A REAL–TIME VIRTUAL–HAND RECOGNITION SYSTEM

By

Tsang Kwok Hang Elton

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

December 1998

摘要

「虛擬現實」 (virtual reality) 是近年來備受關注的研究題目。「虛擬現實」有兩個主要特徵—實時互動性 (real--time interactivity) 及代入感 (sense of immersion)。由於傳統的人機交互技術 (human--computer interaction techniques) 並不能滿足這兩個特徵，研究人員正在尋求一些新的辦法去處理人、機之間的溝通問題，而手勢識別正是其中一個主要的研究方向。

這篇專題論文的研究題材是「虛擬手實時識別系統」，即如何在虛擬環境中，利用一些手勢識別算法來辨認靜態和動態的手勢。本論文旨在發展一個簡單、 準確、快速及靈活的虛擬手識別系統，而有關的背境資料, 包括手部的結構、手的模型、手追蹤技術、現有的手勢識別方法及系統均會在本論文中詳細介紹。

手勢識別並不是一個容易解決的問題，現有的識別方法都存有很多問題，例如不夠準確、效率不足等等。本論文提出了一個新的識別算法，該算法利用「模糊理論」 (fuzzy theory) 去解決一些以往遇到的限制，並且進行了一系列的實驗，以証明該算法的可行性。此外，本論文還發展了兩個應用程式，以驗証該手勢識別系統的實用價值，分別是一個手勢資料庫的編輯器和一個三維造型.。而組實驗結果均証實該手勢識別算法已達到預期中的效果。

Abstract

Virtual reality (VR) is attracting more and more attention in both the academic and commercial communities. As traditional human–computer interaction technologies failed to fulfill both key features of virtual reality, which are interactive and immersive, researchers are looking for more sophisticated methods for the communication between human and computers. Inspired by the capabilities of hand for controlling computer–mediated tasks, hand–input has become one of the major research directions.

This dissertation is a research on *virtual–hand recognition*, which is defined as the utilization of recognition algorithms to identify hand postures and gestures in virtual environments. It aims at developing a virtual–hand recognition algorithm which is accurate, flexible, simple, and most importantly, efficient.

In this dissertation, a wide range of related background knowledges are studied, including the structure of the human hand and the common hand models, hand–tracking technologies, current hand recognition algorithms and a number of example hand–input systems.

Posture and gesture recognition is a not an easy problem. Existing recognition algorithms are either inaccurate or inefficient to incorporate with virtual reality applications. A novel posture recognition algorithm using fuzzy theories is proposed to overcome some series limitations of previous approaches. A serious of experiments are carried out to examine the behaviour of the proposed algorithm. Moreover, two applications, a posture database editor and a 3D modeler, are also developed to investigate the practical performance of the proposed algorithm. It is found that the proposed system fulfills the mentioned objectives.

Acknowledgements

First, I would like to thank my supervisor, Prof. Hanqiu Sun, for her supervision over these two years. She gave me a lot of valuable advice and suggestions to my work. I would also like to thank for the valuable comments about this thesis from other committee members Prof. Mark Green, Prof. Heng and Prof. King.

Many thanks to my friends Michael Fung, Victor Kwok, Kevin Wong, Tommy Siu, Philip Fu and Kai Lam, who frequently share their experience with me. With these experience, many unnecessary errors are avoid and my time can be spent more efficiently. Thanks should also give to the Assistant Computer Officers in our department, Raymond Chan and Fiona Lam for their technical support during the development of the virtual world modeler.

Finally, I would especially like to thank my family for their love and support in my postgraduate study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Virtual reality (VR), or sometimes called *virtual environment (VE)*, is a combination of various technologies which enables a user to interact with an immersive and real–time computer–rendered 3D environment in a natural and intuitive way [1]. In terms of functionality, virtual reality is a high–end user interface which involves real–time simulation and interactions through multiple sensory channels, including visual, auditory, tactile, smell, or even taste [8]. The above definitions characterize one key feature of virtual reality: *real–time interactivity.* Real–time interactivity means that the computer should be able to detect the user's action and generate the corresponding feedback without any noticeable lag. Another key feature is the *sense of immersion.* With the appropriate equipments, virtual reality let users not just only see, but also feel what the computer has done in the simulated 3D space.

Human–computer interaction (HCI) has long been a popular research topic. As virtual reality is developing rapidly in recent years, 3D interaction draws more and more attention. Conventional input devices, such as keyboards, mice, joysticks, etc., failed to fulfill both two key features of virtual reality. Keyboards involve indirect interactions, and the other devices do not possess enough degrees of freedom to perform the complex tasks in 3D space. To solve the problem, many

3D instruments, such as Head Mounted Display, stereo glasses, gloves, 3D trackers, etc. are invented by different parties from various parts of the world. These 3D I/O devices, which are often referred to as VR devices, allow users to navigate the 3D virtual world and manipulate virtual objects.

Unfortunately, there is a serious drawback of these VR input devices. Unlike the multi–purposed input device, keyboard, VR input devices are designed specially for 3D interaction, but not command input. There are quite a number of common operations, such as loading and saving data, exiting an application, etc., which cannot be simply controlled by these devices. One solution for triggering these operations is to place 3D widgets in the virtual space. However, these widgets may occlude other parts of the virtual scene and distract the user. As the processing power of the computers grows rapidly, researchers are looking for more sophisticated alternatives, such as voice input and hand–function input.

Hands are our basic physical communication channel with the world. We touch, we feel and we manipulate real–world objects with our hands. When we perform various tasks, the hand forms different shapes and motions. In this dissertation, the term *posture* is defined as a static pose of the hand, while *gesture* refers to a sequence of postures over a short time serious. In addition to the physical functions, the hand also plays an important role in inter–personal communication, such as hand signal or sign language. Postures and gestures help us to express ourselves better. As an insight, posture and gesture recognition has aroused much interest as an alternative for human–computer interaction.

The term *virtual–hand* generally refers to a graphical representation of the hand which is emerged in the virtual environment. To fully represent and simulate hand functions, we must use a hand–tracking device to accept the whole–hand input, and utilize efficient recognition algorithms to identify the hand input. However, both technologies are immature. Current hand–tracking devices suffer from the problems such as low accuracy, high noise–to–signal ratio, and so on.

The major problem of posture and gesture recognition comes from the complexity of the human hand structure, which possesses about 30 degrees of freedom in its full representation. About 20 input parameters are required to specify a single posture. Another problem is the lack of accurate and standard measure for the hand. Both the sample patterns stored in the posture database and the input postures to be recognized may be imprecise. One source of the impreciseness is the mechanical noise comes from the hand–tracking device. The second source comes from us, human, as no one can repeat the same posture without any variance. Gesture recognition faces extra difficulties including time–varying signals and gesture segmentation.

As the title suggested, this research aims at developing a *real–time virtual–hand recognition system.* The term "virtual–hand recognition" indicates that the target system is not just a posture or gesture recognition system, but also a system which is able to integrated with real–time virtual reality applications practically. Virtual reality applications are usually very computation intensive. In order to achieve a smooth animation effect for visual feedback, it is necessary to refresh the display at about 20 to 30 frames per second. A number of jobs, such as collision detection, scene rendering, object behaviour modeling, etc., must be finished before displaying the next frame. Moreover, the latency time between the user's action and the computer's response must be very small to avoid user dizziness and sickness. Under these constraints, the virtual–hand recognition system must be *simple* and *efficient*, so that it would not affect the overall performance of these computation–intensive applications.

Following this introduction, the next two chapters provides the background information about this research. Chapter 2 consists three sections. The first section briefly examines the human hand model and the second section describes the current hand–tracking technologies. The last section discusses the problems encountered in virtual–hand recognition. Chapter 3 reviews the previous work which contributes to this field. The first half of the chapter describes the current

hand recognition approaches and the second half introduces some example systems which employ hand input.

A novel fuzzy posture recognition system is proposed and implemented in this dissertation. Chapter 4 gives an introduction to the key ideas of fuzzy concepts and then describes the proposed system in detail, including the goal, the features, the design and architecture of the system. Chapter 5 evaluates the performance of the proposed system.

Chapter 6 introduces a graphical user interface named *PostMan*, which stands for "Posture Manager", as the first application of the fuzzy posture recognition system. The application is developed for editing and validating posture records in system. The interface design and functions of the application will be described in the chapter.

Another application called *3DVWM (3D Virtual World Modeler)* is also developed. 3DVWM is a CAD–liked modeling tools, which is designed for rapid VRML modeling. Chapter 7 describes the system design and the use of virtual–hand recognition in the application.

Finally, Chapter 8 concludes the work, and Chapter 9 points out the future work of the research.

Chapter 2

Virtual–hand Recognition

The hand is an important part of our body. It is dextrous and flexible, so that we use our hands to manipulate real–world objects in our daily life. Moreover, using postures and gestures let us express ourselves better and hence communicate with others more effectively. The hand serves as an excellent interface between human and computers, especially for 3D virtual reality applications. The term *virtual–hand* generally refers to a graphical model of the human hand which is emerged in the virtual world. As the human hand is very complex in its full representation, simplified virtual–hand models are usually used in practice.

Virtual–hand recognition is defined as the utilization of recognition algorithms to identify the postures and gestures in the virtual environment. This definition features that the recognition algorithm not only can recognize postures or gestures, but also suitable for incorporating with VR systems, which are usually very computation intensive. In other words, the recognition algorithm must be simple and efficient enough, so that it would not affect the normal routines of VR systems.

One enabling technology of virtual–hand recognition is the hand–tracking devices, which monitor the pose and orientation of the hand in the 3D space. Hand–tracking devices have now become a standard equipment in virtual reality. With

these devices, the user is able to manipulate *virtual objects* with the hand and the computer, on the other hand, generates visual, audio, or even tactile feedback to the user.

This chapter covers the related background information about virtual–hand recognition. Section 2.1 briefly examines the hand model. Section 2.2 describes the current hand–tracking technologies. Section 2.3 discusses the problems encountered in virtual–hand recognition.

2.1 Hand model

2.1.1 Hand structure

From a surgical point of view, the human hand is a complex organ which is composed of bones, ligaments, muscles, vessels and skin. Bones make up the skeleton of the hand and ligaments connect the bones to form joints, which restrict the freedom of movement of the hand. Muscles drive the hand to form different shapes. These tissues are wrapped by skin, which protects them from physical damages. Structurally, the human hand consists six major parts: the *palm* and the five fingers, namely *thumb*, *index*, *middle*, *ring* and *pinkie*. The last four fingers are structurally identical, where each of them have three segments, called *phalanxes*: the distal phalanx (the one furthest from the palm), middle phalanx and proximal phalanx (the one connecting to the palm). The structure of the thumb is slightly different from the other fingers. There are only two obvious phalanxes, and a large proportion of the third segment is connected to the palm by skins. The special structure of the third segment allows us to perform *anteposition and retroposition*[1].

As the human hand is too complex, it is inefficient and impractical to model the hand in full manner. Instead, simplified hand models are usually used in

[1] Anteposition is the bending of the thumb towards the other side of the palm, while retroposition is the reverse action of anteposition.

