parameters of DH notation form the basis of link coordinate frames.

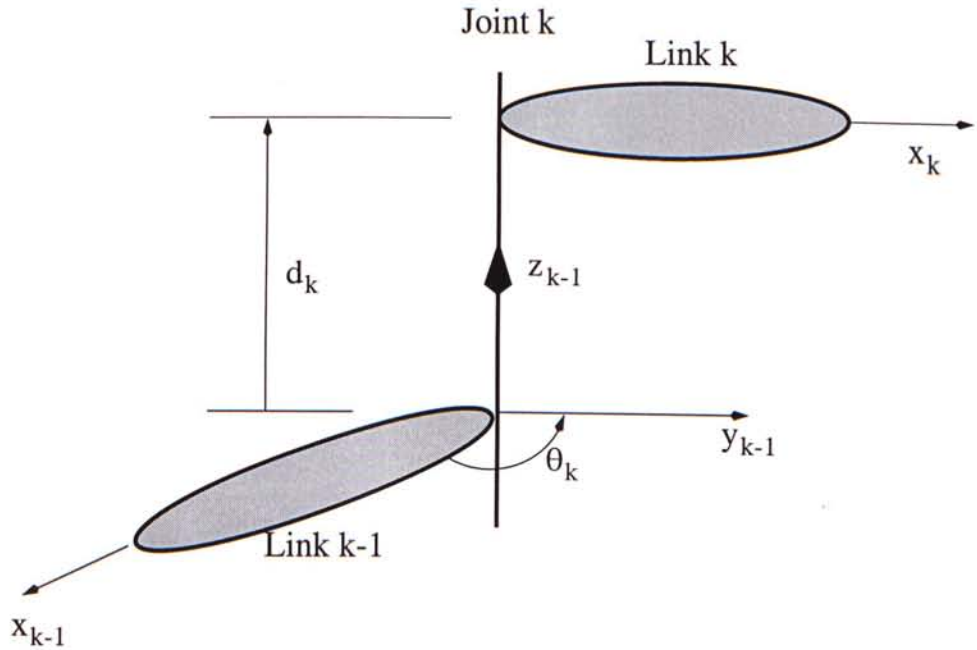

Figure 4.5: DH notation : joint angle θ and joint distance d

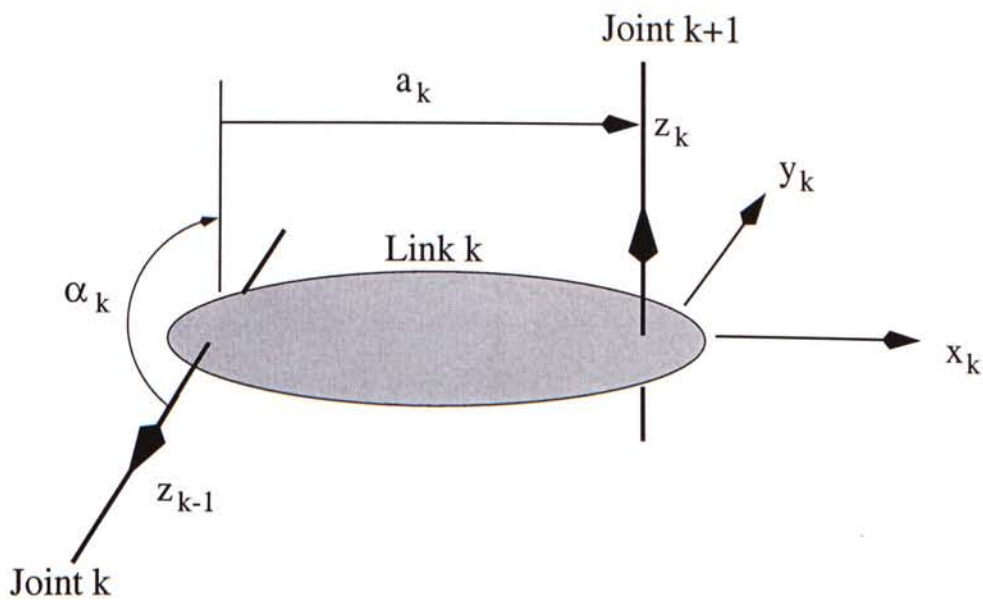

Figure 4.6: DH notation : link length a and link twist angle α

Chapter 5

Virtual Manipulators

In the last section, the method to describe a general articulated structure is discussed. In this section, we will present two examples of manipulator. The first one is arm structure, with 3 links collected by 6 rotational joints. It is an example of a general N-link structure. This model is used to simulate how an arm structure is moved, rotate or manipulate other objects. The second example is hand structure, with 5 fingers connected to a common base. It is an example of a tree-type structure. This model will be used in our experiment to test how hand-oriented manipulation can be produced in an interactive system.

5.1 Arm(N-link) Structure

Figure 5.1 shows the outlook of a 3-link robot arm. The whole-arm structure consists of 3 joints, which will be decomposed into 6 1-D joints. The 6 DOFs can be divided into two groups, each with 3 DOFs. The first 3 DOFs distribute among the shoulder (2 DOF) and elbow (1 DOF). These DOFs control the position of the end-effector(hand) in the 3-D environment. Changing these joint variables will not affect the orientation of the hand. The next 3 DOFs appear at the wrist position. They control the orientation (yaw, pitch, roll) of the hand. During the calculation, we can divide the DOF into two groups to reduce the computation complexity.

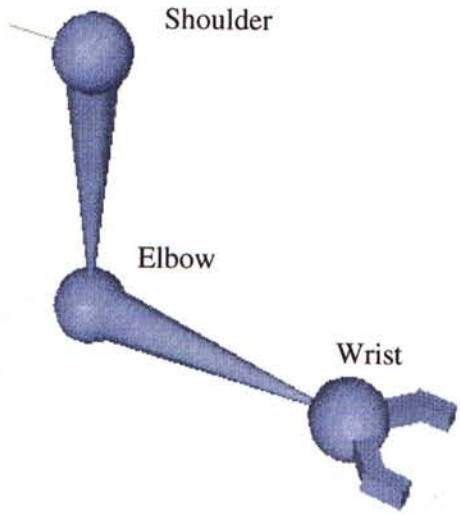

Figure 5.1: Configuration of a robot arm

We can represent the structure using DH notation mentioned in Section 4 and use the parameters to calculate the corresponding coordinate frame of each joint. The following table shows the parameters of each coordinate frame:

Frame	θ	d	a	α	Home
L_1	θ_1	0	0	90	-90
L_2	θ_2	0	a_2	0	0
L_3	θ_3	0	0	90	0
L_4	θ_4	d_4	0	-90	0
L_5	θ_5	0	0	-90	90
L_6	θ_6	0	0	90	-90

Table 5.1: Parameters of each coordinate frame

The Home column is the initial value of θ_n. It specifies the normal position of the joint. Figure 5.2 shows a detailed view of the coordinate frames of the 6 joints. Note that dotted lines between the origin of L_2 and L_3, L_4 and L_5, L_6 indicate that the origins of these coordinate frames coincide. They are drawn in

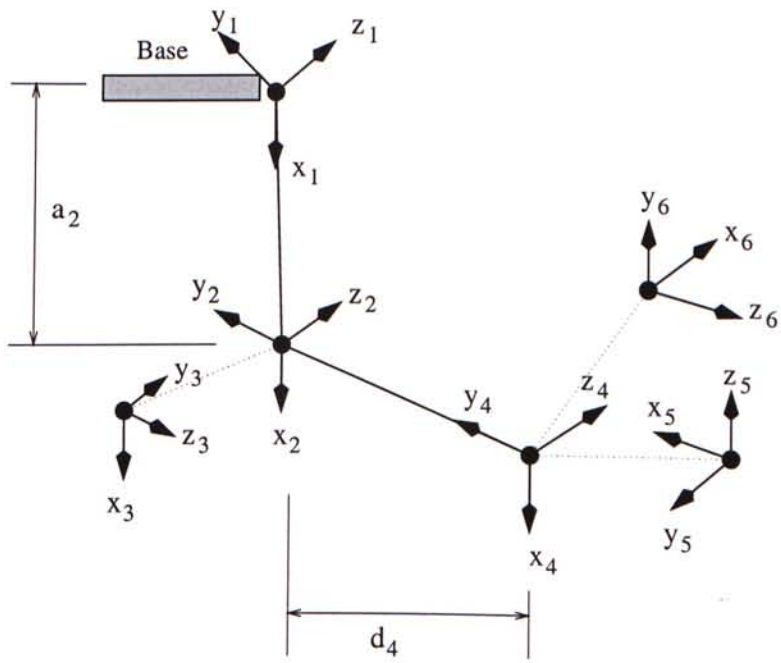

Figure 5.2: Coordinate frames of the joints

separate space in order to make the diagram more clear. a_2 and d_4 are the lengths of two links.

In this structure, we can interactively specify the location and orientation of the goal of end-effector. According to the input of the user, structure is moved using different motion control techniques. These techniques affect the flexibility and accuracy of the motion, and will be discussed in the next section.

Figure 5.3 shows a case of goal specification. Coordinate frames of the structure and the goal is draw to the figure for reference. First, a goal position and orientation is specified by the user. The system reads this information from the 3D device and calculates the required movement. Figure 5.4 shows the intermediate frames of the motion towards the goal, in which the last frame is drawn in solid rendering while in-between frames are rendered in wire-frame.

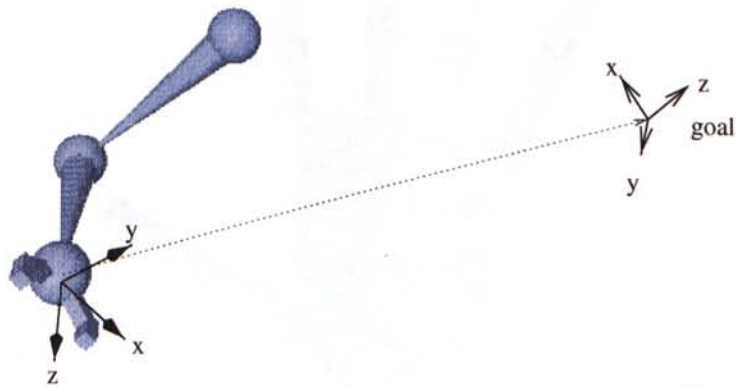

Figure 5.3: Goal configuration of a robot arm

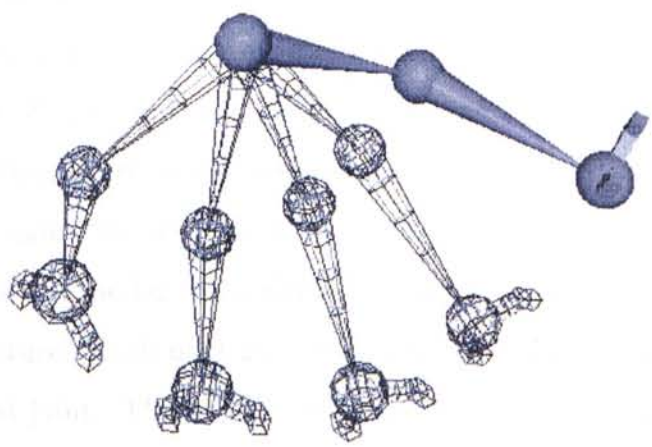

Figure 5.4: Motion frames towards the goal

Figure 5.5: Outlook of simplified hand model

5.2 Hand Model

Hand is one of the most important parts in our body. We use it to manipulate other other objects around us. The advantage of the hand lies in its flexibility and dextrousness. It contains more than 20 DOFs, which allow us to performs most complicated operations. However, due to the complexity of our hand, it makes the simulation of our hand very difficult. Instead of simulating the hand in full manner, simplified hand models are usually used.

Figure 5.5 shows the outlook of the virtual hand model and figure 5.6 shows the structure of our model. Basically, the four fingers, other than the thumb, has the same structure. Each of them consists of three links, connected together by a 1D rotational joint. The fingers are connected to the palm by the base joint, which has 2 DOF. The additional DOF comes from the abduction and adduction of the finger. Abduction and adduction refer to the side movements of the finger. Adduction moves the fingers toward each other and abduction is the vice verse. The only different between the thumb and other fingers is that it does not has the third link, which make the thumb has one less degree of freedom than the other

fingers. Table 5.2 gives a summary of the DOF of hand .

Joint		DOF
Thumb	1st joint	1
	base joint	2
Fingers(x4)	1st joint	1
	2st joint	1
	base joint	2
Total		19

Table 5.2: The degrees of freedom of hand joints

A new coordinate system is assigned to each joint of the fingers. Figure 5.7 shows the configuration of the coordinate frame of a finger. All coordinate frames are assigned at the joints, with its x-axis pointing towards the next joint or the finger tip. The y-axis is assigned as axis of rotation and all joints rotate along it. The z-axis is assigned to the same direction as the cross product of x-axis and y-axis, which usually pointing away from the palm.

The base joint of all fingers are divided into two 1-D joints. One is the folding joint, which is the same as the other two joints of the finger. The other is the abduction joint, which control the angle between each fingers. The abduction joints rotate along the z-axis of each base joints.

A coordinate frame is assigned to the wrist of the virtual hand. It is used to represent the position and orientation or user hand. The x-axis of this coordinate frame is pointing towards the tip of fingers, which the z-axis is pointing along the direction which the palm facing. The y-axis is the cross product of the other two axes.

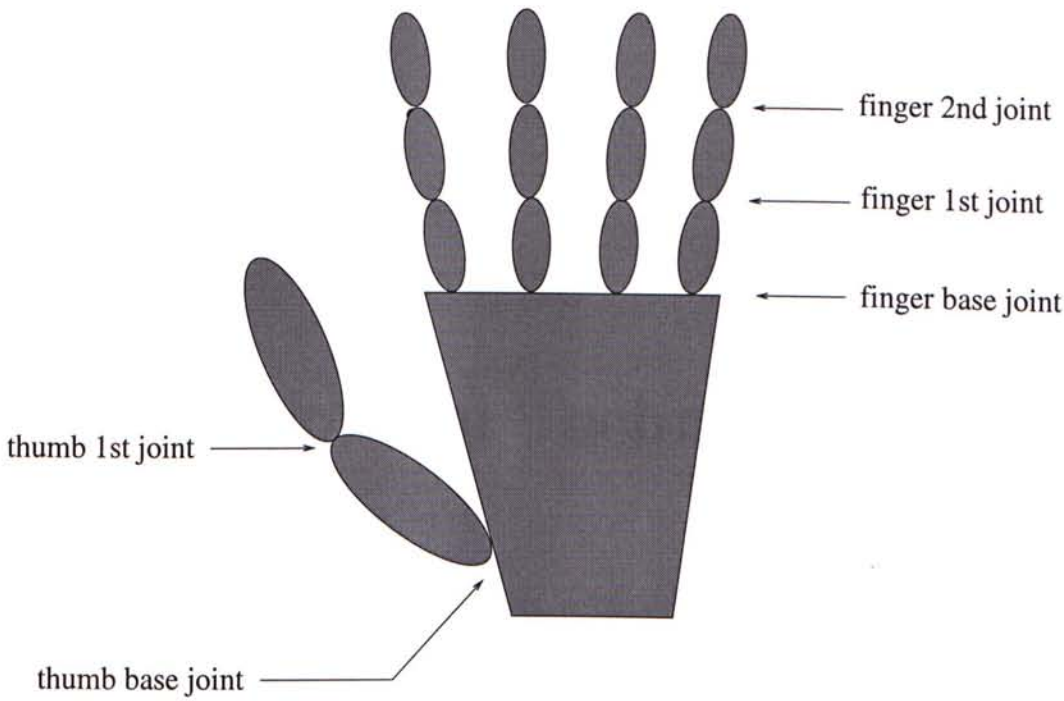

Figure 5.6: Structure of hand model

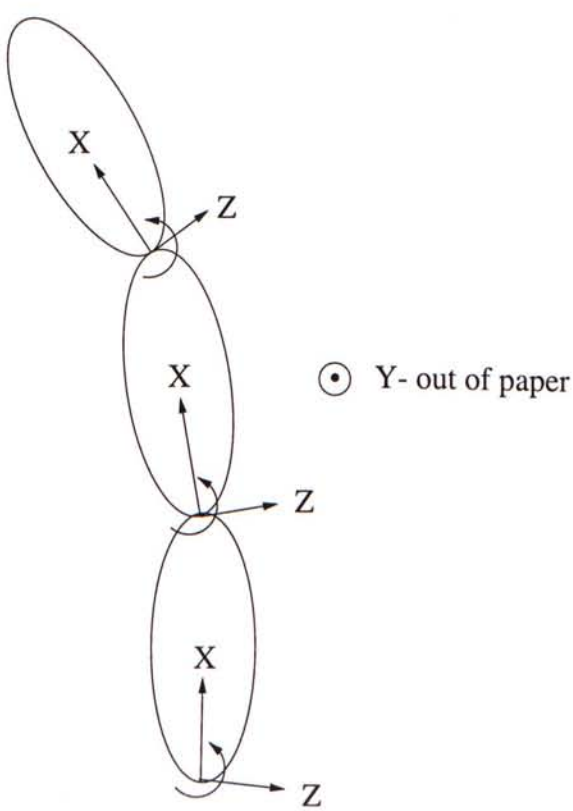

Figure 5.7: Configuration of coordinate frame of one finger

Chapter 6

Motion Control Techniques

Two major control techniques, kinematics and dynamics, have been used in motion control. Both techniques can be applied in either forward mode or inverse mode, which derives motion from either the joints or the end-effector. In the following, both techniques and each of the control modes are described in detail.

6.1 Kinematics

For several decades, kinematics [10, 11, 18] has been studied extensively in the robotic field. It mainly focuses on the geometrical relationship between the joints and the end position and orientation of a manipulator. The motion of a linked structure is purely determined by the joint parameters, which is θ for rotational and a for translational. To move the structure, the the joint variables will be changed to meet the requested goal configuration. Two modes of control, forward kinematics and inverse kinematics, can be used to guild the motion and determine the change required for the goal.

6.1.1 Forward Kinematics

Forward kinematics involves explicitly setting the position and orientation of objects at specific frame times. For our articulated structure, this means directly

setting all joint variables. To simplify the work and avoid setting the variables repeatedly for each frame, a series of key-frames can be specified at different frames, and the system interpolates the intermediate frames by the joint parameters between the key-frames [34]. Among the interpolating methods which can be used to generate the motion, linear interpolation is the simplest. However, linear interpolation cannot guarantee the continuity of first derivatives at the key-frames. This may not be a problem for most applications. However, when the system is applied to robotic simulation, such a jerky motion is not desired. Therefore, a higher-order interpolation method, such as piecewise splines, can be used to provide continuous and smooth motion to the system.

In general, forward kinematics is not suitable for interactive robot control. One major difficulty is the complexity problem. Even for a very simple structure, its DOFs can easily go over 10. For some complex structure, like human body, its DOFs can be over 100, which is impossible for the user to control interactively. Even supposing that the number of degrees of freedom within a figure is manageable, it is difficult to control the motion of the joints, especially rotational joints. Unlike translations, an ordered series of rotations do not combine intuitively, making it difficult to predict the consequence of editing a single rotation trajectory. It is almost impossible to decide on the appropriate changes to all three rotations (X, Y and Z directions) which will produce a desired change in a single body segment's motion.

Different approaches can be used to improve the efficiency of forward kinematics. Keyframe-based control is used in the making of computer animation. The animators only require to create the keyframes of the motion of the character, while the computer is used to calculate the intermediate steps. This can significantly reduce the amount of work required. However, keyframe-based control is not suitable to use in an interactive environment. In most interactive application, the user need to modify the motion of the structure according to the situation. It is difficult to pre-set the key-frame beforehand.

6.1.2 Inverse Kinematics

Using forward kinematics, it is difficult to control the position of any parts in the structure. The position and orientation of the parts can only be controlled indirectly by specifying the joint variables between the root and the parts itself, which is tedious and non-efficient. Inverse kinematics, on the other hand, is used as an alternative solution of above problem. Inverse kinematics provides direct control over the placement of an end-effector at the desired location. To change the configuration of the manipulator, user can directly specify the position and orientation of end-effector, while the system automatically computes the joint variables required to meet the requirement. It is not surprising that the inverse kinematics problem has been studied extensively in the robotics field, although it is only recently that the techniques have been adopted for the VR environments.

Using inverse kinematics, the required change of intermediate joints to fulfill certain goal are calculated automatically without the aids from users. The calculation is based on the geometrical and kinematical relationships between the links. Compares with forward kinematics, inverse kinematics significantly reduces the control details and time required to create the desired motion. This makes interactive control possible using kinematics method.

6.1.3 Solving Kinematics Problem

At each joint of an articulated structure, a coordinate frame is assigned. In the last chapter, the method to assign coordinate frame to each joint is discussed. The transformation matrix M_i is used to transform the coordinates in frame $i-1$ into frame i. According to chapter 4, M_i is consisted of four transformations. That is,

$$M_i = R_{x_i}(\alpha_i)T_{i-1}(a_i, 0, 0)T_{i-1}(0, 0, d_i)R_{z_{i-1}}(\theta_i) \tag{6.1}$$

where R_{axis_n} is the rotational matrix about $axis$ of frame n and T_i is the translational matrix at frame i. Combining all the transformation matrix, we can find the matrix $M = M_n M_{n-1} ... Mi ... M_2 M_1$, which relates the position and orientation

of the end-effector to the base.

Given a vector q in the control space of the structure, the position and orientation vector $\mathbf{x}$ of the end-effector can be found by *forward kinematic* using the following equation,

$$x = f(q) \tag{6.2}$$

where $\mathbf{f}$ can be found by M. On the other hand, given the position and orientation vector $\mathbf{x}$, *inverse kinematic* is used to find the joint variables of the intermediate joints. It requires us to solve the inverse of the equation 6.2,

$$q = f^{-1}(x) \tag{6.3}$$

However, due to the nonlinear property of function $\mathbf{f}$, it is difficult to find its inverse. Also, although we can find a unique mapping from q to x in equation 6.2, the same cannot be said for the inverse mapping of 6.3. This is because the structure may contain redundant degree of freedom, and it will be discussed in the next section.

A common method to solve the inverse kinematics problem is linearize the problem about the current structure configuration, and the relationship between joint velocities and the velocity of the end-effector is,

$$\dot{x} = J(q)\dot{q} \tag{6.4}$$

where the linear relationship is given by the *Jacobian* matrix $\mathbf{J}$,

$$J = \frac{\delta f}{\delta q} \tag{6.5}$$

$\mathbf{J}$ is an $m \times n$ matrix, where n is the number of joint variables and m is the dimension of end-effector vector $\mathbf{x}$. Inverting the equation 6.4 allow us to solve the inverse kinematics problem

$$\dot{q} = J^{-1}(q)\dot{x} \tag{6.6}$$

If we can found the inverse of $\mathbf{J}$, we can compute the incremental changes in joint variables from the incremental change in the end-effector position and orientation.

Base on equation 6.6, we can solve the inverse kinematics problem by a simple iterative scheme. At each iteration, $\dot{\mathbf{x}}$ can be computed from the current and desired end-effector positions. Then, the joint velocities $\dot{\mathbf{q}}$ can be computed using the Jacobian inverse, and integrated once to find a new joint state vector $\mathbf{q}$. Since $\mathbf{J}$ is only valid for small perturbations in the structure configuration, $\mathbf{J}(\mathbf{q})$ must be recomputed at each iteration. The same calculation process is repeated until the desired goal is reached. However, the above scheme base on the fact that the Jacobian matrix is invertible. This assume that $\mathbf{J}$ is both square and non-singular. Unfortunately, this assumption is generally not valid. Problems arise when the articulate structure contain *redundant* degree of freedom, or when it passes through or near a *singular* configuration.

6.1.4 Redundancy

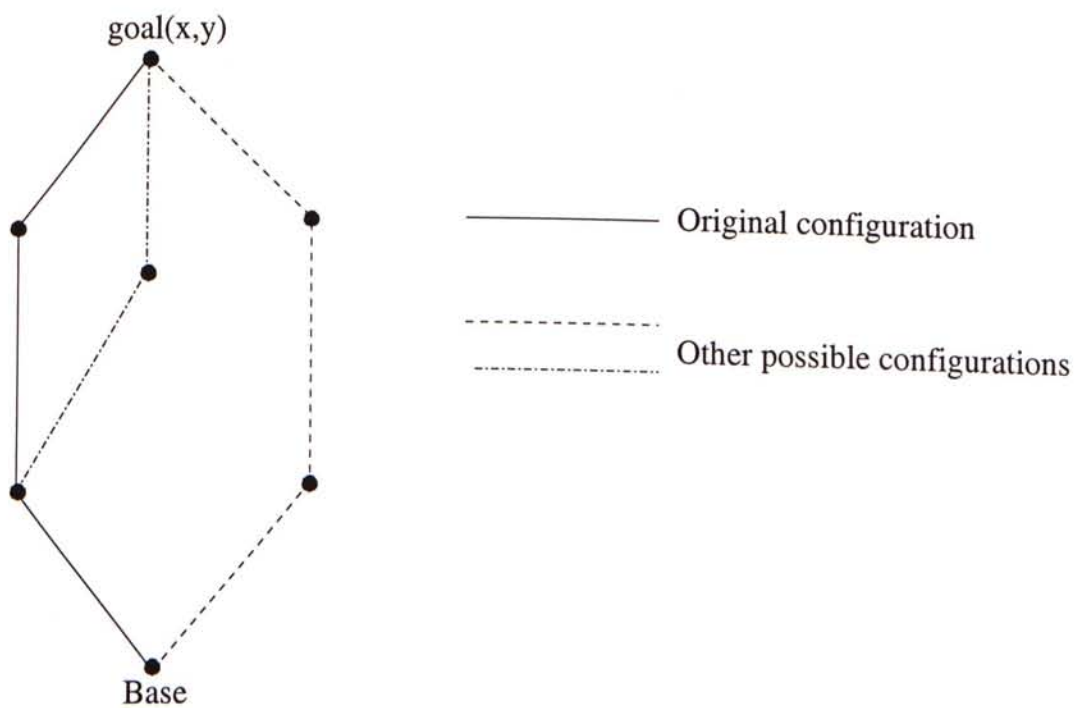

Figure 6.1: Three configurations of a 2D redundant manipulator

A manipulator is considered kinematically redundant when it possesses more degree of freedom than are required to specify a goal for the effector[28]. For example, consider the simple 2D case in Figure 6.1. The structure has 3 degree

of freedom, which the links can rotate above each joint. The position of the end-effector (goal) can be changed by adjusting the rotation angles at each joint. However, to specify the goal of end-effector on the 2D plane, only 2 DOF is required. As the figure shows, for a given goal, it can be archived by more than one configuration. Therefore, the structure is called redundant.

For a redundant structure, the Jacobian matrix has fewer rows than columns, which means it cannot be inverted. In this case, $\mathbf{J^{-1}}$ is replaced by some generalized inverse $\mathbf{J^{+}}$. One such generalized inverse is the Moore-Penrose *pseudo-inverse* [12]. It has been shown [16] that this pseudo-inverse yields solutions with a minimum Euclidean norm for cases in which equation 6.6 is under-determined ($m < n$), and that in cases in which the system is over-determined ($m > n$) a least-squares solution is obtained. This ensure that the joints move as little as possible to match the desired end-effector velocity as closely as possible.

Redundant is sometimes necessary. For example, if we want to get an object around the corner by a human-like robotic arm, we must extend the arm so that the arm will not hit the wall. In this case, redundant is required to avoid the obstacle. In general, extra degree of freedom add flexibility to the manipulator, which allows it to reach objects around the obstacle and manipulate an otherwise inaccessible object.

6.1.5 Singularities

Another problem of inverse kinematics is *singular*. A matrix is called singular when two or more of its rows are linearly dependent, and a structure is said to be in a singular configuration when the Jacobian becomes singular. A singular configuration usually appears when the structure reaches the workspace boundary or two or more joint axes lining up. When a structure is in a singular configuration, it has lost one or more degree of freedom. This means that there is some direction (or subspace) in Cartesian space along which it is impossible to move the hand of the robot no matter which joint rates are selected.

Figure 6.2 shows a structure in a singular configuration. In this example, an

incremental change to any of the joint angles will result in approximately the same movement of the end-effector in the y-direction. No combination of joint velocities will move the end-effector along the singular(i.e.x) direction. The Jacobian matrix for this case will contain zeros in one it the rows, and is therefore singular and cannot be inverted.

Similar to redundant, pseudo-inverse can be applied to obtain a useful solution when **J** is singular. However, when the structure approaches this configuration, the pseudo-inverse tends to produce large joint velocities. This may create discontinuities in control space and instability to the system.

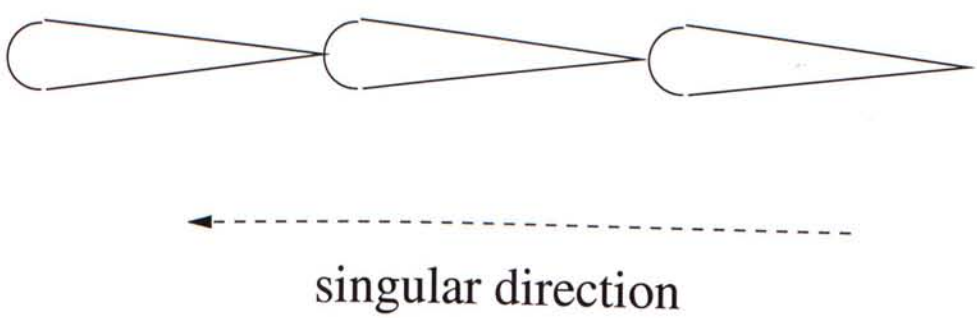

Figure 6.2: A structure in a singular configuration

6.2 Dynamics

Dynamics [1, 43] is another major motion control technique. For dynamic analysis, object attributes including center of mass, total mass, the moments and products of inertia, are required in object description. The motion of the object is affected by the forces and torques which are applied to different parts of the system. Dynamics method is to simulated the actually physical interaction happened in the simulating environment. Although there are many formulations for the equation of motion, two methods are most commonly used. The *Newton-Euler* approach is based on the elementary dynamic formulas and on an analysis of forces and moments of constraint acting between the links, which can be called as a "force balance" approach. As an alternative to the Newton-Euler method,

the *Lagrangian dynamic* formulation is proposal base on the "energy-based" approach. By the conservation law of energy, the total energy of a system is always a constant. The Lagrangian dynamic formulation provides a means of deriving the equations of motion from a scalar function called the Lagrangian, which is defined as the difference between the kinetic and potential energy of a mechanical system. This function is in terms of angular and translational velocity and acceleration of each parts of the structure. Using this function, the resulting motion can be determined.