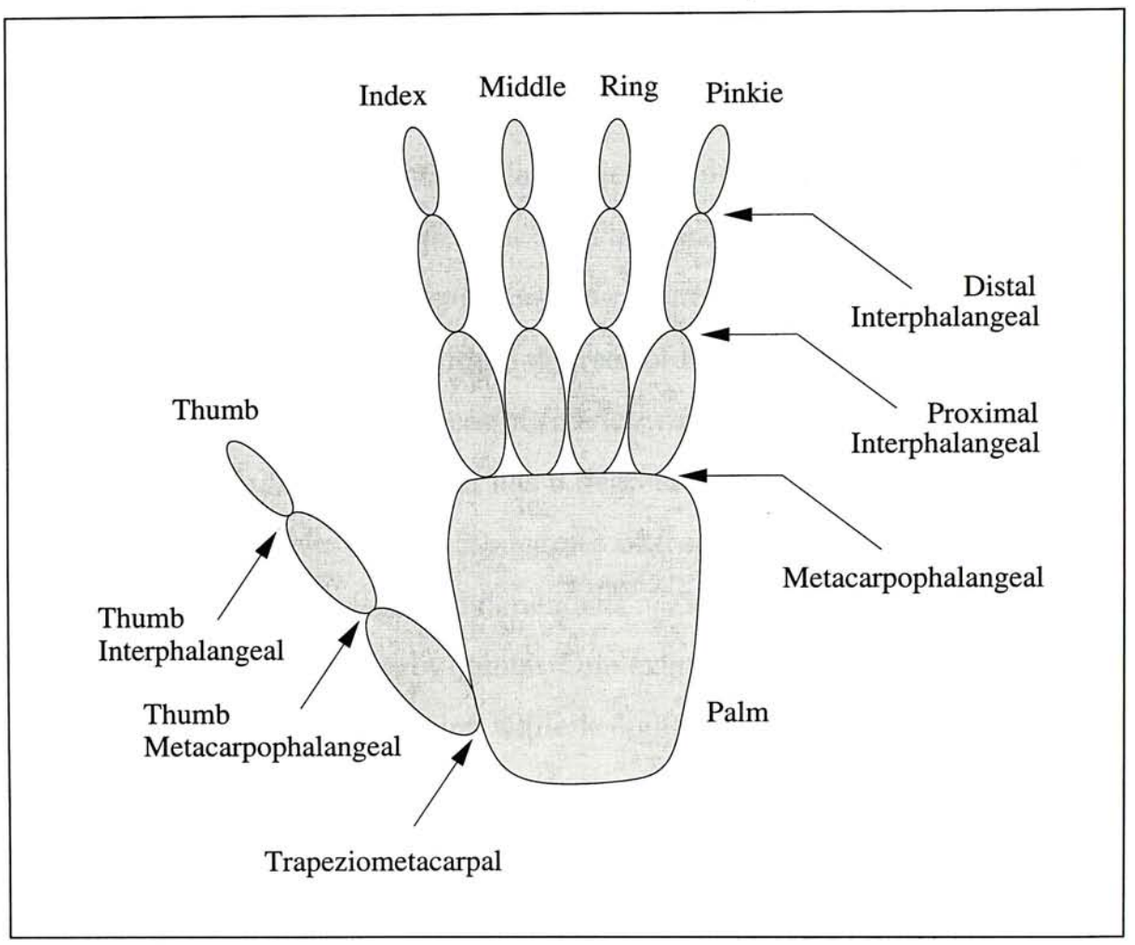


Figure 2.1. The human hand model.

virtual–hand systems. There are many tricks to simplify the hand model. The most common one is to represent the palm by one single non–moveable object, despite that it does have a limited mobility (see Section 2.1.2 for detail). Other tricks include using only two phalanxes rather than three for the finger models, neglecting the *abduction and adduction*[2] of the fingers, etc. Figure 2.1 shows a commonly adopted hand model.

2.1.2 Motions of the hand joints

The *degree of freedom (DOF)* of an object is defined as the number of independent variables needed to represent it. The human hand possesses a very high degrees of freedom in its *full representation*, contributed mainly from the finger joints. Table 2.1 shows the individual degrees of freedom of the hand joints above the wrist, which totalled 23 degrees of freedom. Considering also the 3D position and orientation of the hand, which has 6 degrees of freedom[3], the full representation of the hand involves totally 29 degrees of freedom. Some degrees of freedom in the hand are not completely 'independent'. They are constrained by the muscles and ligaments of the nearby joints. One example is that we could hardly flex the proximal interphalangeal joint while keeping the distal interphalangeal joint unflexed.

In addition to the degree of freedom, which provides a mean for measuring the complexity of the hand model, sometimes it is also necessary to concern about the *range of motion* of the hand. Unlike the degrees of freedom, the ranges of motion of the joints varies from person to person, and thus can be a source of uncertainty in posture and gesture recognition. On average, the *extension–flexion*[4] motion

[2]Abduction and adduction refer to the side by side movements of the fingers. Abduction moves the fingers away from each other and adduction is the vice verse.

[3]The free motion of the hand is enabled by the joints of the arm. The shoulder has 4 degrees of freedom, the elbow has 1, and the forearm has 1 (rotation about itself).

[4]Extension is the unflexing of a joint, and flexion is the reverse action of extension.

Joint		DOF
Thumb	Thumb Interphalangeal (TIP)	1
	Thumb Metacarpophalangeal (TMP)	1
	Trapeziometacarpal (TMC)	3
Fingers (×4)	Distal Interphalangeal (DIP)	1
	Proximal Interphalangeal (PIP)	1
	Metacarpophalangeal (MCP)	2
Palm	Metacarpocarpal (×2)	1
Total		23

Table 2.1. The degrees of freedom (and their common abbreviations) of the hand joints above the wrist.

of the finger joints has a range of around 90 to 100 degrees, and the abduction–adduction motion has a range of around 40 degrees. It is worth to note that, although the palm has 2 degrees of freedom, contributed by the metacarpocarpal joints near the wrist, it has a very limited range of motion (up to several degrees only). These two joints are often omitted in most hand models.

2.2 Hand–tracking technologies

In order to monitor the pose and 3D orientation of the hand, and translate them into electrical signals which can be recognized by computers, researchers from various parts of the world started to develop different kinds of hand tracking devices since the 1970s. Although the hand–tracking technology has been developed for more than two decades, existing devices are still far from ideal. The main difficulty comes from the complexity of the human hand.

There are various existing hand–tracking techniques. Most of them can be classified into two basic types: glove–based tracking and imaged–based tracking. The following explains the principles, examples, and benefits of the two types of tracking techniques.

2.2.1 Glove–based tracking

Glove–based tracking is probably the most common hand–tracking technique nowadays. It detects the hand pose by using a number of small sensors attached to particular positions on a light–weight glove. Hand position and orientation is monitored using a 3D tracker attached to the hand. The glove[5] is usually connected to a special hardware, which returns the sensor and tracker readings to the computer. In general, glove–based tracking is simpler and more efficient than other techniques.

A number of gloves have been invented in the past decades. The first glove, *Sayre Glove*, is developed, based on the idea of Rich Sayre, at the University of Illinois, Chicago in 1976. Although the glove itself is inaccurate and it is impractical to be used as a gesture device, the idea raised the attention of many researchers. Afterwards in the 1980s, various parties in the world built their own gloves. A comprehensive survey of glove technologies is given by Sturman and Zeltzer in 1994 [43].

In the early 1980s, researchers at the MIT Architecture Machine Group and the MIT Media Lab built the *MIT LED Glove* for real–time computer graphics animation. In 1983, Grimes of the Bell Telephone Laboratories developed the *Digital Data Entry Glove* [17], which is designed for recognizing alphabets for the deaf. Although the above gloves are much better than the Sayre Glove, they are still problematic and never put into actual use. The first commercialized glove is the VPL DataGlove, developed by Zimmerman and Lanier in 1987 [54]. Despite that both the accuracy and the sampling rate of the glove are unsatisfactory for most applications, the VPL DataGlove is still widely used by many research institutions for its reasonable cost. In 1989, an exoskeleton glove, called *Dexterous HandMaster (DHM)*, is developed by Exos. This glove accurately measures 20 degrees of freedom of the hand, but is uncomfortable to wear. In the same year,

[5]Here (and in the following context) the term "glove" refers to "glove–based hand–tracking device", rather than its semantical meaning.

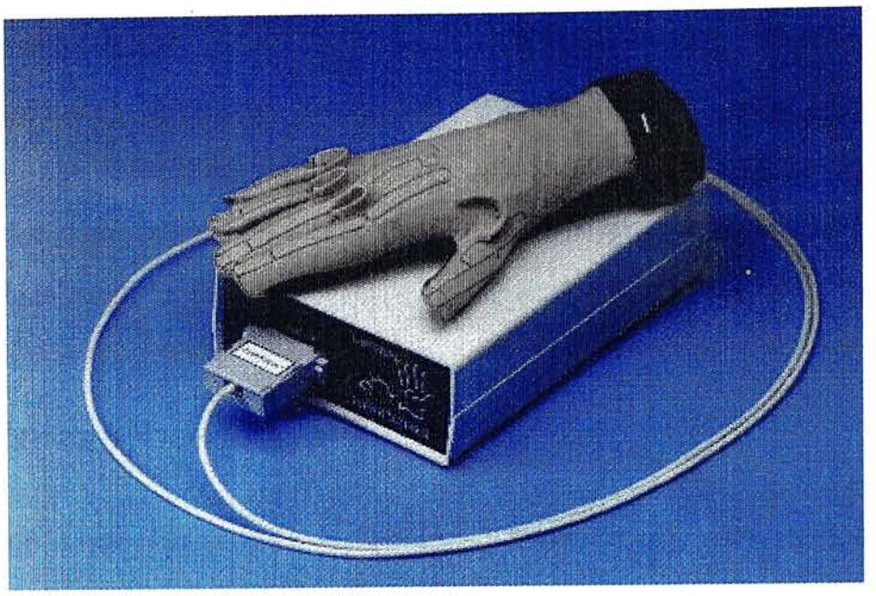


Figure 2.2. An 18–sensor CyberGlove and the CyberGlove Interface Unit (CGIU), which is responsible for the amplification and digitization of the sensor signals of the CyberGlove. (Picture adapted from *http://www.virtex.com/*, the official homepage of Virtual Technologies.)

a low–cost product, the *Power Glove*, is manufactured by a toy company called Mattel for home video games. As the glove is neither accurate, nor comfortable, it disappeared in the market after a few years.

The most common glove right now is probably the *CyberGlove*, shown in Figure 2.2. The CyberGlove is first developed by James Kramer at Stanford University in 1990, and is now commercially available from the Virtual Technologies. The glove can be equipped with 18 or 22 flex sensors[6], where each sensor has a resolution of 0.5 degree, and the Polhemus 3D tracking sensor, which detects the 3D orientation and position of the hand. In 1996, a tactile feedback option for the 18–sensor CyberGlove, called *CyberTouch*, is developed by the same company. Software–controlled vibrotactile stimulators are placed on each finger and the palm to simulate simple tactile feedback.

Although the glove–based devices are very popular, there are certain limitations on them. Some users found that it is uncomfortable to wear a glove, especially after a long period of usage. The mechanical parts attached to the glove,

[6]The 18–sensor CyberGlove does not detect the DIP joints of the last 4 fingers.

such as sensors, cables, power cords, etc., may sometimes inhibit the free movements of the hand. Moreover, re–calibration for individual user is usually required to maintain the smoothness and stability of the glove. Periodic re–calibration is necessary for some gloves, because the sensor materials are sensitive to environment changes, such as temperatures, electric and magnetic fields, etc.

2.2.2 Image–based tracking

As their name suggested, *image–based tracking*, or *camera–based tracking*, monitors both the pose and the motion of the hand by analyzing the captured images of the hand. Early systems use a single camera to capture the silhouette images of the hand and analyze the images by using image processing and computer vision techniques. These systems cannot recognize 3D postures since silhouette can only provide 2D informations. Later systems use 2 or more cameras and reconstruct the 3D postures from the stereo images obtained.

Although image–based systems do not attach any mechanical device to users and thus provide a more natural and convenient interface, there are quite a number of drawbacks. First, analyzing hand images involve a lot of image processing and computer vision techniques, such as image segmentation, filtering, etc., which is usually very time consuming. Therefore, most image–based systems failed to perform recognition in real–time. Second, since the palm and the fingers are naturally close to each other, occlusion happens frequently as the hand and fingers move. Using more cameras can probably solve this problem, but more computation is then resulted. Moreover, noise is created by the skin deformation around finger joints. Accuracy can be seriously affected by the above two factors. Lastly, as the hand is not physically connected to any hardware device, it is impossible to implement tactile or force feedback on these systems.

2.3 Problems in virtual–hand recognition

The problems encountered in virtual–hand recognition can be divided into two types: problems in posture and gesture recognition, and how to utilize the recognition algorithms in real–time VR applications. The following explains the common problems in virtual–hand recognition.

2.3.1 Hand complexity

Obviously, posture and gesture recognition falls into the category of pattern recognition problems, which involve the classification of an input pattern according to the known sample patterns. In general, the difficulty of a pattern recognition problem is directly related to the complexity of the *pattern representation.* As mentioned in Section 2.1, the human hand possess up to 29 degrees of freedom in its full representation. Even some of them can be safely ignored, about 20 input parameters are usually required to describe a specific posture. Gesture recognition usually involves a lot more parameters, depending on the data representation scheme of the gestures in the recognition algorithm.

2.3.2 Human variations

Similar to some pattern recognition problems, such as speech recognition and on–line hand–writing recognition, the posture and gesture recognition suffers from one common problem: there is no *"absolute"* way to represent a pattern. For example, everyone knows how to write an "A", but none of them writes it perfectly the same as others. Even the same person cannot reproduce it exactly again. In other words, there are infinite "correct" ways to represent a pattern. This kind of variation is very significant in posture and gesture recognition.

2.3.3 Immature hand–tracking technologies

Another source of inaccuracy comes from the noise of the hand–tracking device. As mentioned earlier in this chapter, the current hand–tracking technologies are still far from ideal. Most common glove–based devices suffer from the problems such as unsatisfactory sensor resolution, low accuracy, high noise–to–signal ratio, and so on. Image–based tracking is even worse. Current image–based systems are too computation expensive to perform recognition in real–time.

2.3.4 Time–varying signal

Just like speech and on–line hand–writing, gesture is a time–varying signal. Therefore, the pattern representation of gesture recognition involves both the space and time domains. This makes gesture recognition more complex than other pattern recognition problems, which involve either one domain only. Moreover, signal segmentation in gesture recognition is extremely difficult, as there is no obvious starting or ending point of a gesture when the hand is moving continuously in the 3D space.

2.3.5 Efficiency

Frame rate is one of the most important concerns in virtual reality applications. In general, virtual reality systems must maintain at about 20 to 30 frames per second to provide a smooth visual effect of animation [14]. A number of necessary tasks, such as collision detection, scene rendering, object behaviour modeling, etc., have to be completed before displaying the next frame. The processing time allocated for each task is definitely limited. It is not reasonable to devote a large proportion of computation resources to the posture and gesture recognition process. Another concern is the *latency*, which is the time between the user's action and the computer's response. High latency usually results in user dizziness

and sickness. These two concerns impose a very high demand on the processing power of the computers. Although the processing power of the computers has been increased rapidly in recent years, it still cannot fulfill the demand of VR applications. Therefore, practical virtual–hand recognition algorithms must be simple and efficient, so that it would not affect the regular routines of the VR application.

Chapter 3

Previous Work

This chapter discusses previous work which contributes to this study. Section 3.1 describes the development of the posture and gesture recognition algorithms and examines the current approaches, including template matching, neural networks, statistical classification, discontinuity matching, model–based analysis and Hidden Markov Models. Section 3.2 introduces some existing systems which employ virtual–hand input.

3.1 Posture and gesture recognition algorithms

The first posture recognition system [17] is introduced by Grimes, along with his *Digital Data Entry Glove* as a whole, in 1983. The system is designed specifically for recognizing the American Sign Language alphabets. The sample postures for the alphabets are hard–coded electronically into particular combinations of the glove's sensor readings. This hardware approach is efficient and robust, but inflexible, in a sense that only a small number of pre–defined postures can be recognized. There is no mechanism to recognize user–specified postures.

Watson's technical report [48], published in 1993, gives an in–depth analysis of some posture and gesture recognition approaches, including template matching,

neural networks, statistical classification and discontinuity matching. Despite that some of these approaches are still being widely used, two new approaches, such as model–based analysis and Hidden Markov Models, are proposed in recent years. The following explains the general ideas, as well as the advantages and limitations of these approaches.

3.1.1 Template Matching

Template matching is the first available software approach in the field. Nevertheless, it is still the simplest algorithm among all existing ones.

In the template matching approach, a posture is represented by a *template*, which is a data structure containing a set of maximum and minimum valid values of each parameters of the posture. Figure 3.1 illustrates the idea of the template. In the recognition process, the value of each parameter of the input posture is compared with the corresponding maximum and minimum values of all templates. The absolute value of the difference of each parameter is summed for each template. The posture with lowest total difference will be chosen, if the difference value is below a pre–defined threshold.

The earliest system that applied this technique is the *VPL's Gesture Recognizer* [54], proposed by Zimmerman and Lanier in 1987. Although the term "gesture" is used, the system can recognize postures only. At each sample time, the sensor values read from the VPL DataGlove is fed into the recognizer and the recognizer returns the most similar match among all postures. A simple template editor is also provided with the recognition system to let users create or modify templates of the postures stored in the recognizer.

The major advantages of this approach are simple and flexible. The idea of the template representation is simple enough for most users to understand and define their own postures quickly and easily. In addition, the approach is very efficient, due to its simpleness. However, the correctness of the recognition system can

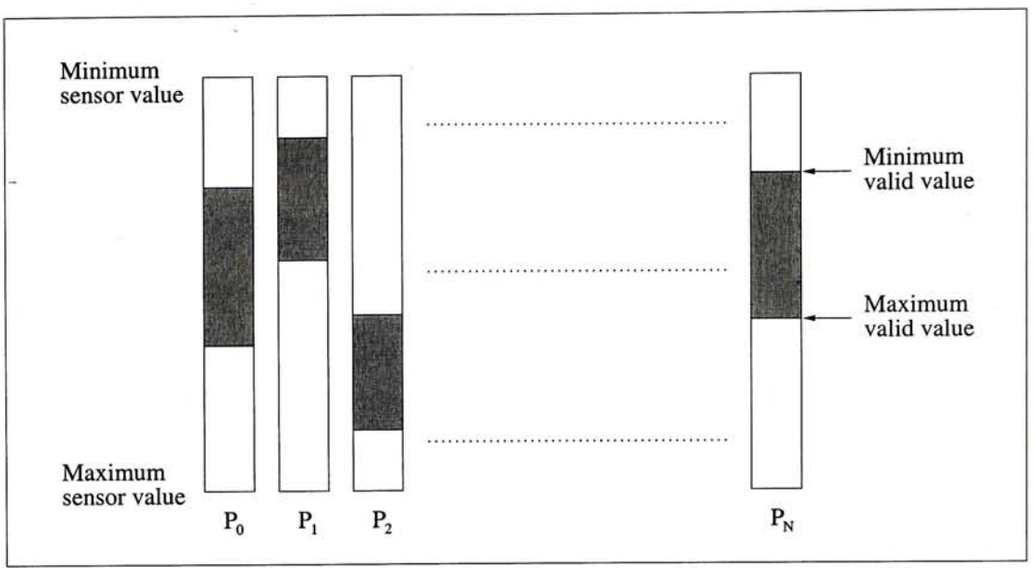


Figure 3.1. A graphical representation of a template. The shaded areas indicate the valid ranges of the input parameters P_0, P_1, ..., P_N of a posture.

be seriously affected by noise. In order to deal with the inaccuracies, mentioned in Section 2.3, the range of each sensor must be wide enough, which may be up to 30% of the full sensor range, to maintain a reasonable recognition rate. As a result, the recognizer can only support a very limited number of postures. The recognition rate drops significantly when more postures are stored, due to serious range overlapping of the templates. For instance, the VPL's Gesture Recognizer, which uses the VPL DataGlove as the input device, supports about 10 to 15 postures only.

3.1.2 Neural networks

Neural works are introduced to simulate human intelligent in computers and have been successfully applied in many pattern recognition systems. As a result, researchers started to study the possibility of using neural networks in recognizing postures and gestures. There are quite a number of successful examples, notably Fels' *GloveTalk* [13] and its extension, *GloveTalk II* [12]. More detail descriptions of Fels' systems is given later in Section 3.2.

Neural networks are *fine–grain connectionist models*, which consists a large number of highly inter–connected *processing elements* configured in regular architectures. Each processing elements, or sometimes called *node*, collects the values from its input connections, performs a pre–defined operation on the values, and finally feeds the resulted value to the output connections. Every input connection is associated with a *weight*, indicating the importance of the input value to the node. The process of obtaining these weights is called *training*, or *learning*, which is necessary for the neural network to be put into actual use. During the training, labeled examples of the recognition patterns are passed to the neural network so that the network can "remember" these patterns. There are various types of neural networks, which can be classified according to the network topology, architecture and learning algorithm.

Neural networks have many desirable features in posture and gesture recognition. First, neural networks are highly *noise–tolerant* and handle incomplete information pretty well. This feature solves the inaccuracy factors in the posture and gesture recognition. Moreover, re–calibration for different users can be avoided as the systems are adaptive. However, this approach has also a number of serious drawbacks. First, there is no systematic way to design the architecture and topology, as well as the learning algorithm of the neural network. These elements must be determined by trial–and–error. This leads to a long system design stage. Another problem is the training of the network. A large number of training samples, which can be in terms of hundreds or thousands, are required for training the network. The training process must be restarted after adding new samples or removing existing samples. The time taken for training the network cannot be estimated. It may take hours or even days. Sometimes the training may even fail if the training data contains contradicting entries.