6.2.1 Forward Dynamics

Similar to Forward Kinematics, Forward Dynamics involves explicit application of time-varying forces and torques to objects. Some forces, such as those due to gravity and collisions between objects, may be handled automatically. Other forces are applied directly to objects in the environment by the user. At each discrete time steps, the motion is approximated by solving the equations of motion for the acceleration an object undergoes in response to the applied forces. These applied forces can be constant force, such as gravity, or time-varying force, such as user acting forces, or impulsive force, such as collusion forces.

Applying this approach to interactive motion control is very difficult. Usually, the required force that should be applied to the structure in order to get the desired motion is not explicitly known. The only way for a user to control the motion is by trial-and-error, which makes the control extremely unstable and imprecise. Also, when the structure becomes more and more complex, it is nearly impossible for the user to control it interactively using this method. It is often the case that the user applies the force or torque leading to a wrong position/orientation or cannot apply the suitable force on time to generate a desired motion. Therefore, the user cannot control the position and orientation of the end-effector efficiently using this approach. In autonomous robot systems, trial and error approach is used to apply force or torque on the structure to get the desired result, which is very time consuming and costly.

6.2.2 Inverse Dynamics

Inverse dynamics method automatically determines the force and torque functions needed to accomplish a specified goal. The goal can be a directed motion of the structure towards a particular position and orientation of the end-effector. Similar to forward dynamics, the motion of the structure is calculated by physical laws and attributes acting on each linked segment. While deriving the motion, interactions between the links and the force and torques applied between intermediate joints are calculated.

The computation of inverse dynamics is rather costly. For each degree of freedom of the structure, there will be one equation of motion. This usually leads to a large system of equations, which must be solved by numerical method at considerable computational expense. In general, dynamic simulation of complex articulated structure cannot be performed at interactive speed on a singer-processor machine. Although the processing power of available machines becomes more powerful, it is still difficult to use pure dynamics approach for a complex system.

6.3 Combination of Two Control Modes

Kinematics uses the geometrical relationship between links, which involves the use of Jacobian matrix during the process of velocity calculation. During the calculation, J^{-1} is necessary to be found [45]. However, Jacobian matrix is usually not square. In this case, $pseudo - inverse$ techniques, which is an approximation technique, is used to find the solution. It is almost certain that errors will be introduced during these operations.

Dynamics bases on physical laws and properties to derive the object's movement, instead of positioning the object by geometrical transformations. Using forces and torques, and physical properties, the required motion of the system is calculated. As a result, the motion is produced more physically accurate, and appears more attractive and natural. There are many types of motion, such as pushing and reacting to collisions, which can be automatically calculated from the

dynamics environment, but not in kinematics environment. However, the physical realism comes with a cost of heavy calculations. A general system with 6 DOF can result in a large equation set, not to mention a more complex system. In comparison with dynamics, the calculations involved in kinematics are much simpler, which can greatly reduce the time required to find the solution.

Kinematics is fast in calculating end position and orientations, while dynamics is natural in calculating the interactions between objects. Our goal is to combine the strength of both approaches. When the structure is moving in a free space, without interacting with other objects, we apply kinematics method to get a fast calculation of required end positions and orientation. When the structure interacts with other objects, such as picking, dropping, or colliding, dynamics calculation is used. By finding the responding forces and torques between objects, we can simulate the physically-accurate motion. Usually, the interaction between objects lasts only short period of time, like several mini-seconds for the case of collision. After the interaction is produced, we can switch back to kinematic mode to speed up the calculation. The system will not suffer from serious latency problem for a long period of time.

6.4 Constraints and Optimization

In most interactive control system, we need to specify the constraints of the system [44]. These constraints are arose usually because of the physical configuration of the system. We may also apply constraints to the system so that we can limit the movement of the system. This can reduce the time needed to find the solution and minimize the chance of machinery failure.

Another reason to apply constraints is to avoid the case of ill condition. Under some configuration like full extension, the system will lose some degree of freedom. In such cases, the system will become unstable and the behavior will become more erratic. Applying constraints to the system can prevent the system going into ill conditions.

Another thing to consider is optimization. Sometimes, redundancy is introduced to the system so that the system can reach around an obstacle and manipulate an otherwise inaccessible object. For a given goal there are more than one solution; each of the configurations will place the end-effector at the goal position. In this case, the system needs to select the best solution. How a optimal solution is selected is very difficult in most cases. Even the same structure can have different optimal solutions under different situations. In general, we want to have a solution that satisfy the goal while requiring minimum joint movement. How the solution should be optimized is important in interactive motion control.

Chapter 7

Physical Feedback Systems

To provide an interactive virtual environment, the reaction between the objects and virtual manipulator must be considered. Feedback mechanism is created due to this purpose. When user performs certain action, the system will determine the result of the action and simulate it, which will immediately feedback to the user.

A good feedback system requires a small latency to provide good interaction between the system and the user. Also, the system should not require heavy CPU computation, so that the whole system will not be slow down. In this chapter, two kinds of physical feedback systems are considered. They are touch and force feedback.

Tactile feedback is one of the most basic sources of information from our surrounding environment. We use our hand (or other parts of the body) to touch and feel the object. It is very important because it tells us the nature of the object, whether it is soft, hard, smooth, rough, round, etc. The information requirements of many tasks needing dextrous manipulation and sense of touch is not met without tactile feedback. Also, the reaction force from the object gives us the information about the structure of object. A simple object (like a box) will give a single direction (parallel to the normal of the surface) reaction force, while a complex articulated structure will give multiple reaction force together with torques.

Recently, people have come to understand that force/touch feedback can be very useful. In the situations where the field of view is occluded or dark, the users need to get the information of the environment by their sense of virtual touch. Also, it is a must for some applications, like training of surgeons on virtual bodies, to let the user to "touch" the object they manipulate with.

7.1 Touch Feedback

In [30], the author identified five main approaches for finger touch feedback through visual, pneumatic, vibro-tactile, electro-tactile and neuromuscular stimulations. A newer approach using multi-modal "enhanced" feedback will also be discussed.

Pneumatic use micro air pockets which placed in a glove to provide touch feedback. This approach was used in the design of the "Teletact Glove" [35]. 20 air pockets are located in different positions inside the glove, mostly finger tip and palm. A proportional control interface is used to drive the inflation and deflation of the air pockets. Air pressure necessary for the feedback is obtained with a small compressor placed in the control interface. The glove is used to generate simple tactile patterns when users grasp real objects. These patterns are then transmitted and sensed by the person wearing the tactile-feedback glove.

Vibro-tactile use voice coils as the source of touch feedback. The voice coils were driven at high frequency and the amplitude of the vibration is determined by the force required. Experiments [23] has shown that in a simple two-finger scenario, the use of tactile can improve the performance of the work by 10-30%.

A variation to Vibro-tactile is to use micro-pin arrays instead of voice coils. The system changes the pattern of "active" micro-pins to simulate edges, holes or other surfaces of the object. Figure 7.1 shows the situation when the finger move along the edge of a virtual object. When the finger move from right to left (figure a and b), the pattern of the pins changed to provide tactile feedback to the finger. The quality of the simulation depends on the density and pattern of

the micro-pins array.

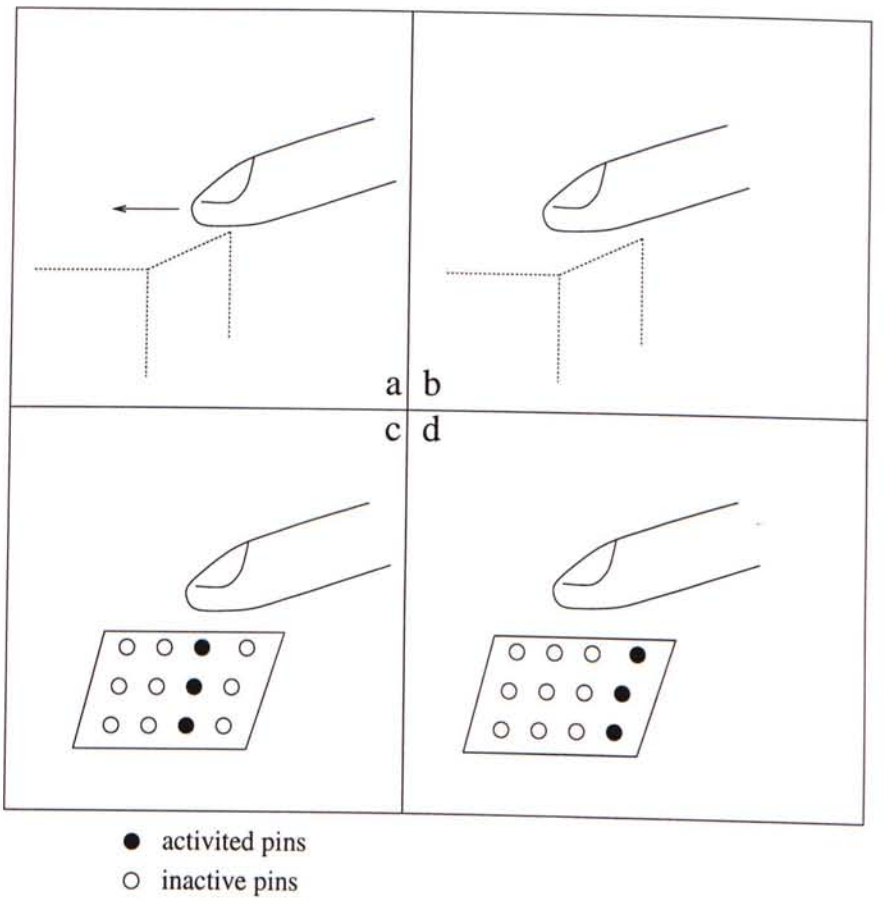

Figure 7.1: Tactile feedback of object edge by micro-pin arrays

Electro-tactile feedback uses electric pulses with varying width and frequency to simulate the skin of the fingers. Neuro-muscular stimulation provides the signal directly to the user's primary cortex. Due to the natural of these two techniques, they can be very dangerous and harmful to the user.

Research has been done to produce enhanced tactile feedback by adding temperature and thermal conductivity feedback [8]. Temperature information can help identify the natural of the virtual object. In addition, "pain" feedback (by very high or low temperature) can be implemented by this method.

7.2 Force Feedback

In addition of viewing the result of simulation, force feedback provide real force reaction to the user. For instance, when the user holds an object in his hand, he/she should feel the weight of the object.

Force feedback is different from touch feedback in several aspects. Touch feedback focuses on providing the surface information (like smooth, rough, edge, hole, etc.) of the virtual object, while force feedback provide the total contact force (like mass, fiction, etc.) which the object applied on the user.

An earlier research [17] has designed a force feedback system called "Master arm". This system focuses on simulating the weight of an object, its inertia and its contact with stiff walls. Although the system composed of large mechanic components, it is gravity and inertia compensated so that no forces are felt at the handle as long as there is no interaction with the virtual environment. However, due to the complexity of the system, the cost of it is very high and it is difficult to more the system around.

Force feedback joystick is another approach taken by Schmult and Jebens [27]. Force is applied when the object controlled by the user is contacted with other objects. The joystick can produce up to 75g of force on each axis (X and Y). The handle motion has a resolution of 3201 parts, which is enough to produce a number of force and tactile sensation such as direct forces, impulses, vibrations and change in stiffness. However, the system can only work in 2-D, which is the major disadvantage of it.

To solve the problem of 2-D working space, Iwata [15] has developed a "six degrees of freedom pen based force display". The system has two three degree of freedom arms placed in parallel, which each connected to one end of pen-shaped handle. By controlling the direction of force applied by the two arms, translational forces and torques can be produced on the user hand. The advantage of this system is extremely intuitive to use. Also, it is desktop based, has six degrees of freedom and a large work envelope than simple joysticks.

Another force feedback created by Burdea and his colleagues [5] is called Rutgers Portable Master. The feedback structure uses four pneumatic micro-cylinders placed in the palm of a glove on a small L-shaped platform. Each actuator has a conical work envelope which allows both flexion and aduction/abduction of the fingers. When the virtual hand grasped objects such as virtual rubber balls or soda cans, users could feel forces on their fingers.

7.3 Force/Touch Feedback Systems

Although touch and feedback system are very useful, a good feedback mechanism is not easily available. Here are the disadvantages:

- special devices require by force feedback usually increase the price of the system significantly.
- training is required to make the user familiar with the system
- additional mechanics providing feedback will reduce the manipulation range.
- due to the complexity of human body, it is very difficult to create a system which can cheat our sensory organism.
- limited by current technology, most force feedback systems can only provide a limited set of force simulator, which is not enough to simulate the force interaction completely in the virtual environment.
- it is difficult to modify the requirement or goal of the system
- when system failure occurs, the system may generate a very large force and injure the user.

Chapter 8

Virtual Object Manipulation

Object manipulation is one of the most basic operations in virtual reality. One of the aims of our research is to provide an user friendly interface to the user. Current research efforts [19, 20, 33, 7] are to develop the virtual environment which allows the users to control the objects by their hands. In the last chapter, different kinds of feedback systems has been discussed. However, all the systems has their limitations and are not suitable to be used as interactive control interface. Instead, we prefer to enhance the operator's visual feedback.

It is a tedious task to manipulate the virtual objects by means of digital glove which often carries mechanical errors and sensing noise. So the user has difficulty using sensed hand to precisely control the objects in the virtual world. Without the tactile/force feedback, the user can hardly adjust the hand posture in interaction and the forces they should apply in controlling the manipulation.

To solve the above problems, different methods have been proposed. One of them is posture/gesture recognition [36]. Based on some pre-defined commands, users can use few recognized postures/gestures to control the virtual objects. However, gesture recognition has is problem [42]. When the virtual environment becomes complex, the number of posture commands also increases,which demands the power of hand recognition system. Even the system has sufficient computation power, the user may have difficulty in remembering all the commanding postures. Also, The posture-command approach does not support the contact control in

object manipulation.

Rezzonico and Ronan [25] proposed an approach of interactive grasping where the hand posture is directly used to establish the grasp on the virtual object. The user do not need to remember the specific posture to express a grasping command. However, the main limitation of this approach lies in the lack of finger manipulation of the object relatively to the hand coordinate system. Once the grasping is established, the object is rigidly linked to the hand.

In a recent article of Ronan and Rezzonico [3], they suggested a complementary approach. They developed a virtual contact model, which allows the virtual object to be rotated or moved according to the movement of the fingers. However, the limitation of this model lies in the lack of force interaction between the hand and the object. Also, using this model will create some odd configuration of the fingers, like over bend the finger towards the back of the hand.