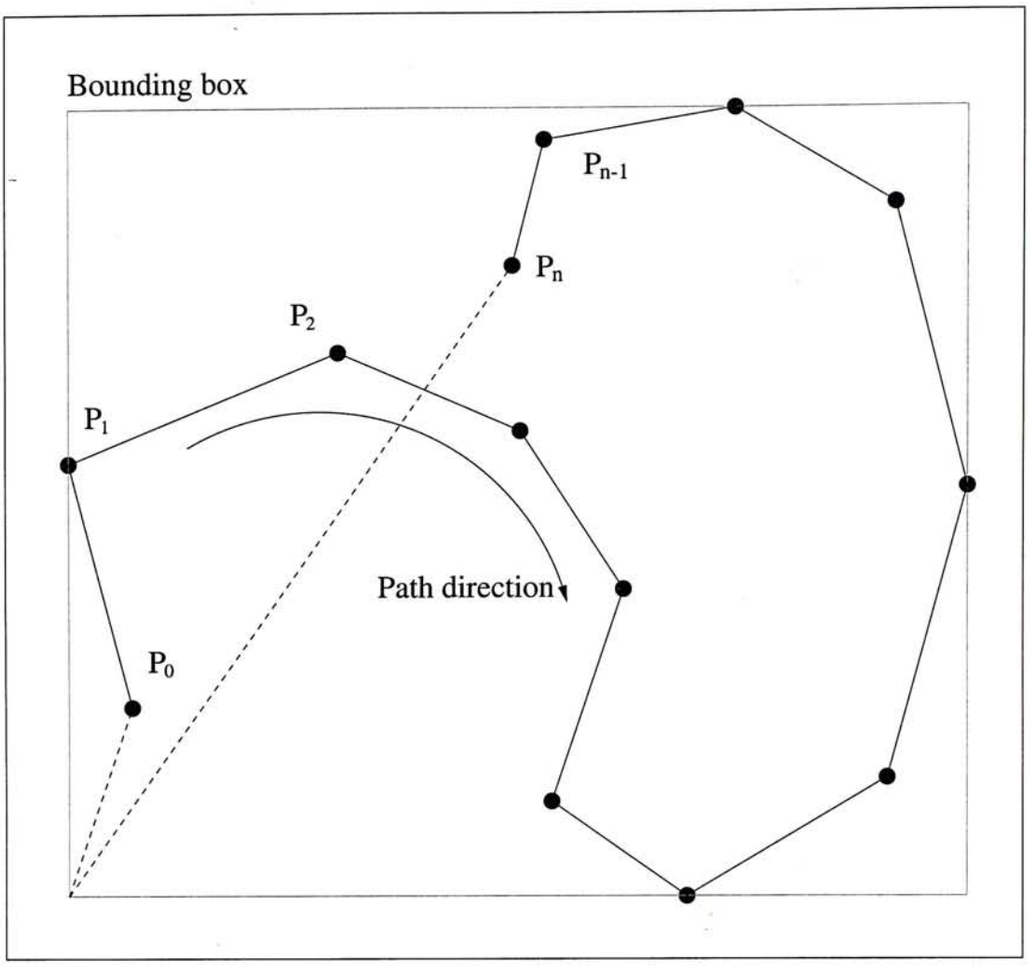


Figure 3.2. Some example features extracted from a gesture path.

3.1.3 Statistical classification

Unlike the previous two approaches, the statistical classification is far less common in the field. The statistical classification approach is originated by Rubine, who applied the method in his system, *GRANDMA*[1] [39], which is an object–oriented toolkit for building gesture–based applications.

In Rubine's system, "gesture" is simply defined as a path of points on a 2D plane. The recognition process applies statistical decision theory, such as Bayesian maximum likelihood theory, to classify an input gesture. The classification is based on a set of *features* extracted from the gesture path. The features chosen are usually the geometrical information, such as bounding box, total length of

[1]GRANDMA stands for Gesture Recognizers Automated in a Novel Direct Manipulation Architecture.

the path, sine and cosine value of the starting point and ending point, etc., as illustrated in Figure 3.2. The major strength of Rubine's approach is that it handles dynamic patterns in a simple way. Unfortunately, there are several deficits of the approach. First, the recognition is not fully automatic. The start and end of a gesture path must be indicated explicitly by the user. Second, there is no general rule which guides the selection of gesture features. The feature set used in the GRANDMA system was empirically determined by Rubine himself to work well on several different gesture sets, including the digits and the alphabets. Moreover, training is required as the classification is based on statistical information of the sample patterns. Rubine reports that 15 training examples for each gesture is usually sufficient.

Rubine's system is extended by Sturman [42] to support three dimensional paths and to deal with the deficits mentioned above. More complex gestures are can be recognized by combining the results of applying the method on multiple paths of key positions of the hand, such as finger tips, palm, wrist, etc. Another significant change is that the feature analysis in the extended system is continual. Recognition can be done without indicating the start and end points explicitly. However, the problem of manual feature selection is remained unsolved.

3.1.4 Discontinuity matching

The discontinuity matching approach is proposed by Watson and O'Neill in 1995 [50][49] as the gesture recognition system of the *GLAD-IN-ART*[2] project.

Discontinuity matching is the extension of the classic template matching technique to recognize dynamic patterns. Instead of raw sensor readings, the discontinuity matching approach represents a gesture by a template of a set of *sequences of discontinuities*, which, instead of using control points, approximate the motion

[2]GLAD-IN-ART stands for Glove-Like Advanced Interface for the Control of Manipulatory and Exploratory Procedures in Artificial Realities.

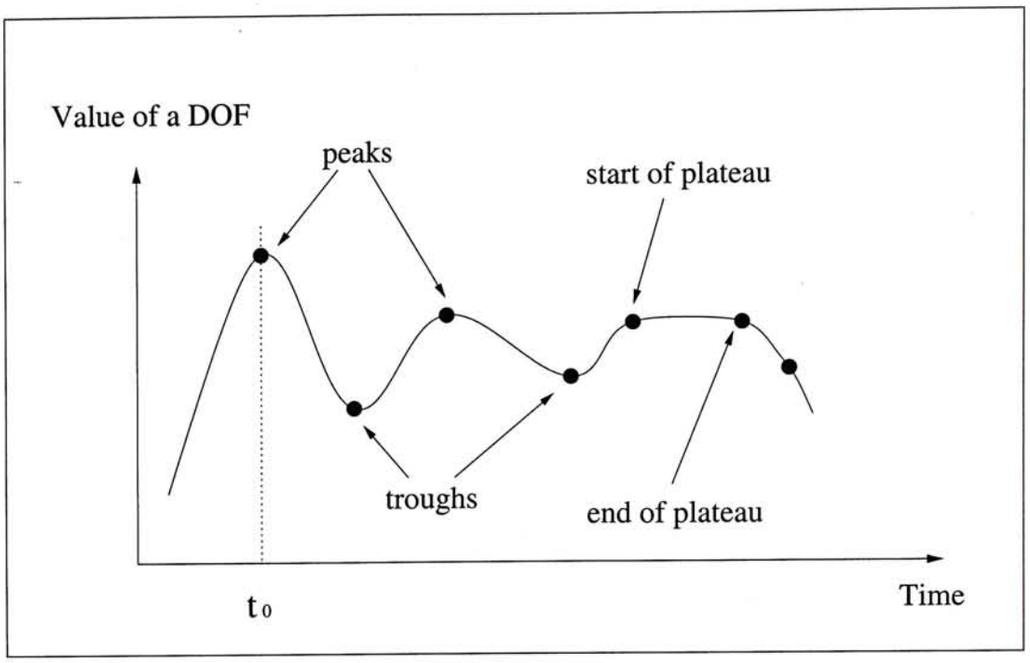


Figure 3.3. The changes of one of the degrees of freedom in the hand over time in performing a gesture.

curves of the degrees of freedom of the hand. As illustrated in Figure 3.3, a discontinuity, such as a peak, a trough, the start and end of a plateau, is the first derivate of the changes of a degree of freedom over time. During the recognition process, the discontinuities are extracted by analyzing the angular velocity of the degrees of freedom. The sequence of discontinuities for each degree of freedom considered are then compared with all sample gesture templates stored in the system. The process is the classic template matching technique with some necessary minor changes.

There are a number of advantages of discontinuity matching over the previous approaches. First, the gesture representation in this approach is relatively *motion oriented*, which is a significant feature in recognizing dynamic patterns. As the approach considers discontinuities rather than control points, the system is robust to scaling of gesture pattern over time. This also makes the system less sensible to input noise. Jitters can be removed from the input data by using simple filters. Nevertheless, the major limitation of the approach is the time complexity. The computations involved is more complex compared with the previous approaches.

The template comparison is not as simple and efficient as in the classic template matching. Watson recommended that future work should be concentrated on more effective indexing of sample gestures stored in the system such as tree or hash table. Another minor deficit of the system is that gesture training is only semi–automatic. A certain extent of human intervention is required.

3.1.5 Model–based analysis

In 1995, Lee and Kunii at the University of Aizu proposed a method to recognize hand postures [30]. Stereo camera images captured are compared to the sample hand models by an iterative improvement approach, guided by a set of constraints.

There are a number of disadvantages of this system. Lee and Kunii pointed out in their report that both the efficiency and accuracy of the system are unsatisfactory. It takes about 45 minutes to obtain a precise recognition result on an SGI Iris workstation. Obviously this is not acceptable in real VR applications. Taking less iterations improves the efficiency, but the accuracy, on the other hand, is seriously affected. In addition to the number of iterations, the accuracy of the is also affected by several more factors. First, noise is introduced by skin deformations around joints and the occlusion between fingers and the palm. The difference between the sizes of the hand models and the user's hand may also led to computation errors.

3.1.6 Hidden Markov Models

The success of Hidden Markov Models (HMM) in speech recognition and on–line hand–writing recognition has drawn researcher's attention recently. In general, Hidden Markov Models are effective in extracting and recognizing both static

and dynamic patterns, and are thus suitable for both posture and gesture recognition. The approach is very robust and quite efficient. Most importantly, segmentation of gestures is implicitly handled in the models. Although this approach is very new in posture and gesture recognition, both Liang and Nam showed their successes recently. Liang's sign language recognition system [31] uses HMM to recognize 50 fundamental postures in the Taiwanese Sign Language, while Nam [34] applies HMM in recognizing the hand movement path over time.

Nevertheless, the Hidden Markov Model approach has some serious limitations very similar to neural networks. There is no formal rule which guides the development of the system. The design of the important elements of the system, such as model formulation, pattern representation, etc., may not be obvious. The number of states in the HMM must be fine–tuned for each sample pattern and determined empirically. The HMM approach has a slightly edge over neural networks that the training time is shorter, and re–training is not necessary when adding or removing sample patterns. However, large number of labeled training examples is still required to build the statistic basis of the model.