In this section, we propose an visual correction approach, which based on the work of Ronan and Rezzonico. We focus on providing an accurate and easy-to use interface in the virtual hand simulation. We propose a hybrid control approach that uses both kinematics and dynamics methods at different stages of picking to generate physically-accurate hand interactions in real time. The main advantage of this approach is that it does not require any special hardware but a sensing DataGlove. This approach utilizes both the visual feedback in contact interaction and physics-based simulation in hand picking. It can reduce the cost of additional hardwares and prevent the problem of limited control space in the feedback system.

8.1 Previous Work

The main theme of Ronan and Rezzonico's approach is to maintain the hand posture by unfolding (correcting) the hand so that the finger positions will be fixed on the surface of the object. After a durable grasp between the thumb and at least one finger is established, a secure grasp state is reached where the relative

position of the hand and object is fixed. The process can be divided into three different states (figure 8.1):

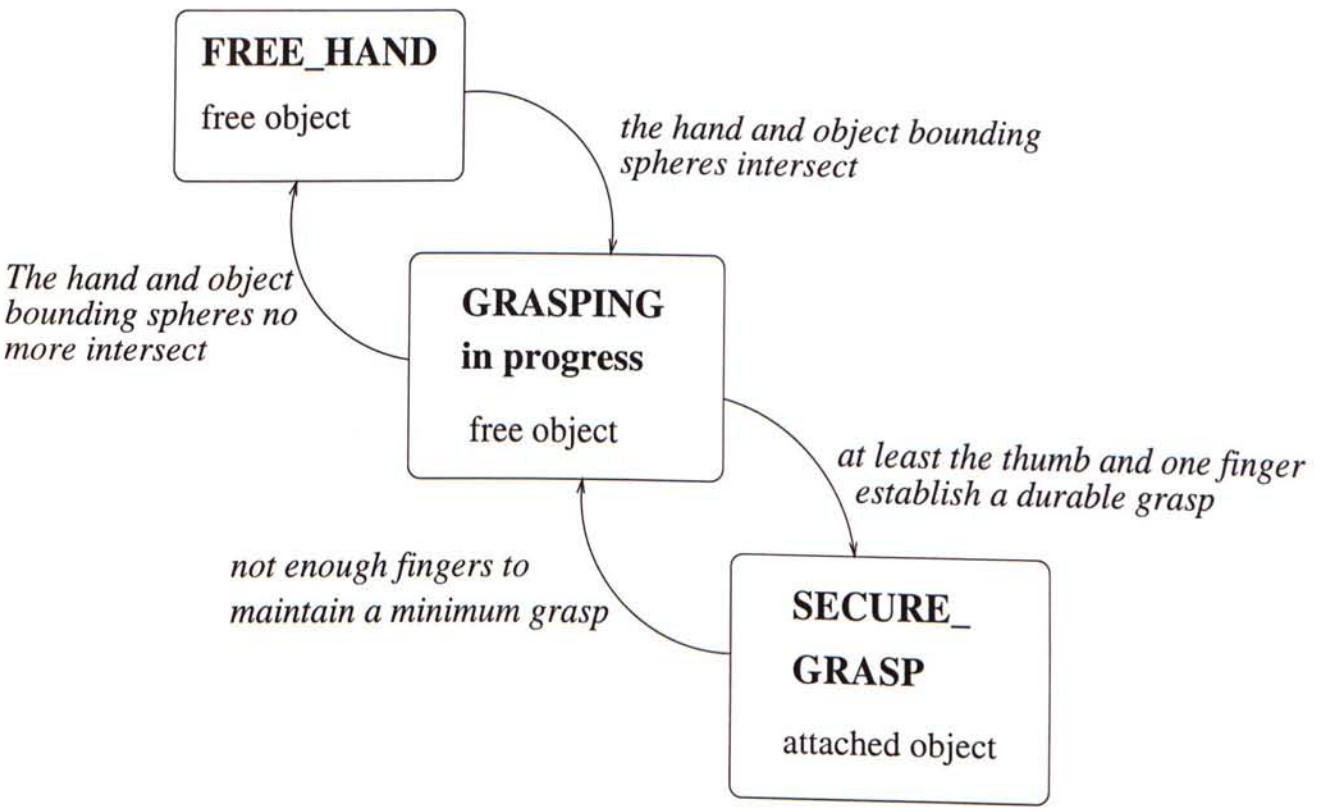

Figure 8.1: The interactive grasping automata

- **FREE_HAND** : there is no contact between the object and the hand. The hand posture is displayed as measured with the digital glove.
- **GRASPING** : the posture of colliding fingers are continuously corrected so that the finger can lie on the surface of the object. The object is free to more relative to the hand. If the simplified grasp condition is established, i.e. at least the thumb and one finger are maintaining a durable contact with the object, the system enters the "**SECURE_GRASP**" state.
- **SECURE_GRASP** : the posture of the colliding fingers are continuously corrected. However, the relative position between the object and the hand is unchanged until the simplified grasp condition is broken.

Whenever a finger collide with the object, interactive grasping procedure adjusts the hand posture by opening it. The reason for this is that it is a common practice for the user to penetrating into the object when they want to grasp it, especially without the help of force feedback. So we assume that the operator will permanently close the grasping fingers slightly more than geometrically necessary. In such way, we can ensure that the unfolding process will result in a durable contact between the fingers and the object. Figure 8.3 shows the steps to perform the unfolding process. The algorithm starts by unfolding the collided finger base joint until it no longer penetrate the object. Then, it unfolds the next and the last joints with the same process. In this case, the final finger posture consistently wraps around the object.

8.2 Physics-based Virtual-hand Grasping

To improve the work of Ronan and Rezzonico's approach. We suggest the *physic-based virtual-hand grasping* approach, which based on my previous work in [13]. The virtual picking in our approach is divided into two phases: *visual-correction* and *active grasping*. *Visual-correction* grasping is based on the work in [3]. It uses kinematics method to calculate the possible colliding contact between the moving fingers of the hand and the object to be grasped. The sensing angles from a CyberGlove device are used for contact checking at each time frame. The checking is conducted first from the joint closest to the palm then outward to the other joints of the finger. If the joint collides with the object, its joint angle will be revised in a way that the joint just lies on the surface of the object, using inverse kinematics. The revised joint angle will be propagated to the other articulated joints of the finger. The reason of using visual-correction instead of full-dynamics is that visual-correction requires less computation than the full-dynamics approach, and can be easily programmed for various kinds of objects.

Active grasping, on the other hand, uses gravity, friction coefficients, and lifting velocity to calculate the physics-based behavior of virtual hand picking. A force

is applied to the object when a joint touches it. The maximum friction between the contact fingers and the object is determined by a constant, which depends on the material of the object. The forces collected at the contact points balance the lifting, while the actual lifting speed produces the forces which are proportional to the friction coefficient of the object. These forces act against the gravity to determine whether the lifting is slippery or stable. A lifting motion is produced when the sum of lifting forces is greater than the gravity.

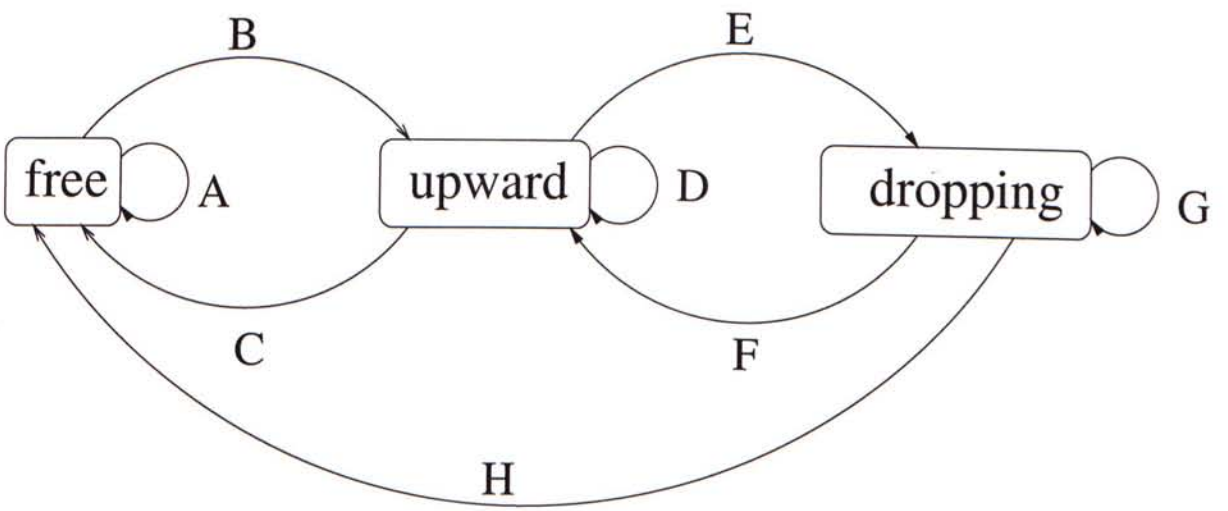

Figure 8.2: Automata of the virtual picking

Figure 8.2 shows the whole set of transitions considered for the virtual picking automata. We can divide the whole process into three states:

- *free* : no collision is detected between the hand and the objects. The hand posture is displayed as measured with the digital glove.

- *upward* : collision is detected. The hand posture is corrected and the forces from the fingers are applied to the object. The frictional forces are strong enough to move the the object and thus the object moves with the hand. ($V_{obj} = V_{hand}$)

- *dropping* : collision is detected. The hand posture is corrected and the forces from the fingers are applied to the object. The frictional forces are

not strong enough to hold the object and thus the object drops from the hand. ($V_{obj} \neq V_{hand}$)

In the *free* state, there is no constraint on the hand posture or the object (transition A). When a collision between one of the fingers and the object is detected, the posture of the collided finger is corrected and forces are applied to the object at the positions where the corrected fingers contact with the object (transition B). The hand is in the *upward* state when the hand first collides with the object. Then as the hand moves upward, the contacts between the fingers and the object are checked to see whether the frictional forces are strong enough to maintain the upward movement of the object. If the upward movement is slow enough, The object will stay with the hand and the upward movement continues (transition D). However, if the upward movement is faster than what the friction forces can support, the object will slip from the hand. Then the system goes into the *dropping* stage (transition E), in which the object keeps dropping in relative to the hand.

Each picking interaction in our system is described by a cycle of states. The hand is initially in the *free* state if none of the fingers collides with the object (transition C and H). When a collision occurs, the hand is changed to the *upward* state, which remains as long as the friction forces can maintain the upward movement of the object (transition D and F). The *upward* state can be again transited to the *dropping* state (transition E and G), when the friction forces are less than the lifting force.

8.3 Visual Correction

This section describes the steps used in the *visual-correction* stage in virtual picking. Ronan and Rezzonico [3] suggests that the hand posture is directly used to establish the grasping interaction on the virtual object. However, the main limitation of this work lies in the lack of force interaction between the fingers of the hand and the object in grasp. Our approach aims at directly using the hand

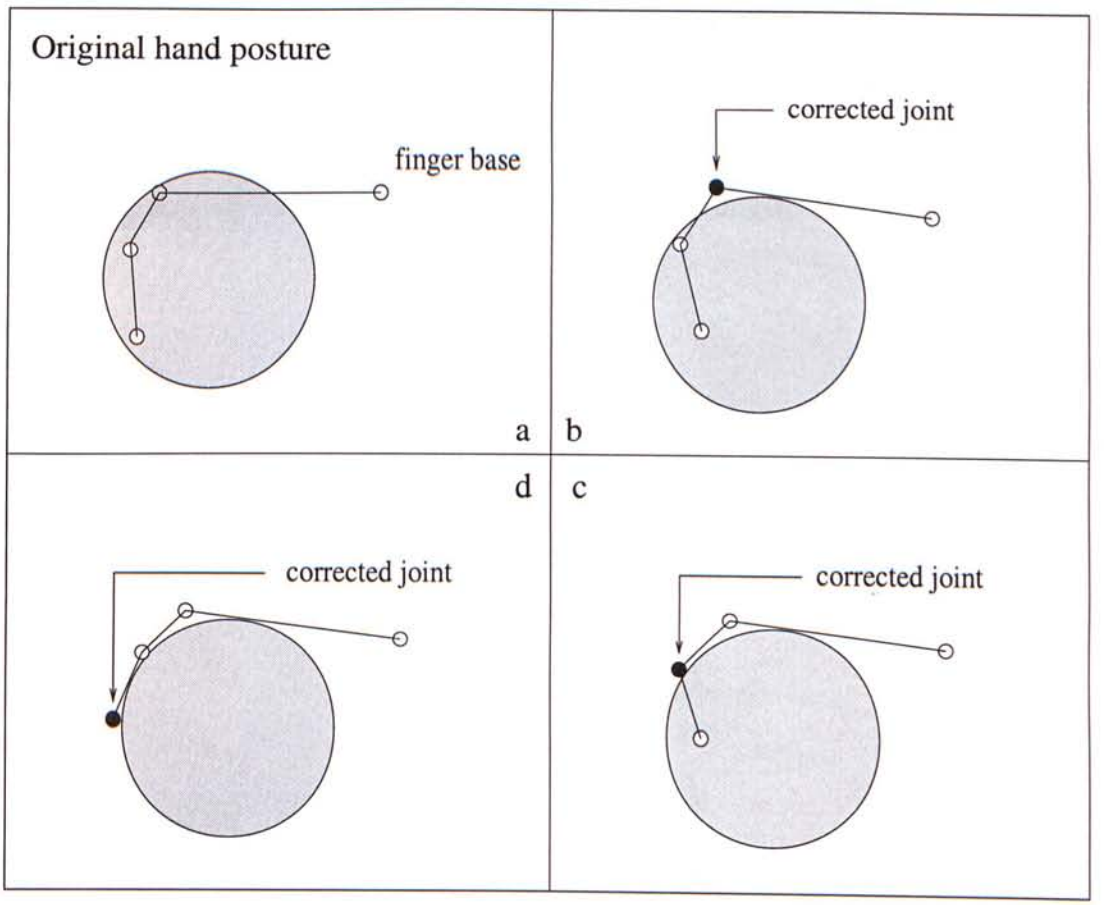

Figure 8.3: Steps of unfolding colliding finger

posture to maintain the grasping interaction as suggested in [3], and in addition the grasping posture is used to compute the balance forces between the contact fingers and the object in grasp.

The main theme of *visual-correction* is to maintain the hand posture by unfolding (correcting) the colliding finger so that the fingers are placed on the surface of the object. Whenever a finger collides with the object, the system adjusts the hand posture by unfolding the finger. The reason for such correction is the common case for the user to penetrate into the object when trying to grasp it in the virtual space, especially without the help of force feedback. Then, the unfolding process is necessary to maintain a durable contact between the fingers of the hand and the object. Figure 8.3 shows the steps of unfolding process. The process starts from unfolding the base joint of a colliding finger until the joint no longer penetrates the object. The revised motion of the joint is propagated to the

lower articulated joint(s) accordingly. Then the same process repeats to unfold the next colliding joint and so on. Again, the unfolding process is applied to the next colliding finger until all the fingers are checked. After the correction, the final hand posture wraps around the surface of the object.

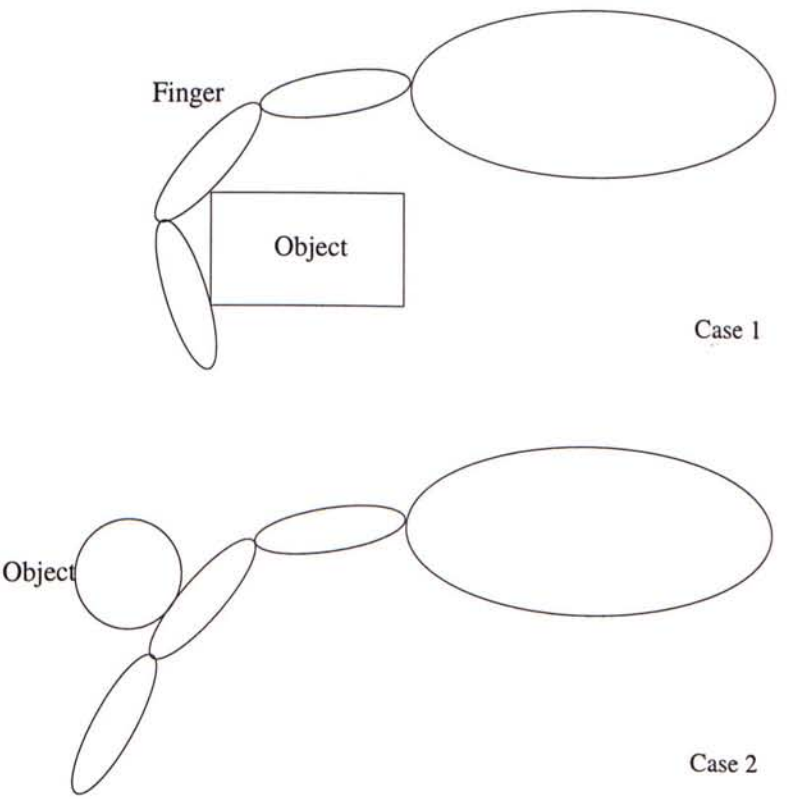

Figure 8.4: Configurations which the joints cannot touch the object

8.3.1 Joint Correction

Different from the approach used in [3], it is not necessary for the joint of the fingers to lie on the surface of the object in our approach. It is because there are some cases which it is not possible for the joints to touch the object. Figure 8.4 shows some of these cases. If we adjust the joint to lie on the surface, some parts of the finger may penetrate into the object, which is not desired.

In our approach, we first calculate the angle which the x-axis of the joint touches the object. Then, we check whether the finger can touch the object.

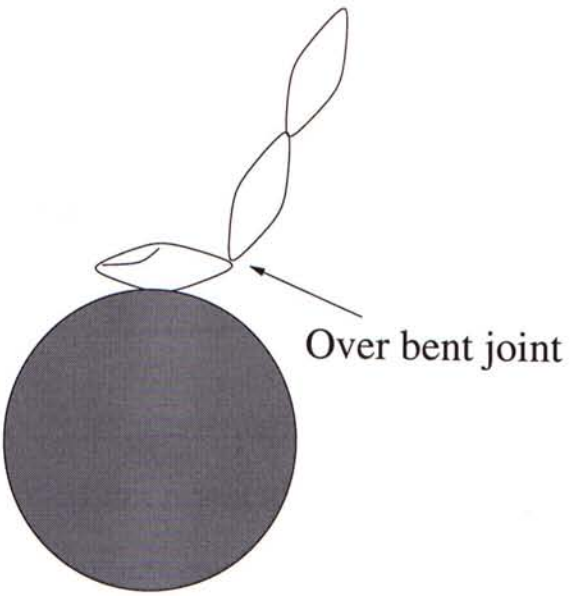

Figure 8.5: Odd configuration after unfolding the colliding joints

If the part of the finger is not long enough and cannot touch the object, we calculate the fold angle again so that the finger tip will lie on the surface. Using this approach, we can ensure that the finger can lie on the virtual object while at the same time no part of the finger will penetrate into it.

8.3.2 Odd Finger Configurations

In some cases, the above approach may result in odd finger configurations. One of the odd configurations is the case when a joint (except the base joint) bends over the normal (flat) position towards the back of the palm. Figure 8.5 shows the above odd configuration. The last joint of the finger is over-bent, which is impossible to human hand. When this happens, the orientation of the joints should be changed for the second correction. Figure 8.6 shows the procedure of the second correction. The second correction is done in a back-up process: Our system first checks the last joint of the finger. If the odd condition is found, the joint angle is flipped to the configuration which keeps the two ending positions the same as before. Then the second last joint of the finger is checked. If the odd condition is found again, the second joint is moved in the same way as keeping

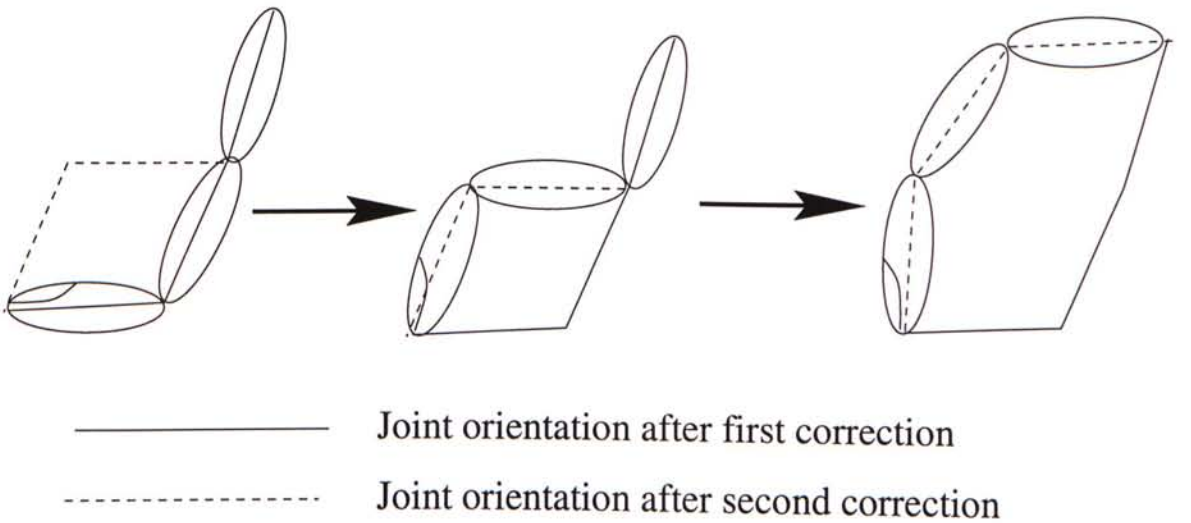

Figure 8.6: Second correction of odd finger configuration

the two end positions fixed. Using this back-up method, we can ensure that the hand configuration after correction is natural and at the same time our correction only changes the contact between the hand and object slightly.

8.4 Active Grasping

After the natural contact between the hand and object is established, the system will proceed to the *active grasping* stage, in which the force interaction between them is calculated. The *active grasping* stage is responsible for the *upward* and *dropping* states in figure 8.2. Our system determines whether the picking process is successful by determining the interacting forces between the contact fingers and the object.

At each contact point, a normal force and a frictional force are calculated and applied in controlling the object's motion.The frictional force is proportional to the normal force, and it follows the Coulomb law. The Coulomb law states that the tangential force of friction during sliding is proportional to the normal force thus defining the coefficient of friction as this constant of proportionality. To simplify the problem, we only used the static coefficient of friction during our calculation. The coefficient of friction is determined by the material of the object, which is

different for every object. The rougher the surface, the higher the coefficient.

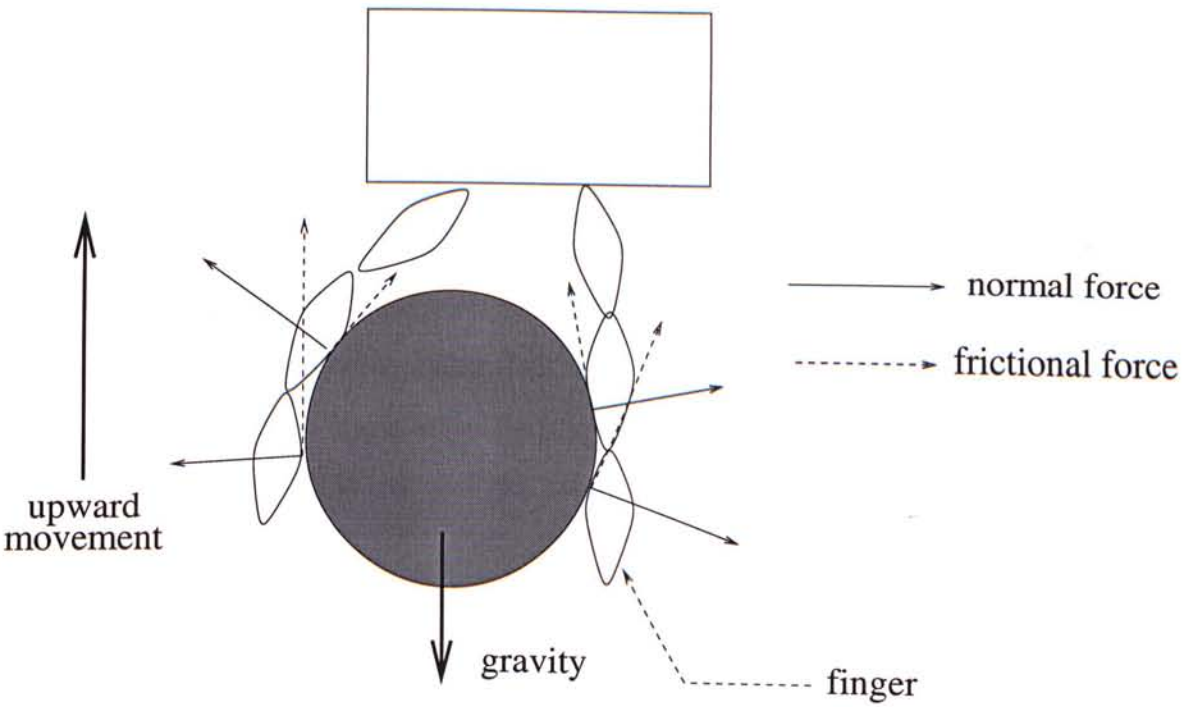

Figure 8.7: Force balance in upward movement

The normal force at each contact point is proportional to the difference between the original and the corrected angle of the colliding joint. This is due to the fact that people usually close their fingers harder when trying to hold the object tighter. So we assume that stronger forces are applied to the object while closing the grasping fingers. We then calculate the frictional force at each contact point by this normal force. Figure 8.7 shows force balance in the upward movement during picking. The equations of the normal force and frictional force are as follow:

$$n_i = k\theta_i \tag{8.1}$$

$$f_i = c_f n_i \tag{8.2}$$

where k is a constant, c_f is the friction coefficient of the object and θ_i is the different between the original and the corrected angle of the colliding joint.

During each time interval δt, we calculate the forces acting on each contact

points. The y-component of these forces is used to calculate the vertical acceleration of the ball. The following equation is used:

$$a_{ball} = (\sum(n_{yi} + f_{yi}) - mg)/m \tag{8.3}$$

We then use the new acceleration to find the velocity of the ball in y-direction. The new velocity of the ball will be compared with the hand's velocity. If the velocity of the ball matched with the hand, the grasping operation success. However, if the hand is moving faster then the ball, the system goes into the *dropping* state and the object starts slipping from the hand. After the comparison is finished, the next cycle starts and the calculation is repeated. We can summarize the process as follow:

During each interval δt
  calculate all contact points on the object
  calculate normal force n_i and respective frictional force f_i on each contact points
  calculate $a_{ball} = (\sum(n_{yi} + f_{yi} - mg)/m$
  calculate the new velocity of the ball in y-direction
repeat for next δt

Algorithm 8.1: Force calculation algorithm

When the object drops out of the hand, its motion is determined by the new velocity and position calculated by the gravity. This also leaves the hand in the *free* state which does not report any collision between the hand and object. The next cycle of picking can be started by another attempt of grasping the object. The hand interaction in grasping is then guided by the two stages: *visual-correction* and *active grasping*, which are described by the three distinct states.

8.5 Collision Detection of Complex Objects

It is well know that collision detection is difficult for complex or irregular objects. Various methods have been suggested to simplified the process [21, 31, 14]. One

of them is hierarchical bounding volumes(spheres and boxes).

The basic idea of using bounding volume is to bound a complex object by simple geometrical primitives. The collision detection is then done by checking the collision of the decomposed primitives, instead of the complex object itself. Therefore, the time required for collision detection is independent of the modeling complexity. Figure 8.8 shows a teapot bounded by a sphere, which should be further decomposed into other primitives that for better approximate the teapot geometry. Figure 8.9 shows one hierarchical bounding volume of the teapot, which can be further decomposed into smaller fitting primitives of the geometry.