3.2 Hand–input systems

There is no doubt that hand interfaces are more natural and convenient. With the invention of hand–tracking device and the introduction of hand recognition algorithms, it is now possible to use our hands to communicate with computers. Sturman conducted a comprehensive study on the topic *whole–hand input* in his doctoral thesis [42]. He mentioned that whole–hand input possess three important features, including *naturalness*, *adaptability* and *dexterity*, which make it suitable for various applications. The following introduces some hand–input systems.

3.2.1 Gesture languages

Interpreting gesture languages is one of the driving tasks of hand–input systems. The two major application areas of gesture languages are command input in computer applications and human communication.

Gimes' Digital Data Entry Glove Interface [17], developed in 1983, is probably the earliest project in the field. His system uses a hardware approach to recognize the American Sign Language (ASL) alphabets. Afterwards, many other systems are developed using various techniques. In 1989, Kramer, who created the CyberGlove, used a statistical approach to translate ASL into spoken English [27]. Postures are mapped to predefined ASL letters or symbols and stores in an output buffer. When a word phrase is complete, the result will be produced by a voice synthesizer. Fels did a very similar work, but in gesture level. His *GloveTalk* [13] capture the hand motions and translate it to speech. He used five independent neural networks to analyze 5 different properties of hand motion, including the trajectory, shape, direction, displacement and speed. The system recognizes 203 signs based on 66 postures combined with 6 movement stages. The result is directed to a speech synthesizer. Fels extended his work and developed *GloveTalk II* [12] in 1996. The extended system used only 3 feed–forward networks and is more efficient. More features are supported, including unlimited vocabulary, direct control of speech frequency and volume. In 1996, Liang proposed a Hidden Markov Model approach to recognize Taiwanese Sign Language (TSL) [31]. His system recognizes 50 TSL fundamental postures and analyzes the posture streams by using dynamic programming techniques.

3.2.2 3D modeling

3D interactive modeling is one of the major application areas of virtual reality. Most commercial 3D modeling tools provide only 2D interfaces and greatly reduce the effectiveness of the modeling tasks. With 3D hand interface, the modeling is

more convenient and efficient. Direct manipulation, which is one of the most important concerns in human–computer interaction, is also made possible.

In recent years, the topic has become very popular and many 3D modeling systems are proposed. The VLEGO system [22], proposed by Kiyokawa *et al.* in 1996, models virtual environment based on toy blocks. The system uses two 3D pointing devices to simulate two–handed environment. The primitives in the system are all rectangular toy blocks. Several more systems are developed in 1997. Kameyama's Virtual Clay Modeling System [20] supports free–formed direct manipulation on the shape of a virtual object. The input device of this system consists of a 3D tracker, a 3D mouse and a tactile sensor. Billinghurst *et al.* proposed a virtual scene creation tools called 3D Palette [5]. The 3D Palette accepts various input, including tablet and digitizing pen, 3D tracker and even simple voice commands. Korida and Utsumiya at the Oita University incorporates the TGSH, which stands for Two–handed Gesture environment SHell, with a 3D geometric modeler [36]. The system is equipped with two CyberGlove's and performs dynamic gesture recognition using a recurrent neural network and a standard back–propagation algorithm. Due to the high CPU intensity nature of the dynamic gesture recognition, the system runs on a network of machines, including one SGI Onyx, two R5000 SGI Indy's and a Sun Ultra Enterprise 2. The system recognizes 6 two–handed gesture patterns. Later on, they built another interactive 3D interface called *CHINA* [25] on top of TGSH. The system allows the user to create and manipulate 3D objects. It uses a feed–forward neural network for static posture recognition and the same recurrent neural network in TGSH to interpret hand gestures.

3.2.3 Medical visualization

Volume visualization is a technique which applies volume rendering algorithms to create images of three dimensional data. Medical visualization is one of its major application. Medical volume data, such as brain, lung, etc. obtained by using

CT scanners (Computed Tomography) or MRI (Magnetic Resonance Imaging) scanners, are represented by 2D projections of the images or stereo images. Users are allowed to navigate, or manipulate the volume data with the appropriate VR devices. As the visualization itself is very computation intensive, many existing visualization systems employ very simple user interface to reduce the overall workload of the system. The Virtual Workbench, developed by Poston and Serra [38], uses a simple 3D tracker as the input device.

Recently, Clifton and Pang developed a direct manipulation system [10] that allows the user to intuitively slice, dice, and carve the volume data set under investigation using hand postures. The posture recognition approach used in the system is a very simple template matching. Only six postures can be recognized. Nevertheless, the system demonstrated the feasibility of incorporating hand–input in medical visualization systems.

Chapter 4

Posture Recognition

As mentioned in Chapter 2, one major problem of posture recognition is that both the sample postures and the postures to be recognized may be imprecise. This kind of imprecise knowledge can be best dealt with using fuzzy concepts. A fast and simple posture recognition algorithm using fuzzy model is proposed. This chapter first presents the basic ideas of fuzzy sets and fuzzy logic, and then describes the proposed fuzzy posture recognition system.

4.1 Fuzzy concepts

The basic concepts of fuzzy sets and fuzzy logic are introduced by Zadeh in 1965 [51]. Fuzzy set theory can be regarded as a generalization of the conventional crisp set theory. It provides a mathematical way to represents *vagueness* and *uncertainty* in everyday life. Fuzzy interpretations are usually more natural and useful in solving real life problems, and thus offer a better interface between human and computer. Fuzzy logic, as an extension of the traditional two–valued boolean logic, manipulates this kind of data in a mathematical way based on the fuzzy set theory. Fuzzy sets and fuzzy logic has been successfully applied in various field, including expert system, approximate reasoning, pattern recognition, image

processing and many others. The following explains the major breakthroughs of the fuzzy concepts.

4.1.1 Degree of membership

In the conventional crisp set theory, membership relation has only two outcomes: either "in the set" or "not in the set". Unfortunately, the crisp concept fails to represent some relations in an useful way. For instance, we cannot define the relation 'tall' in this way: "Everyone over k cm is tall, otherwise he is not tall", as it is not sensible to say $k - 0.1$ cm should is not 'tall'. It does not exist a clear–cut boundary in practice which distinguishes 'tall' and 'not tall'.

These relations, which are vague or imprecise in nature, are better represented using fuzzy concept. In the fuzzy set theory, there is no clear boundary which defines the membership relation. Figure 4.1 illustrates the difference between the membership concept of the two theories.

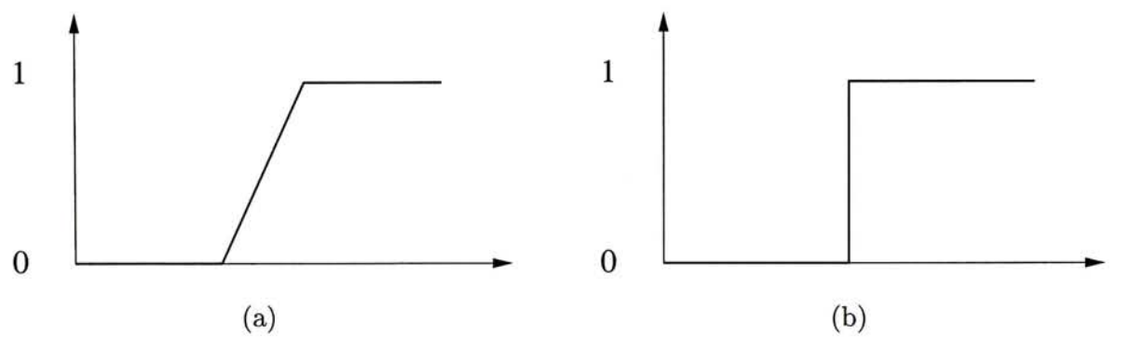


Figure 4.1. Membership function of (a) a fuzzy set and (b) a crisp set

The two graphs in Figure 4.1 are usually referred to as *membership functions*, which map each element of the fuzzy set to its range space. The vertical axes in the graphs represent the *degree of membership* of a fuzzy element. The degree of membership, which can be any real number ranges from the unit interval 0 to 1 inclusively, is used to measure how likely that an element belongs to a fuzzy set. Linguistically, the degree of membership can be regarded as modifiers such

as 'very', 'quite', 'rather', etc. Most common membership functions, as the one shown in Figure 4.1(a), changes gradually from 0 to 1 and thus introduces a soft boundary over a certain range. The conventional crisp set can be regarded as a special case of fuzzy set, whose membership relation is defined by a step function (Figure 4.1(b)).

4.1.2 Certainty factor

In addition to vagueness, very often we have to manipulate uncertain, or even incomplete information, as illustrated in the following example. Someone was sick and he went to the clinic. The doctor asked his patient for his symptoms and then conducted a simple diagnosis. Finally, he told his patient, "*Most probably*, you are ...". The wordings of the doctor reflects that he is not 100% sure about the result, because the information he obtained contains uncertainty. The symptoms provided by the patient may not be completely correct and, moreover, there may be some hidden symptoms that cannot be spotted out during the diagnosis.

An important feature of the fuzzy concepts is its capability to deal with uncertainty. While the degree of membership successfully handles vagueness and imprecision, the concept of *certainty factor*, proposed by Shortliffe in his expert system *MYCIN* in 1976 [41], is introduced to deal with uncertainty. Certainty factor, which is also a real number lies between 0 and 1, describes how certain a piece of information is true. In other words, it is a *measure of belief*, which is usually expressed by "absolutely", "possibly", "maybe", "unlikely" etc. in human language.

4.1.3 Evidence combination

Sometimes there may be two or more pieces of information which directs to the same conclusion. In the "doctor and patient" example mentioned in the last

section, the doctor drew his conclusion based on two things: the patient's description of his symptoms and the result of his diagnosis. Although there are some uncertainties, the doctor is quite confident (he said "Most probably") about the conclusion, since both pieces of evidence lead to the same conclusion. Combining evidence improves the reliability of the result.

The *Z–II* system, a fuzzy expert system shell developed by Leung in 1988 [29], uses a method called *evidence combination* to calculate the reliability of the conclusion from the certainty factors of individual evidence. Suppose that there are two events, whose certainty factors are cf_1 and cf_2 respectively. The certainty factor of the combined evidence, cf_{12}, is calculated by:

$$\begin{aligned} cf_{12} &= cf_1 + cf_2 \times (1 - cf_1) \\ &= cf_1 + cf_2 - cf_1 \times cf_2 \end{aligned} \tag{4.1}$$

If more evidence is available, the overall certainty factor can be obtained by applying Equation 4.1 on cf_{12} and cf_3 to find $cf_{1..3}$ and so on.

4.2 Fuzzy posture recognition system

Pattern recognition is one of the major application areas of fuzzy theories. Fuzzy models have been successfully applied to many pattern recognition problems, such as speed recognition and on–line hand–writing recognition. However, no current posture recognition approach applies fuzzy theories to solve the problems of noise and variance.

Instantiated by the above facts, a novel posture recognition algorithm is proposed to overcome some serious limitations of the previous approaches by using fuzzy theories. This section explains the objectives, working principles and detail designs of the proposed system.