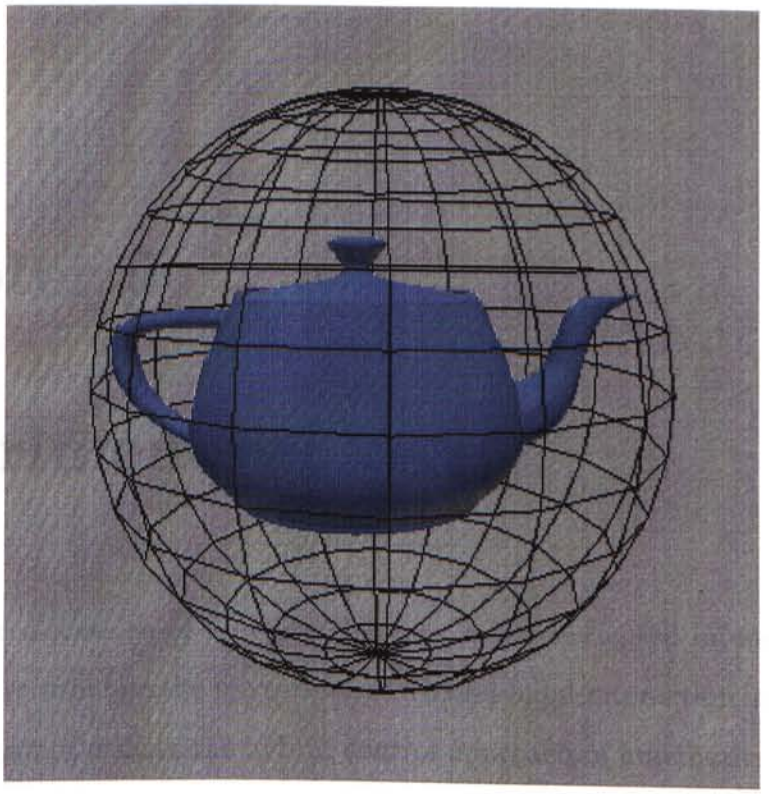

Figure 8.8: Teapot bounded by a sphere

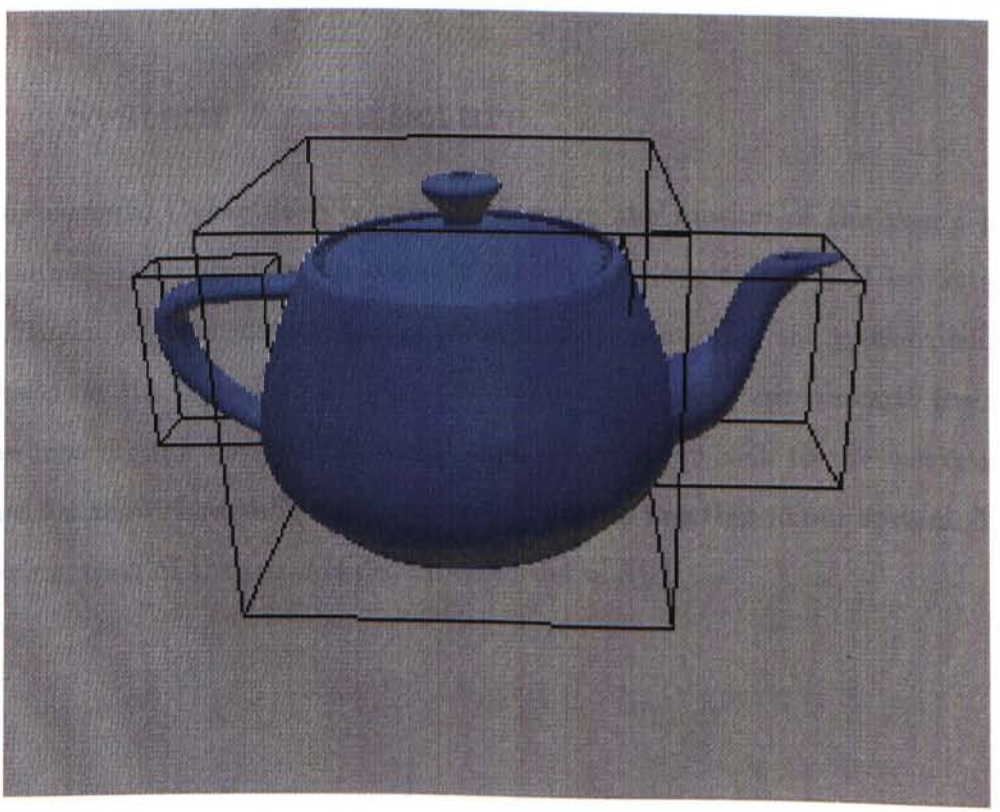

Figure 8.9: Teapot bounded by hierarchical primitives

Chapter 9

Experiments

Based on the discussion of previous chapters, a hand-oriented interactive system is developed to simulate the physics-based virtual-hand interaction in VR applications. The system utilizes the hybrid control approach of kinematics and dynamics, and graph-directed state flow in hand grasping interactions. The following describes the overview of the system architecture and hand grasping interface.

9.1 System Architecture

In our system, CyberGlove is used to capture the motion of the user's hand. Two sub-systems are used to connect the CyberGlove to the SGI workstation. The *Tracking system* detects the position and orientation of the tracker, which is attached on the CyberGlove. The *CyberGlove interface unit* converts the hand shape into digital data and transmit that information back to the workstation. Figure 9.1 shows how different parts are connected together in our system. In the following, each of the parts is described in detail.

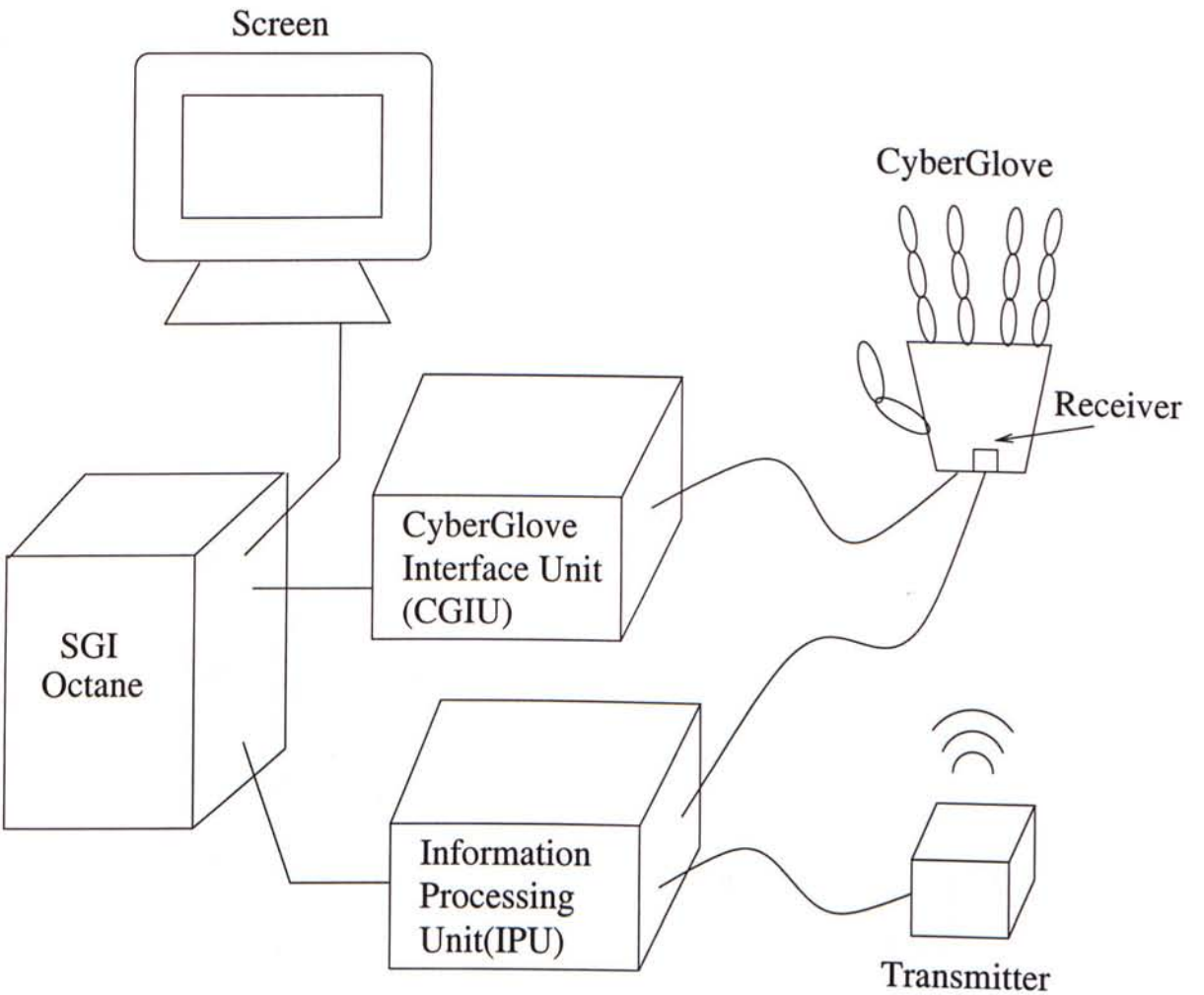

Figure 9.1: Hand-oriented interactive system architecture

9.1.1 Tracking System

The Polhemus^{TM} FASTRAK tracking system [24] is designed to determine the three-dimensional position and orientation of the CyberGlove. The system generates low frequency electromagnetic field by a transmitter. The field is detected by a receiver and the signal is used to compute the receiver's position and orientation relative to the transmitter. The workstation uses this information to determine the movement of virtual hand.

The FASTRAK system consists of an information processing unit (IPU), a transmitter and a receiver. The **information processing unit** controls all the input and output of the system. All signals generated by the transmitter are received by the receiver and processed here. It also provides switches for selecting

working mode of the system. The **transmitter** generates electromagnetic signals received by the **receiver**. The resolution of the system is 0.0005 cm and 0.025°. The receiver can provide an accurate spatial if it locates with 76cm range of the transmitter. Further distance is possible (up to 305cm) but the system will suffer reduced accuracy. Within the 76cm limit, the system has a 0.08cm RMS (root mean square) error for X,Y, and Z position, 0.24cm RMS error for orientation. Also, there will be a latency of 4ms before the IPU receives the current receiver location.

9.1.2 Glove System

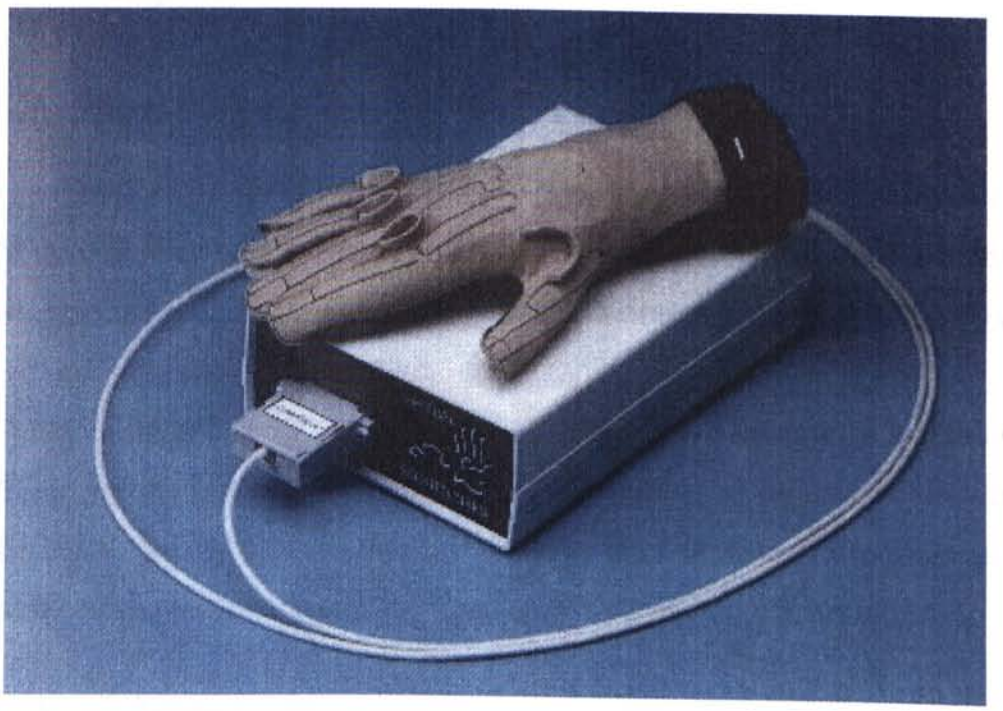

Figure 9.2: A 22-sensor CyberGlove and the CyberGlove Interface Unit (CFIU) (Picture adapted from http://www.virtex.com/prod_cyberglove.html, the official homepage of Virtual Technologies.)

The glove system [40] consists of two parts, the CyberGlove and CyberGlove Interface Unit(CFIU). The **CyberGlove** contains 18 or 22 sensors which are placed on different positions of the glove. The shape of the hand is detected by the sensors and sent to the **CGIU**. The CGIU amplifies and digitizes the signals

received and relay the data to host computer via an RS-232 interface. The host computer then calculates the shape of the hand according to the data and re-generate the hand shape by the virtual hand.

The 22 sensors in the CyberGlove are located over or near the joints of the hand and wrist to capture the motions of the physical hand and fingers. They are made of elastic material which will change its resistance when the length changes. Therefore, when the finger is bent, it will bend the sensor and extend its length, which affects its output voltage. The system can then determine the angle of the joint by the output voltage. The sensor has resolution of 0.5 degree and 1 degree standard deviation. The glove can send up to 112 records each second under normal working condition.

9.1.3 Host Computer

The workstation we used is SGI Octane workstation with a R10000 processor and 128M ram. It serves as the link between the tracking system and the glove system. It uses the data received from the two systems to simulate the interaction between the virtual hand and virtual object.

When all objects are fully rendered, the system can operate in a rate of 18-24 frame per second, which may vary by the number of objects in the virtual environment. When the objects are rendered in wireframe, the system can archive up to 60-70 frames per second.

9.2 Experimental Results

In this section, two sets of experimental result will be shown. The first set is the general application of virtual hand grasping. Different objects and grasping conditions will be used to test the stability of the system. The second set of the experiment is to analyze the relationship between frictional coefficient of the object and the mass of the object. We want to test how large the frictional coefficient required to maintain a stable grasping operation. Different mass and coefficient

combination will be used during the test.

9.2.1 General application

The following is the snapshots of the virtual hand grasping the objects in the virtual environment. Figure 9.3 to Figure 9.5 show the sequence of grasping a sphere. Figure 9.6 shows the view while grasping the sphere from another direction.

Figure 9.7 shows the case when the hand grasping a cube. If the upward force is larger than the gravity, the object can be grasped by fewer fingers. Figure 9.8 shows the examples of grasping the cube with two fingers.

Figure 9.9 shows other examples of virtual hand model grasping the objects in the virtual environment constructed by primitives and polygons.

9.2.2 Relationship between frictional coefficient and mass of the object

In our force equation, the frictional force is proportional to the normal force and the frictional coefficient of the object. It is clear that an object with a rough surface can maintain a stable grasping configuration easier than one with smooth surface. On the other hand, the heavier the object, the larger the gravity force acting on it. In this experiment, we want to analyse the relationship between frictional coefficient and the mass of the object.

Table 9.1 shows the frictional coefficient of different material we used in the test. Note that the coeffient listed is only an average value of the type of object. The variance of the value can be very large. For example, the surface of a wood table can have a coefficient of 0.8, while the surface of a tree can have a coefficient of 2 to 3.

In this experiment, sphere of 6 different friction coefficient will be tested. We try to pick the sphere with different number of fingers and try to find the maximum mass which we can pick up. Starting from the initial mass, if we can successfully

pick up the sphere, we will increase the mass in a stepwise manner until we can no longer pick it up. The maximum mass will be record and the test will continue with another friction coefficient with the initial mass.

Table 9.2 shows the experiment result. The result shows that the maximum mass and the number of fingers uses follow a directly proportional relationship. It is obvious because the force acting on the object depending on the number of contact points. The more fingers we use, the larger the frictional force. Therefore, to prevent the object to slip from the hand, more fingers should be used.

Another observation is the relationship between the frictional coefficient and the mass. We found out that the maximum mass is about $C(1+c_f)$ times of the frictional coefficient, where C is a constant and c_f is the frictional coefficient.