4.2.1 Objectives

As mentioned in the previous sections, there are still many limitations in the existing hand recognition approaches, which made them impractical to be integrated with virtual reality applications. The proposed fuzzy posture recognition system aims at solving both the problems of noise and efficiency at the same time. The major objectives of the proposed system is listed below:

- The system should be capable of dealing with imprecise posture data.
- The recognition process should be efficient and requires little amount of computation time.
- The mechanism of adding and removing posture records should be simple and efficient.
- The system should be flexible. It should work with any set of postures without any tuning or modification to the system.

4.2.2 System overview

The fuzzy posture recognition system is designed to handle 19 degrees of freedoms of the human hand, including the three joints for each finger and the four abduction angles. In the recognition process, the system compares the input posture against the posture records stored in the system using fuzzy operations. When comparing two postures, the system first computes the similarities, which are fuzzy variables, of corresponding parameters of the postures. The overall similarity is then obtained by combining the results, such as the degree of membership and certainty factor, of individual comparisons. This approach is very similar to the classic template matching, which handles the inaccuracies by widening the possible ranges of the parameters.

A classification technique is proposed to improve the efficiency of the system,

which is proportion to the number of posture comparisons taken during the recognition process. When a posture database is inputed into the system, the system classifies the postures according to a subset of the input parameters and groups similar postures into the same class. Only the possible class will be searched during recognition, and thus greatly reduces the number of postures comparisons. Fuzzy theories is also applied in classification.

There are several advantages of this design. First, fuzzy operations involve only very simple arithmetics, and are efficient in nature. Second, assuming that the postures records are uniformly distributed over all classes, only a small number of postures has to be considered during the recognition process. This feature is especially important when the size of the posture database is large. Third, as some of the input parameters are already considered at the classification, which is done at the start–up time, less parameters are required to be handled in run–time and makes the system more efficient.

The system consists of three major functional components: the *posture database*, the *classifier* and the *identifier*. The interaction of the three components is shown in Figure 4.2. The posture database stores the posture records and the classification results. The classifier and the identifier are responsible for the classification and recognition respectively. Detail descriptions on the design of these components are given later in this section.

4.2.3 Input parameters

As mentioned in Chapter 2, the human hand is too complex to model in full manner. It is impractical to handle all 23 degrees of freedom of the hand in the recognition. The fuzzy posture recognition system considers 19 degrees of freedom, including the three extension/flexion angles of each finger and the four abduction/adduction angles. Since the efficiency of the system is direct related to the number of parameters handled, the remaining less important degrees of

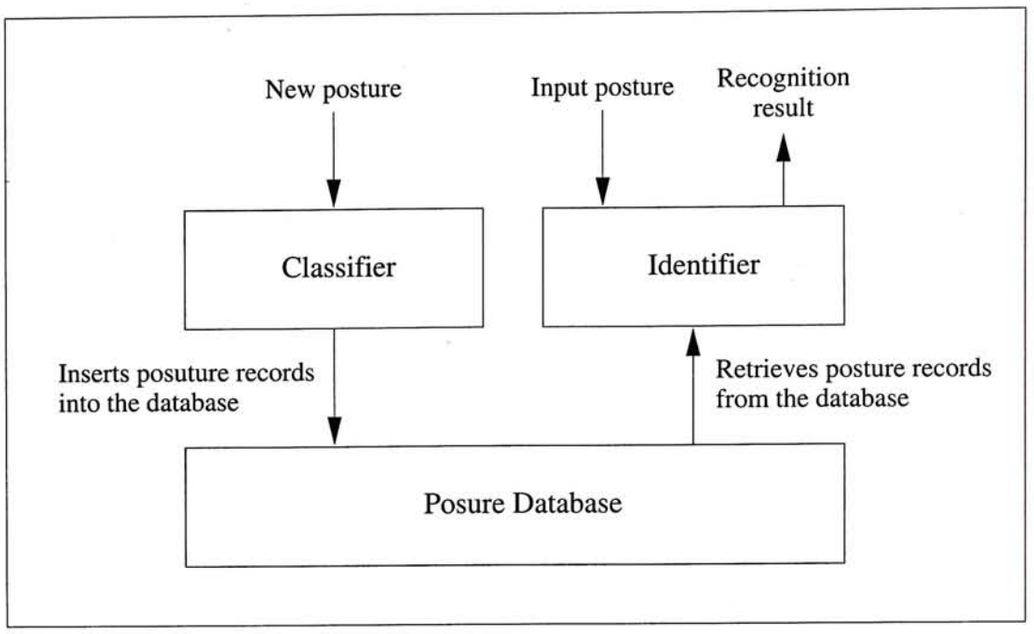


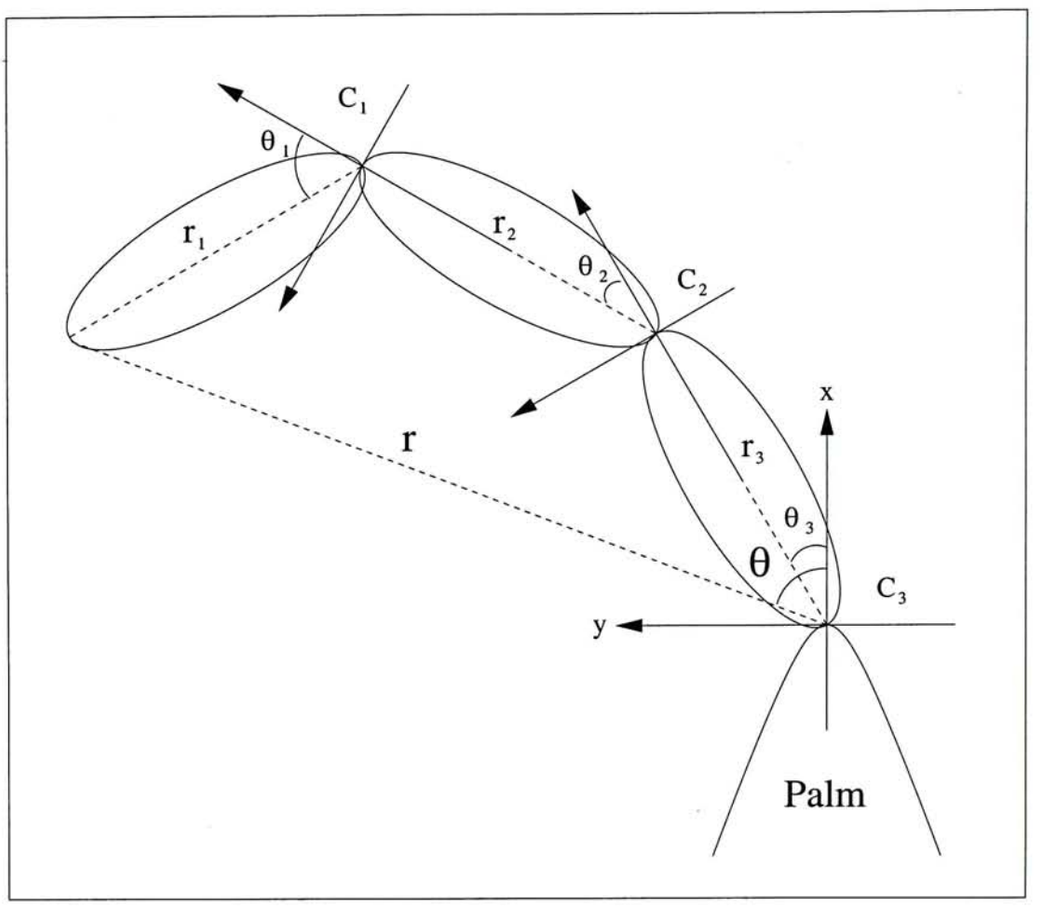
Figure 4.2. System Model

freedom are ignored.

Most popular gloves, such as DataGlove and CyberGlove, detect the extension/flexion angles of the finger joints. Most current recognition methods are based on the values of joint angles. However, it is believed that there are some other derivatives which are better than the joint angles, in terms of both storage and processing time. The tip positions of fingers are the alternative proposed in this approach.

Although each finger involves four degrees of freedom, the extension/flexion movement of the finger always restricted on the same 2D plane. The tip position can be clearly specified by polar coordinates (θ, r), with the origin located at the MCP joint, as shown in Figure 4.3. In the system, a posture is represented by the five tip positions and the four abduction angles. The finger tip position can be easily calculated from the joint angles by using simple 2D rotation and translation of coordinate systems.

Despite that there is a small overhead of calculating the finger tip position, this posture representation has two advantages. First, the number of variables for each finger is reduced from three to two. The overall complexity of the posture, as well



Let (x_1, y_1), (x_2, y_2), (x_3, y_3) denote the coordinates of the finger tip position with respective to the coordinate systems C_1, C_2, C_3 respectively.

$$
\begin{aligned}
\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} &= \begin{bmatrix} r_1 \cos\theta_1 \\ r_1 \sin\theta_1 \end{bmatrix} \\
\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} &= \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} x_1 + r_2 \\ y_1 \end{bmatrix} \\
&= \begin{bmatrix} (x_1 + r_2)\cos\theta_2 - y_1 \sin\theta_2 \\ (x_1 + r_2)\sin\theta_2 + y_1 \cos\theta_2 \end{bmatrix} \\
\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} &= \begin{bmatrix} (x_2 + r_3)\cos\theta_3 - y_2 \sin\theta_3 \\ (x_2 + r_3)\sin\theta_3 + y_2 \cos\theta_3 \end{bmatrix} \\
r &= \sqrt{{x_3}^2 + {y_3}^2} \\
\theta &= \tan^{-1}(y_3/x_3)
\end{aligned}
$$

Figure 4.3. Transformation from joint angles to finger tip position

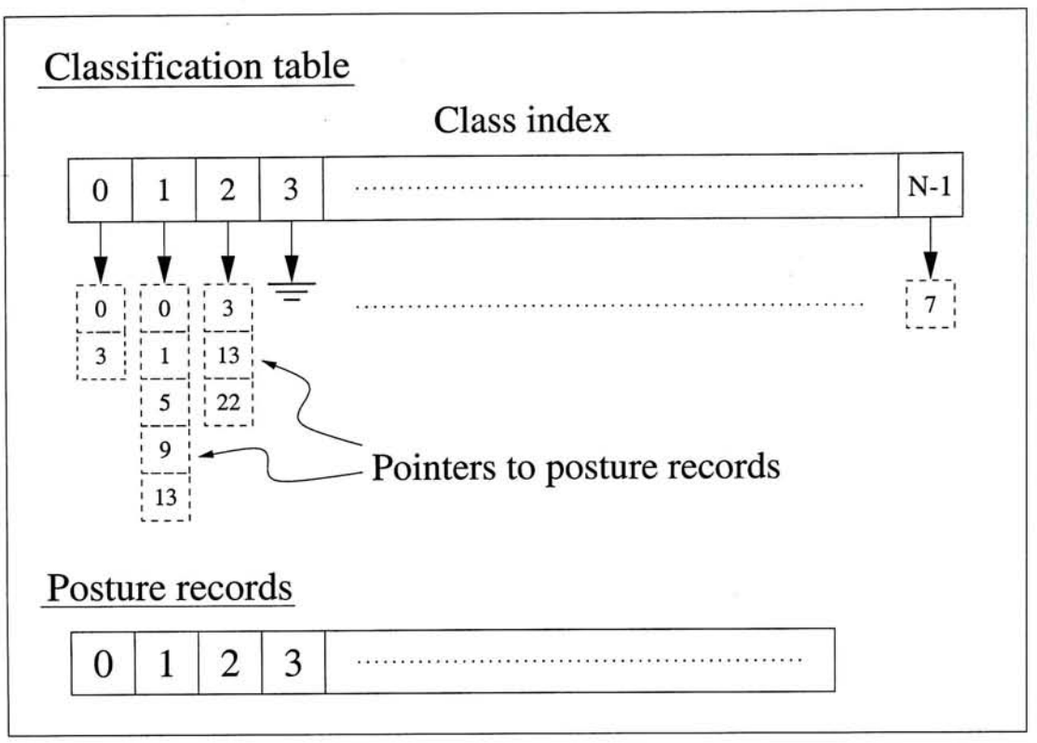


Figure 4.4. A graphical representation of the posture database. The numbers inside the dotted line boxes indicate their corresponding index to the posture record.

as the time taken for each recognition, can be saved. Second, finger tip position is a much clearer feature than joint angles, in terms of the ease of classification and recognition. Usually, the range of each joint angle is around 90 degrees, while the error due to noise or human variation may be up to 10 to 20 degrees. The error is very large compared to the range of movement. Recognition based on finger tip positions is more accurate, since even if each joint has an error of 10 degrees, the error of the finger position is relatively smaller.

4.2.4 Posture database

The logical structure of the *posture database* is shown in Figure 4.4. The posture database consists two components. The first one is an array of *posture records.* Each posture record stores the information required to describe the posture. The stored information are listed below:

- The *name* of the posture.
- A user–assigned *ID number* for easy identification of a posture.
- The *parameters* describing the posture, including the five finger tip positions and the four abduction angles.

The second component is the *classification table*, which is built for effective indexing of similar postures. The table consists many *classes*, where each class represents some characteristics possessed by some postures. For example, the class with index 0 may represents all postures with five fingers unflexed, while the class with index 20 may represents all postures with the last four fingers closed, etc. In other words, a class is actually a list of pointers to the postures with the same characteristics. As there are always some uncertainties in the classification, each pointer is associated with a certainty factor, which tells how certain that the posture belongs to the class. The size of the classification table is determined by the classification rules, described in the next section.

4.2.5 Classifier

The classifier is responsible for the building of the classification table inside the posture database. When a new posture is inserted into the system, the system first constructs its posture record and stores it in the posture database. The classifier is then invoked to classify the posture according to a subset of the posture's input parameters. Lastly, pointers to the posture is appended to the corresponding classes.

In the system, classification is based on eight parameters, including the polar distances, r, of the five finger tip positions and the three abduction angles between the last four fingers. The abduction angle between the thumb and the index finger is not handled here. Each of the above parameters are divided into several regions. The polar distance r is divided into 4 regions, indicated by 0 to 3, and the abduction angle is divided into 2, indicated by 0 and 1. The total possible of

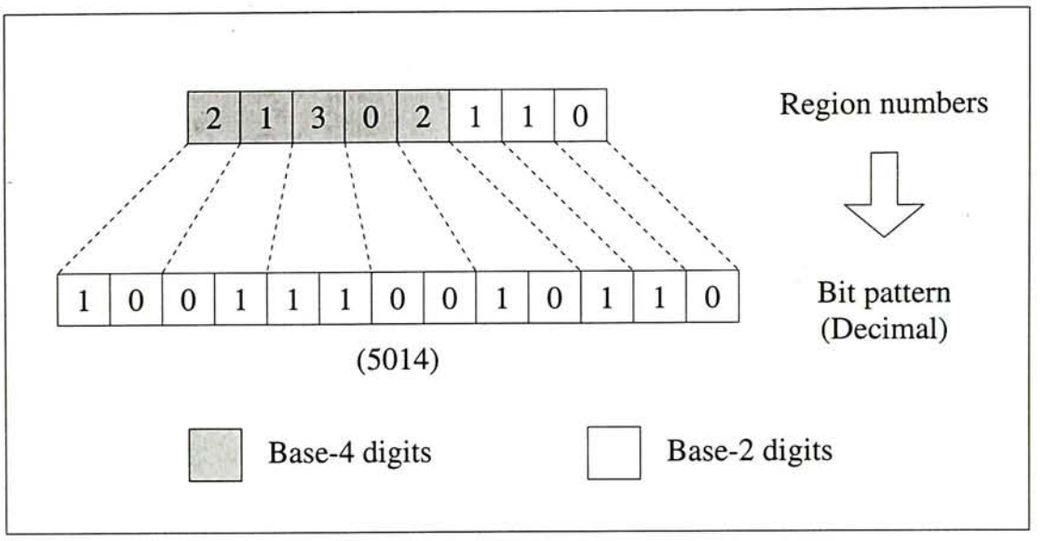


Figure 4.5. Computation of a class index

combinations of the above configuration equals $4^5 \times 2^3 = 8192$, which is taken to be the number of classes in the posture database.

The class index of a posture is computed from the regions where the posture's parameters lay. The computation is similar to transforming an N–based number, whose first 5 digits are base–4 and the last 3 is based–2, into decimal form. The process is illustrated in Figure 4.5. The number of regions are specially chosen to be multiples of 2, so that the computation involves only bit operations, such as bit–shifting and bitwise–or. These bit operations are more efficient than multiplication and addition in most computers.

Although we can assume that the distribution of the values of the joint angle is uniform, same assumption does not hold for the polar distance r. After some experiments, it is found that the distribution of r is bell–shaped with the peak shifted away from small values of r. If r is simply divided into 4 equal regions, the result of classification will be heavily clustered. In order to have a better classification result, the regions is divided according to the quartile ranges of the distribution of r.

Fuzzy boundaries are used in defining the regions. Figure 4.6 shows the boundaries of the four regions dividing r. Each boundary is a *trapezium membership*

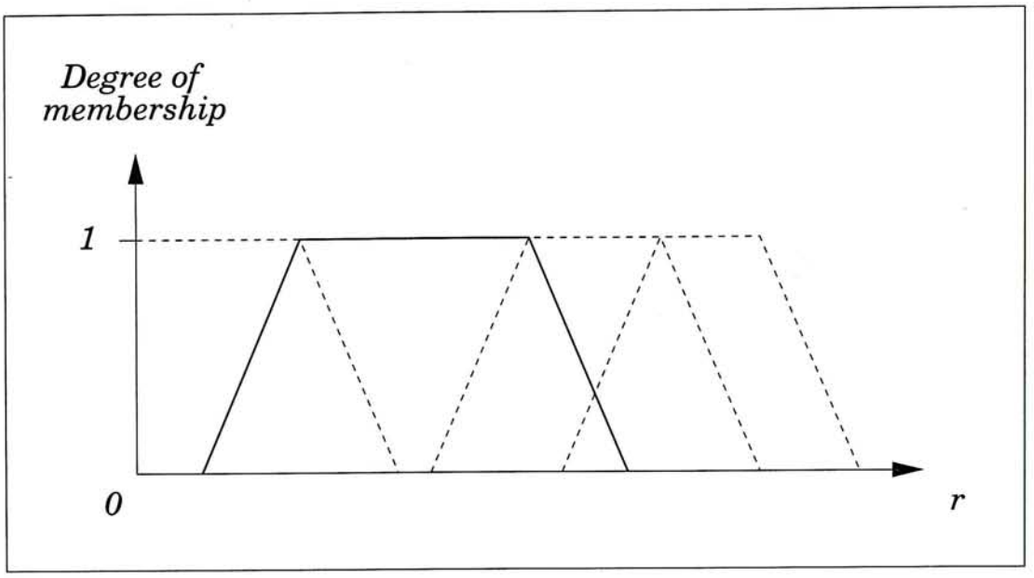


Figure 4.6. The membership functions defining the 4 fuzzy boundaries.

function. The trapezium member function is chosen because the computation involved is simple. As shown in the figure, a data point may fall in the overlapping area of more than one region. That point is considered to be the members of all those regions, with the respective certainty factors.

When a posture is passed to the classifier, the classifier finds out the set of classes which should contain the posture. If one of the eight parameters falls in the overlapping area of two regions, two pointers to the posture will be inserted into the database. If all eight parameters fall in overlapping areas, $2^8 = 256$ or more pointers will be inserted. A class index is computed for each pointer and the pointer is then stored in the corresponding class.

A certainty factor for each pointer will be computed and stored together with the pointer. This certainty factor describes how 'certain' a posture belongs to the class. Let cf_1 to cf_8 denote the certainty factors of the eight parameters. The overall certainty factor $cf_{1...8}$ is obtained by applying Equation 4.1 successively on cf_1 and cf_2, cf_{12} and cf_3, and so on.

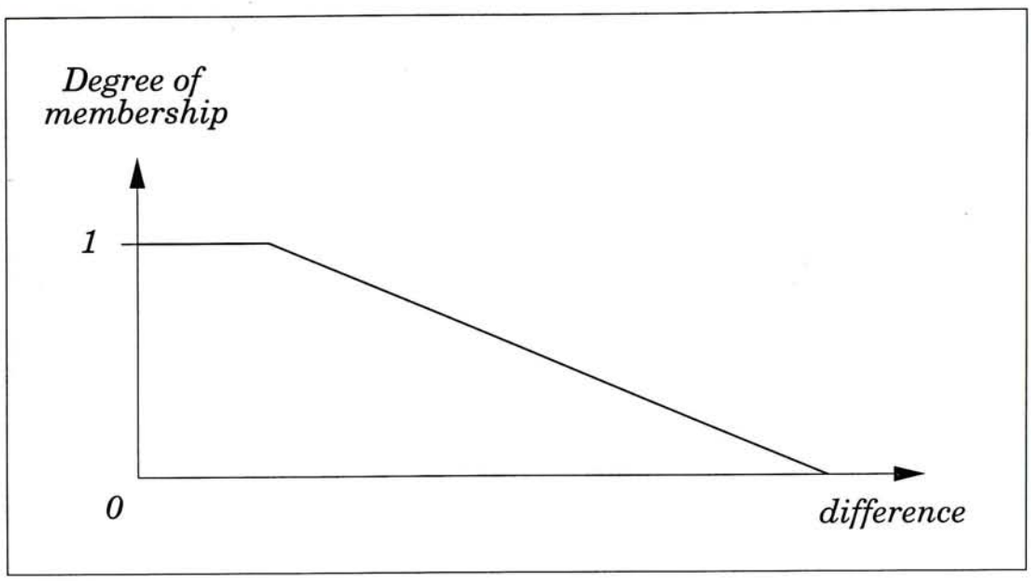


Figure 4.7. Membership function for similarity

4.2.6 Identifier

The identifier is responsible for posture recognition. When an input posture is passed to the identifier, the identifier first decides which class should be searched. A class index for the posture is computed using the same method described previously. The identifier then retrieves the posture records by the pointers stored in the class, and compares them with the input posture.