Object	Frictional Coefficient
Metallic surface	0.4
Glass	0.5
Plastic	0.9
Wood	1.2
Paper	1.3
Cloth	1.6
Rock	2.2

Table 9.1: Frictional Coefficient of different materials

	Maximum mass(Kg) with number of fingers			
Frictional Coefficient	2	3	4	5
0.5	7.5	11.2	14.9	18.4
0.8	9.1	13.3	18.0	22.2
1.1	10.4	15.9	20.6	25.9
1.4	12.2	19.0	24.3	30.8
1.7	13.5	20.1	27.0	33.2
2.0	14.8	22.2	29.1	36.6

Table 9.2: Maximum mass of object which can be grasped

Figure 9.3: The hand trying to grasp the sphere

Figure 9.4: The hand grasping the sphere

Figure 9.5: The sphere is picked up by the hand

Figure 9.6: Grasping the sphere from another direction

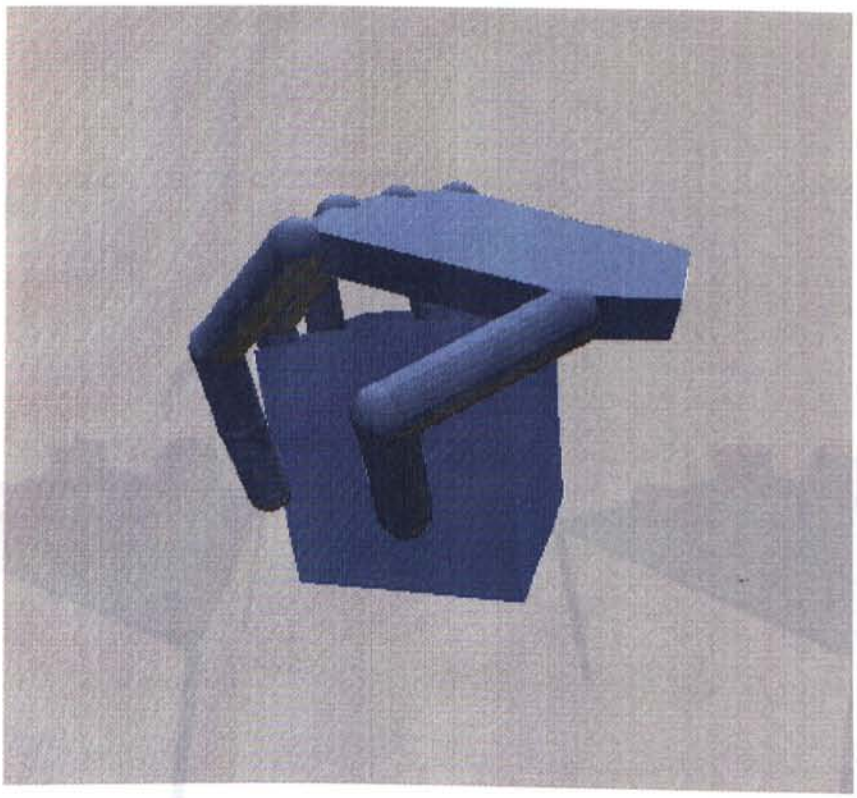

Figure 9.7: The hand grasping the cube

Figure 9.8: The hand grasping the cube by two fingers

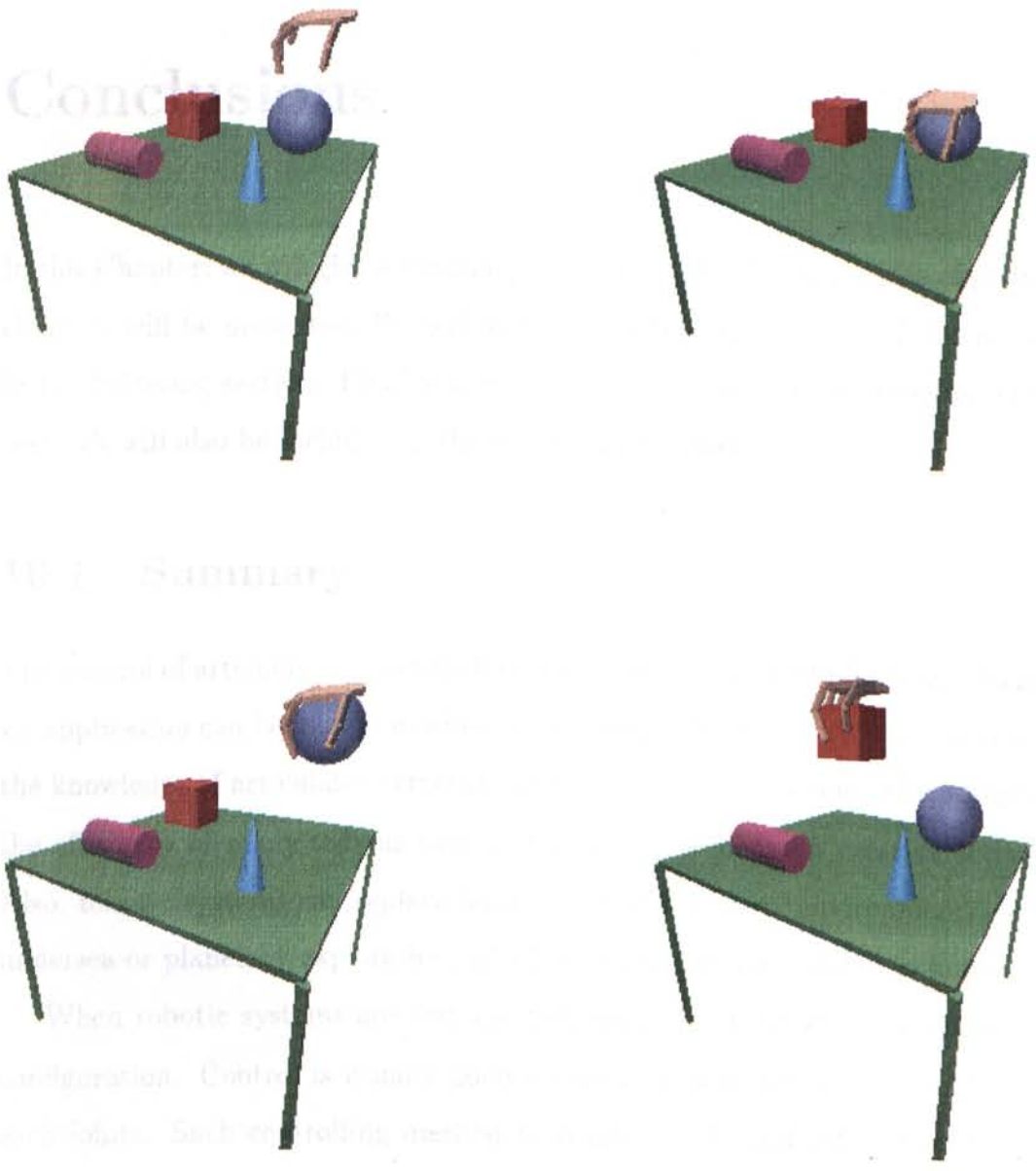

Figure 9.9: Hand model manipulating objects in virtual environment

Chapter 10

Conclusions

In this Chapter, we will give a summary of our research. The main idea of previous chapters will be presented. We will also list out the contribution of this research in the following section. Finally, a discussion of possible future direction of this research will also be included at the end of this chapter.

10.1 Summary

The control of articulated structure has been in extensive study for many decades. Its application can be found anywhere in our daily life. Robotic system combining the knowledge of articulated structure and computer automation, which improves the efficiency of many tedious task and allow us to focus on creative activities. Also, robotic systems can replace human to work in many environment, such as undersea or planetary exploration, which is hazard or inaccessible to human.

When robotic systems are first created, they only have simple structure and configuration. Control is usually done by the means of directly manipulation at each joints. Such controlling method is usually costly and inflexible. With the increase of availability and computation power of computer system, simulation is done before the real system is built. This can reduce the operation cost and building time. For decades, simulation is done in 2D environment, with switches or keyboard as the input device. However, working efficiency is usually affected

by the limitation of control space. Special training is required before the operator can control the system.

In this research, we have done an extensive study on the requirement of robotic simulation. The simulation should be built in an interactive control environment with 3D interface and small latency. Digital glove is selected as the input because it can reproduce the operation performed by the users. Also, the user does not require special training to operate the system, which can significantly reduce the operation cost.

Two motion control techniques, kinematics and dynamics, are discussed. They both have their own advantages and disadvantage, which make no clear winner when controlling and articulated structure. To adopt the efficiency of kinematics without losing the accuracy of dynamics, we suggest an approach to combine the two methods. When the articulated structure is moving in a free space, which it does not collide with any object, kinematics is used to calculate its motion. On the other hand, when collision appears, dynamics approach is used to calculate the interaction. This can maintain a fast response and small latency system without losing its accuracy.

Other than fast response and small latency, accurate feedback system is also important. In this research, common approach of physical feedback has been studied. General description of previous are also included. Unfortunately, it is found that most physical feedback has its limitation and are usually expensive. The special hardwares required are not easily available. As a result, we aim at improving the visual feedback to compensate the lack of physical feedback systems.

One of the most basic operations in virtual reality is object manipulation. Different methods have been purposed to provide a natural and convenience method for object grasping. However, our study show that most of these method fail to fulfill the requirement. Instead, an approach called interactive grasping, which based on the work of Ronan and Rezzonico's study, is suggested. The visual configuration of the digital glove is modified, so that the colliding fingers can easily lie on the surface of the virtual object. The user does not need to take care of

the correct posture or hand shape to grasp an object. Also, dynamic is used to determine whether the grasping is success or not. If the virtual hand fail to supply enough upward force to the object, the object will slip from user's hand and fall.

To place the above discuss in context, an interactive manipulator control system is built. It simulates the operations performed by user hand and reproduces it in virtual environment. The user can control the objects by their hand - just like what we done in our daily life. In the system, different virtual objects are included. The user can manipulate any objects in the environment. This system can be used as the base of other simulation system, such as machine design, long distance robot control, virtual environment exploration, etc.

10.2 Contributions

This research has the following contributions:

- Outline the needs of robot simulation in VR.
- Unify the control space of virtual robot hand and arm structure.
- Propose a hybrid control approach of kinematics and dynamics in virtual robot simulation (global and movement/local hand manipulation)
- Develop a hand-oriented interactive system for VR applications.
- Combine visual correction and active lifting in hand interaction.
- Implement physics-based picking of virtual hand models.

10.3 Future Work

To improve the control interface of the system, a 3D display environment is required. It allows the user to view the environment from different viewpoint. This can be done by changing the display device from 2D screen to Head Mount Display or Crystal Eyes.

In the virtual object manipulating system, the object cannot be re-oriented by the hand. A further enhancement of the system is to allow the orientation of the object to be changed when it is grasped by the hand. The object should move with the hand if the orientation of the virtual hand is changed.

Finally, constraint should be added to the system. In the program, the position of virtual objects are not checked. Different objects can cut into each other, which is not realistic. Constraint mechanism should be added to prevent this to happen.

Appendix A

Description files

In this section, the format of the two description files used in this system will be shown.

A.1 Scene Description

To allow simple modification of the virtual environment, an scene description language is created. It describes the objects in the virtual environment, magnitude of the gravity, the ground level, etc. The scene description is stored in a simple ASCII file. Table A.1 shows one example of a scene description file. The resultant scene is shown in figure A.1.

We can divide the scene description file into two parts. The first part specifies the environment parameters. **FORCE_CONSTANT** is used to specify the force constant between the hand and virtual object. **GRAVITY** specifies the gravitational acceleration acting on the object. We can simply turn off the gravity in the virtual environment by giving 0 to this parameter. **GROUND** tells the system where the ground level lays to avoid the virtual object dropping beyond the ground constraint.

The second part of the file specifies the objects in the virtual environment. Three types of objects, spheres, cubes and polygons, are supported in the current system. The object is specified by a name, its color and property parameters.

```
FORCE_CONSTANT 0.5
GRAVITY 9.8
GROUND -3.0
SPHERE                  # a sphere
1.0 0.0 0.0             # color of the sphere
4.8 2.2 8.0             # center position
4.0                     # radius
POLYGON 3               # a polygon with 4 sides
0.0 1.0 0.0             # color of the polygon
-3.0 0.0 3.0            # position of the vertices
-3.0 8.0 6.0            # note : vertices must be given
7.0 14.0 3.0            # in anti-clockwise manner
CUBE                    # a cube
0.0 0.0 1.0             # color of the cube
-6.0 -4.0 -4            # position of starting vertices
8.0 8.0 8.0             # size of the cube (x,y,z)
```

Table A.1: Scene description language

SPHERE specifies the position (x,y,z) and radius (r). **POLYGON** specifies the number of vertices, the color, the position (x,y,z) of the vertices in counter-clockwise order. Note that the number of vertices must be the same as the one specified early. **CUBE** specifies the color, the starting point, the size extended in (x,y,z).

The object data of the scene is stored in a structure with two layers. The first layer contains the general information shared by all the objects (type, color, velocity). The next layer contains the parameters of the object. The **sphere_node** includes *radius and center position(x,y,z)* of the sphere. The **polygon_node** includes *size*, which specify the number of vertices, *normal* calculated from the vertices, and *vertices*. The **cube_node** contains the *position(x,y,z)* of the starting point, *color*, and *size(x,y,z)* of the cube. Figure A.2 outlines the data structure of scene description.

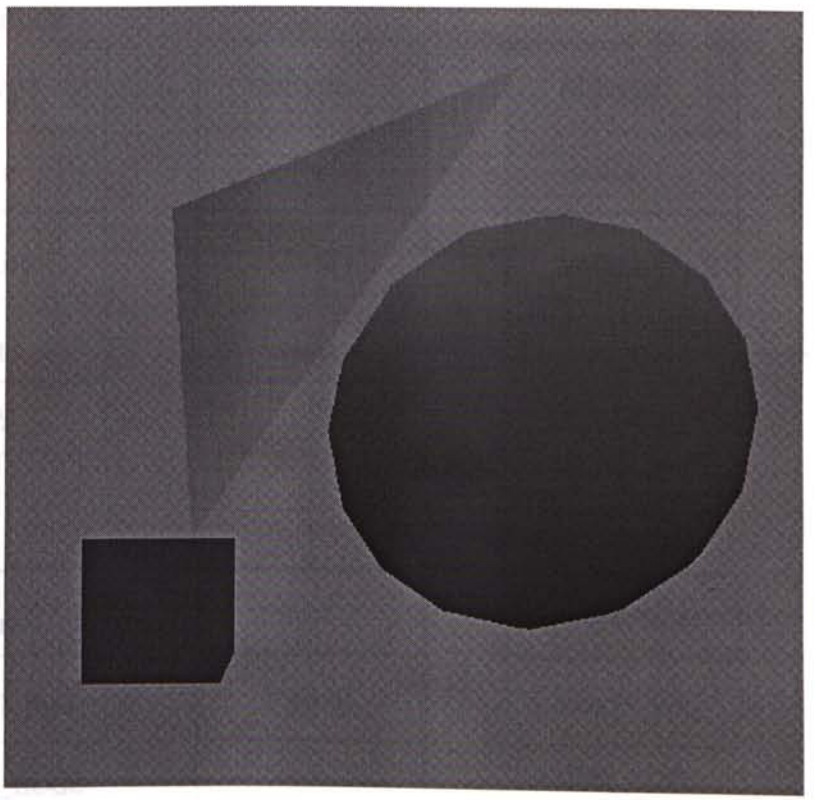

Figure A.1: The scene specified by the scene description file

A.2 Hand Description

Similar to scene description, **hand description** specifies the initial state of the hand structure. Table A.2 shows the hand description and Figure A.3 shows the virtual hand.

The hand file specifies the position(x,y) of each finger joint relative to the center of the wrist. The file specifies one finger at a time, from thumb to pinkie finger. It first specifies the position of the base joint, then 1st joint, 2nd joint and the finger tip. After all the fingers, the file specifies the radius of the finger along its cross section. Finally, the last parameter specifies the switch to turn on/off the CyberGlove. This parameter is used for debugging process. Every time the program starts, the CyberGlove and Tracker system need to be re-initiated. However, the initiation process requires a long time and it will be inconvenient during the system development process. Therefore, this parameter is used to turn the CyberGlove and Tracker system. If off, the system initializes all joint angles

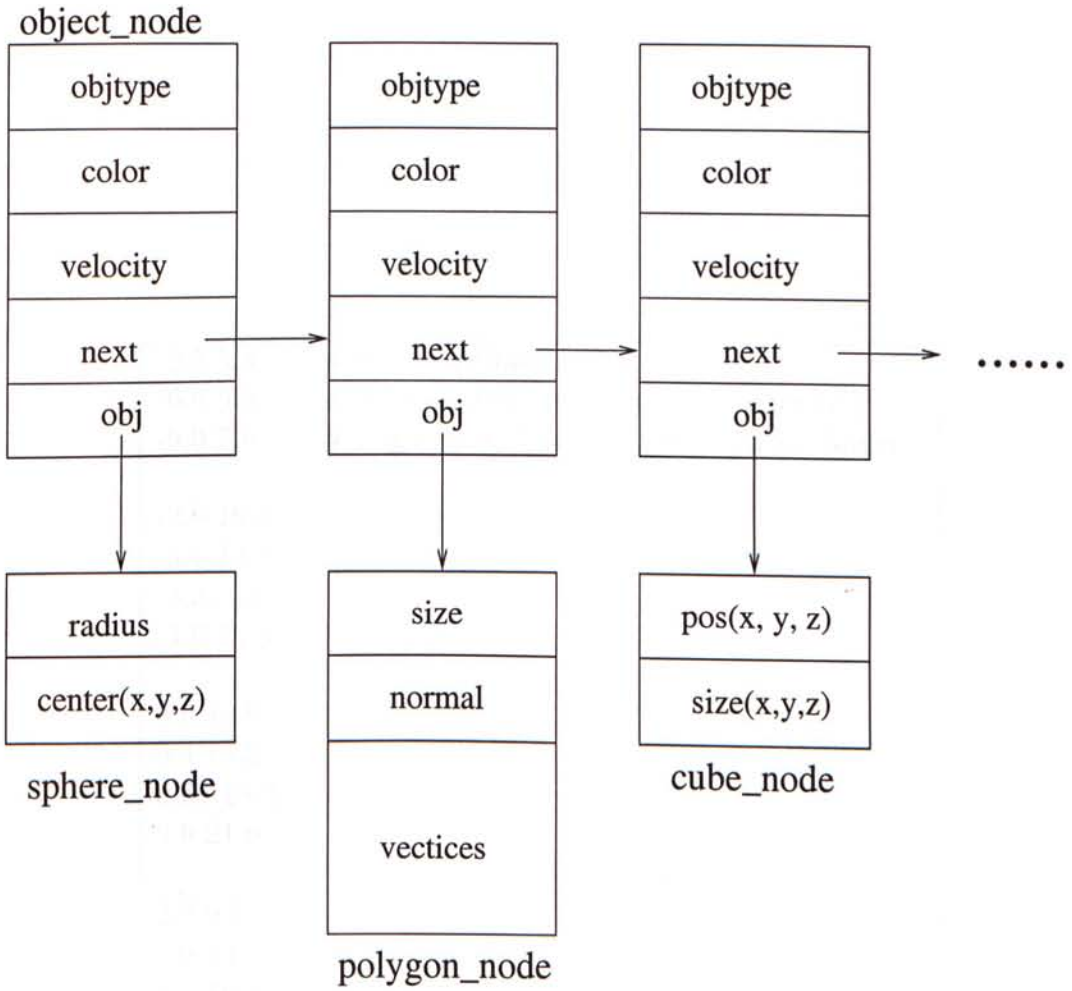

Figure A.2: Data structure of object specification

and glove position to zero, which can speed up the system development process.

```
-2.5 3.4      # x,y coordinate of the joint
-6.6 5.8      # starting from base joint to finger tip
-9.0 7.6      # finger start from thumb to pinkie finger

-2.6 10.4
-3.1 14.8
-3.35 17.4
-3.6 19.8

0.0 10.6
0.4 15.2
0.65 18.2
0.9 21.0

2.2 9.8
3.0 13.8
3.6 16.8
4.2 19.4

4.2 8.6
5.6 11.4
6.55 13.4
7.6 15.5

0.7           # radius of the finger
ACTIVE        # specify whether the glove device is active
```

Table A.2: Hand description file

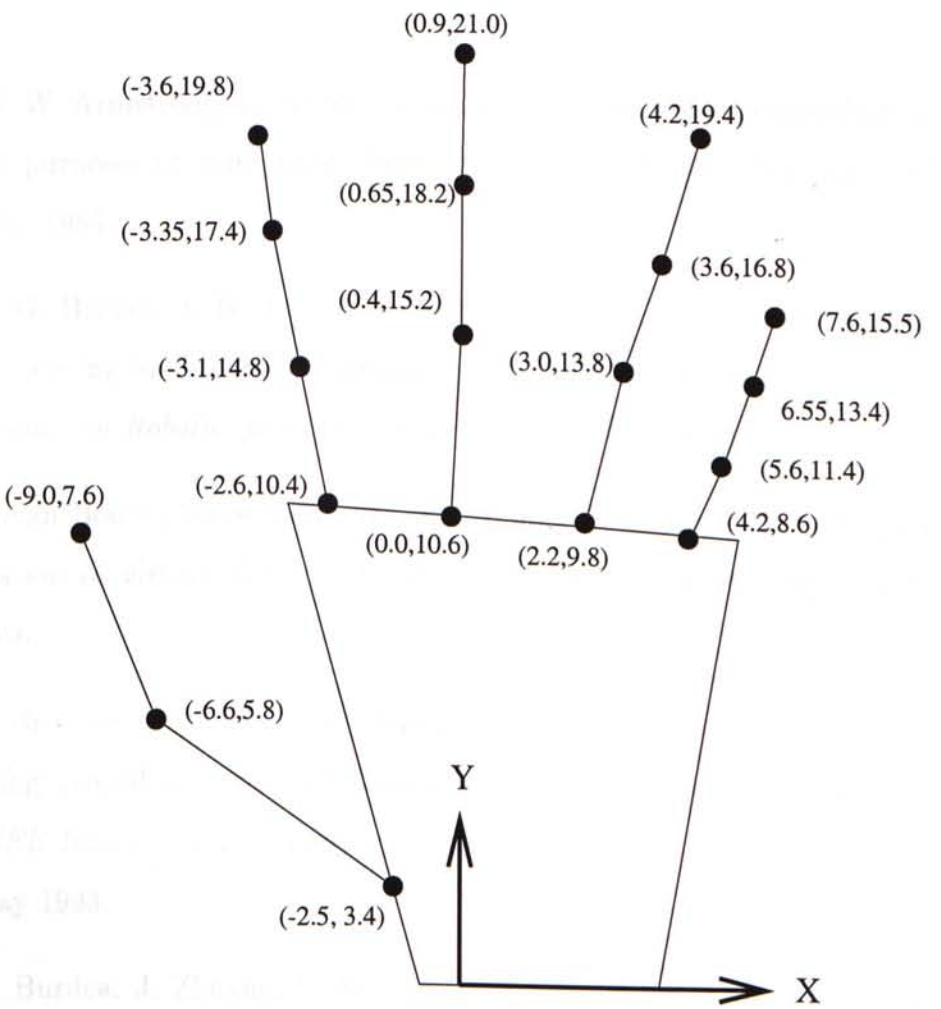

Figure A.3: Hand coordinate according to hand specification file

Bibliography

[1] W. W. Armstrong and M. W. Green. The dynamics of articulated rigid bodies for purposes of animation. *Proc. of Graphics Interface '85*, pages 407–415, May 1985.

[2] P. G. Backes, J. Beahan, and B. Bon. Interactive command building and sequencing for supervised autonomy. *Proc. of the IEEE International Conference on Robotics and Automation*, 2:795–801, 1993.

[3] Ronan Boulic, Serge Rezzonico, and Daniel Thalmann. Multi-finger manipulation of virtual objects. In *Proceedings of VRST'96*, pages 67–74, July 1996.

[4] B. Brunner, J. Heindl, G. Hirzinger, and K. Landzettel. Telerobotics systems using virtual environment display with visual and force display functions. *IEEE International Conference on Robotics and Automation*, pages 98–111, May 1993.

[5] G. Burdea, J. Zhuang, E. Roskos, D. Silver, and N. Langrana. A portable dextrous master with force feedback. *Presence-Teleoperators and Virtual Environments*, 1(1):18–27, March 1992.

[6] Grigore Burdea and Philippe Coiffet. *Virtual Reality Technology.* John Wiley & Sons, Inc., New York, 1994.

[7] Martin Buss and Hideki Hashimoto. Dextrous robot hand experiments. In *International Conference on Robotics and Automation*, pages 1680–1686. IEEE, 1995.

[8] D. Caldwell and C. Gosney. Enhanced tactile feedback (tele-taction) using a multi-functional sensory system. In *Proceedings of the 1993 International Conference on robotics and Automation*, pages 955–960, Atlanta, GA., May 1993.

[9] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *J. Applied Mechanics*, pages 215–221, June 1955.

[10] R. Featherstone. Position and velocity transformation between robot end-effector coordinate and joint angle. *The Internation Journal of Robotics Research*, 2(2):35–45, 1983.

[11] A. A. Goldenberg and D. L. Lawrence. A generalized solution to the inverse kinematics of robotic manipulator. *Journal of Dynamic System, Measurement, and Control*, RA-1(1):14–20, 1985.

[12] T.N.E. Greville. The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations. *SIAM Review*, 1(1):38–43, January 1959.

[13] Kwok Lai Ho. Physics-based virtual-hand picking in robotic manipulation. *IEEE International Conference on Systems, Man and Cybernetics*, October 1998.

[14] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, July 1996.

[15] H. Iwata. Pen-based haptic virtual environment. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 287–292, Seattle, WA, September 1993.

[16] C.A. Klein and C.H. Huang. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):245–250, March/April 1983.

[17] D. Bread M. Ouh-young and F. Brooks Jr. Force display performs better than visual display in a 6-d docking task. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, pages 1462–1466. IEEE Press, 1989.

[18] D. Manocha and J. F. Canny. Efficient inverse kinematics for general 6r manipulators. *IEEE Transaction on Robotics and Automation*, 10(5):648–657, 1994.

[19] D.P. Mapes and J.M. Moshell. A two-handed interface for object manipulation in virtual environments. *Presence*, 4(4):403–416, Fall 1995.

[20] R. Max and D. Thalmann. A hand control and automatic grasping system for synthetic actors. In *Proceedings of Eurographic'94*, pages 167–178, 1994.

[21] M. Moore and J. Wilhelms. Collision dection and response for computer animation. *Computer Graphics (Proceedings of Siggraph)*, 22(4):289–298, August 1988.

[22] Hiroyuki Ogata and Tomoichi Takahashi. Robotic assembly operation teaching in a virtual environment. *IEEE Transactions on Robotics and Automation*, 10(3), June 1994.

[23] N. Patrick. Design, construction, and testing of a fingertip tactile display for interaction with virtual and remote environments. Master's thesis, Department of Mechanical Engineering, MIT, August 1990.

[24] Polhemus Incorporated Colchester, Vermount. *Polhemus 3Space Users Manual*, 1994.

[25] S. Rezzonico, R. Boulic, Z. Huang, N. Magnenat-Thalmann, and D. Thalmann. Consistent grasping in virtual environments based on the interactive

grasping automata. In *Virtual Environment.* M. Gobel Edt, Springer Verlag Wien, 1995.

[26] Alberto Rovetta. A graphic interface in telerobotics : Utilization for surgery and training. *IEEE International Conference on Robotics and Automation*, pages 2389–2393, 1995.

[27] B. Schmult and R. Jebens. A high performance force-feedback joystick. *Virtual Reality Systems '93 Conference*, pages 123–129, March 1993.

[28] L. Sciavicco and B. Siciliano. A dynamic solution to the inverse kinematic problem of redundant manipulators. In *IEEE International Conference on Robotics and Automation*, pages 1081–1086, March 1987.

[29] Luis Serra, Tim Poston, Wieslaw Nowinski, Chua Beng Choon, Ng Hern, and Prem Pillay. The brain bench planner and trainer for minimal access surgrey. In *Symposium on Virtual Reality Software and Technology*, pages 191–192. ACM, July 1996.

[30] K. Shimoga. Finger force and touch feedback issues in dextrous telemanipulation. In *Proceedings of NASA-CIRSSE International Conference on Intelligent Robotic Systems for Space Exploration*, Troy, NY, September 1992.

[31] Andrew Smith, Yoshifumi Kitamura, and Haruo Rakemura nad Fumio Kishino. A simple and efficient method for accurate collision detection among deformable polyhedral objects in arbitrary motion. In *IEEE Virtual Reality Annual International Symposium*, pages 136–145, 1995.

[32] W. E. Snyder. *Industrial Robots: Computer Interfacing and Control.* Prentice-Hall, 1985.

[33] T.H. Speeter. Transforming human hand motion for telemanipulation. In *Presence*, volume 1 of *1*, pages 63–79. The Massachusetts Institute of Technology, 1992.

[34] S. Steketee and N. Badler. Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control. *Computer Graphics*, 19(3):255–262, July 1985.

[35] R. Stone. Advanced human-system interfaces for telerobotics using virtual reality & telepresence technologies. In *Proceedings of the Fifth International Conference on Advanced Robotics ('91 ICAR)*, pages 168–173, Pisa, Italy, 1991.

[36] D. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, 1994.

[37] H. Sun. A behavioral test-bed using dataglove input device. *Virtual Reality Journal of Research, Development and Applications*, 1(2):109–116, December 1995.

[38] Hanqiu Sun and Victor Kwok. 3d interactive control of robotic manipulations. *Fifth International Conference on Computer-Aided Design and Computer Graphics*, 1:18–23, December 2-5 1997.

[39] T. Takahashi and T. Sakai. Teaching robot's movements in virtual reality. *Proc. of IEEE Int. Workshop on Intelligent Robots and Systems (IROS)*, pages 1583–1588, 1991.

[40] Virtual Technologies, Palo Alto. CyberGlove^{TM} *User's Manual*, 1994.

[41] Collin Wang. Virtual-reality-based point and direct robotic inspection in manufacturing. *IEEE Transaction on Robotics and Automation*, 12(4), August 1996.

[42] Richard Watson. A survey of gesture recogniture techniques. Technical Report TCD-CS-93-11, Department of Computer Science, Trinity College, Dublin2, July 1993.

[43] J. Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *SIGGRAPH '87 Course Notes on Computer Animation*, 1987.

[44] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[45] W.A. Wolovich and H. Elliot. A computational technique for inverse kinematics. In *23rd Conference on Decision and Control*, pages 1359–1362, December 1984.