Since eight parameters of the posture are already considered during classification, the identifier only has to take care of remaining six parameters, i.e. the polar angles θ of five finger tips and the abduction angle between the thumb and the index finger. The similarities of the corresponding parameters, which are fuzzy terms, are measured by the *absolute difference* between the parameters. Figure 4.7 shows the membership function which maps the absolute difference to the degree of membership. The similarity drops with the absolute difference increases.

The six certainty factors obtained are then combined with the one calculated by the classifier, by applying Equation 4.1 successively. The result obtained is the overall certainty factor, which represent how the certainty of the recognition result. The posture record with the highest certainty factor in the class is then sorted

out by the identifier. If the certainty factor is above a pre–defined threshold, the posture is returned. Otherwise, the identification is considered to be failed.

Chapter 5

Performance Evaluation

This chapter evaluates the fuzzy posture recognition system. The behaviour of the system is examined through two sets of experiments. Section 5.1 describes the details of the experiments carried out and their results. Section 5.2 discusses the strengths and weaknesses of the proposed method and compares the system with existing approaches.

5.1 Experiments

The purpose of our experiments is to examine the two basic aspects of the fuzzy posture recognition: *accuracy* and *efficiency*. The following describes the methods and results of the experiments performed. All our experiments are performed on an SGI Octane workstation, which is equipped with an R10000 processor and 128M bytes physical memory, running under the operation system IRIX 6.4. The fuzzy posture recognition system and all testing programs in the experiments are written in C++ and compiled using GNU g++ version 2.7.2.

5.1.1 Accuracy analysis

The first experiment aims at investigating the behaviour of the fuzzy posture recognition system under different noise distributions. The experiment is performed as follows:

1. Load a sample database into the system
2. Select a posture record from the database
3. Add noise to the parameters of the posture record according to a distribution function
4. Recognize the noisy posture
5. Record the correctness of the result

The experiment is repeated using different databases and different noise distribution functions to increase the reliability of our analysis. As noise, like many real–life examples, is usually turned out to be a *normal distribution* [1] (or *Gaussian distribution*), our experiments will be focused on normally distributed noises. Noise under bounded uniform distribution is also considered. The results of the experiments are summarized in Table 5.1 and Table 5.2.

As observed in the tables, the major factor that affects the recognition rate is the distribution of the input noise. The recognition rate of the fuzzy posture recognition system varies as the input parameters are subjected to different noise distribution functions. Figure 5.1 and Figure 5.2 summaries the relationship between the recognition rate and the noise distribution function in graphical representation.

As shown in the Figure 5.1, the recognition rate only declines slightly when the standard deviation of the noise distribution in the input parameter increases from 5 degrees to 15 degrees. The recognition rate drops dramatically when the

[1] $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{\sigma^2}}$

Number of postures in the database	Accuracy					
	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$	$\sigma = 30$
10	100.00%	99.93%	94.56%	77.22%	56.83%	39.81%
20	100.00%	99.84%	94.93%	74.80%	54.60%	38.48%
30	100.00%	99.74%	93.93%	71.41%	53.29%	38.04%
40	100.00%	99.50%	93.16%	70.73%	52.35%	37.36%
50	100.00%	99.41%	92.42%	70.65%	51.48%	36.66%
60	100.00%	99.35%	91.91%	70.49%	51.06%	36.36%
70	99.99%	99.17%	90.61%	69.04%	49.71%	35.54%
80	99.99%	98.96%	89.83%	67.99%	48.97%	34.82%
90	99.99%	98.86%	89.63%	67.32%	48.07%	34.02%
100	99.99%	98.64%	88.16%	67.08%	47.24%	33.61%

Table 5.1. The behaviour of the fuzzy posture recognition system under normal distributions with different values of standard deviation σ ($\mu = 0$)

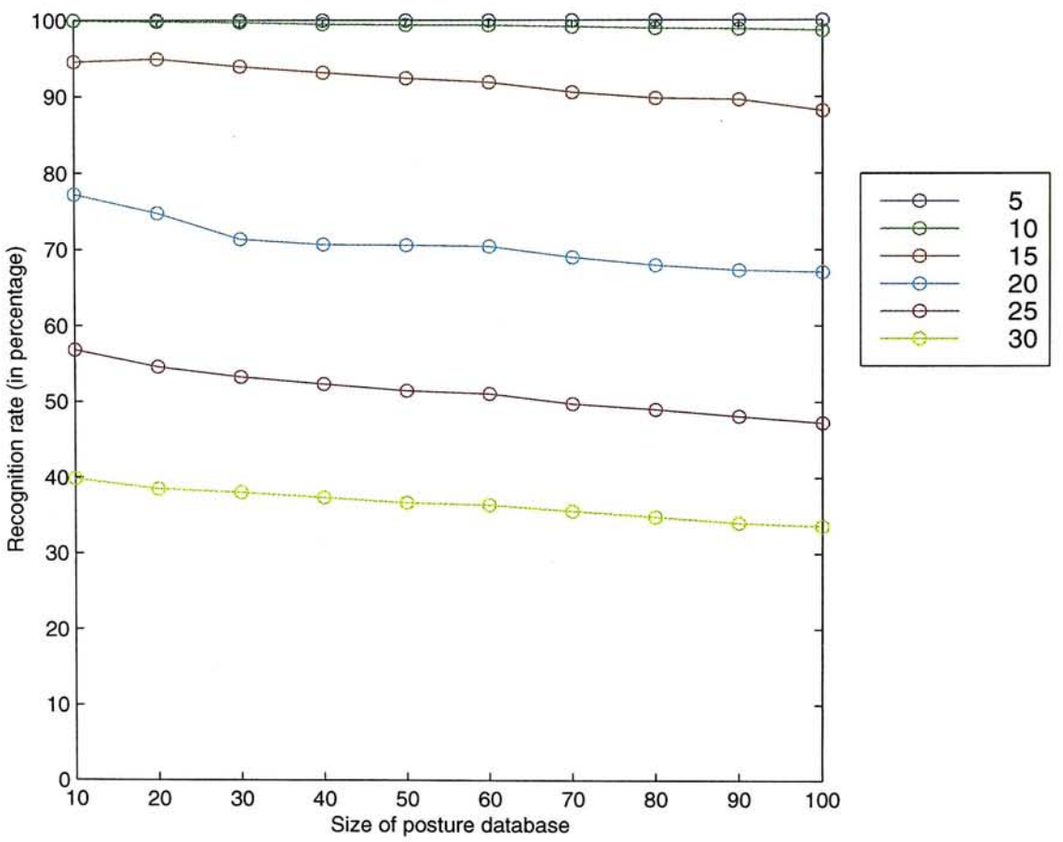


Figure 5.1. The relationship between recognition rate and the size of posture database for different noise distributions (normal distribution)

Number of postures in the database	Accuracy		
	$n = 10$	$n = 20$	$n = 30$
10	100.00%	99.82%	70.06%
20	100.00%	99.63%	67.96%
30	99.99%	99.26%	67.62%
40	99.97%	98.84%	67.10%
50	99.97%	98.56%	65.34%
60	99.96%	98.45%	63.50%
70	99.94%	98.00%	61.95%
80	99.92%	97.62%	60.95%
90	99.90%	97.55%	60.01%
100	99.87%	97.04%	58.89%

Table 5.2. The behaviour of the fuzzy posture recognition system under uniform distributions with different bounding values (n)

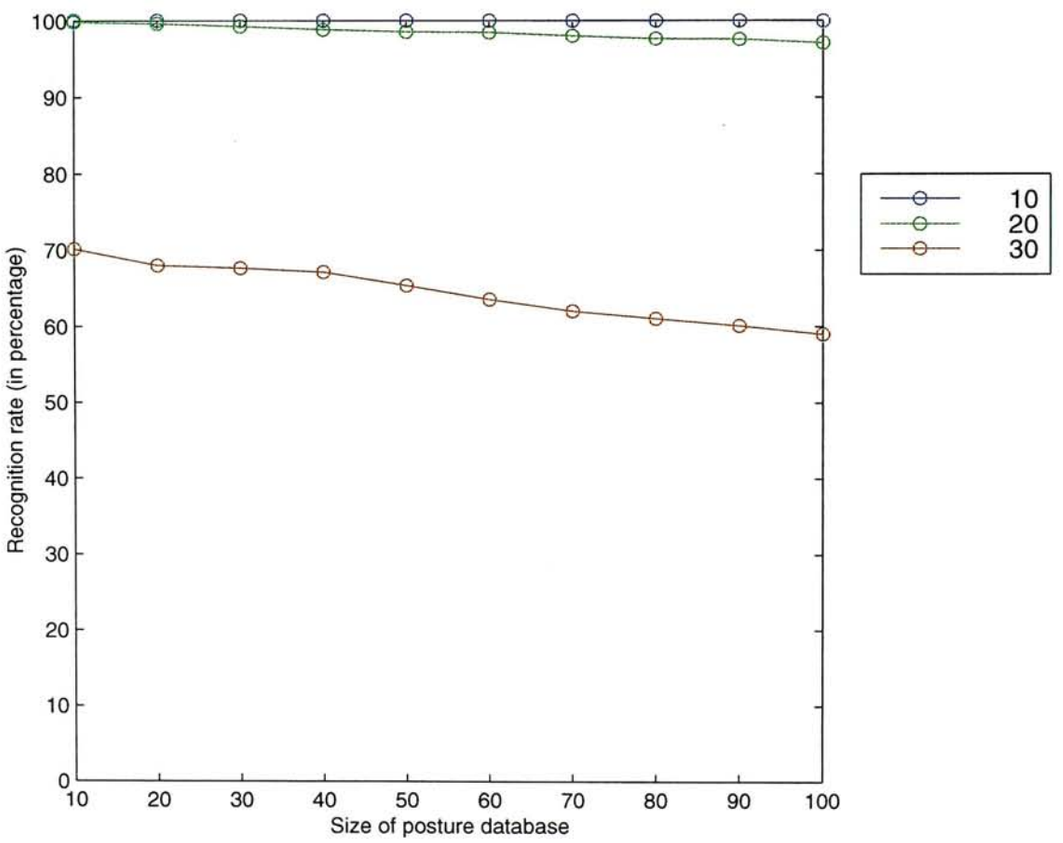


Figure 5.2. The relationship between recognition rate and the size of posture database for different noise distributions (uniform distribution)

standard deviation exceeds 15 degrees. This result is normal and can be easily explained. In practice, the error in the input parameters would not be too large. In such cases, the input posture should be regarded as some other distinct postures. Figure 5.2 shows a very similar trend.

From the experimental results, we conclude that the fuzzy posture recognition system is accurate. The use of fuzzy logic successfully deal with the noises in the input parameters. The correctness of the system is not affected by the size of the posture database.

5.1.2 Efficiency analysis

Another major concern of the recognition system is the *efficiency*. Our second experiment aims at investigating the efficiency of the recognition system when different posture databases are used. The relationship between the *CPU time taken* and the *size of the posture database* is particularly interested. The experiment is performed as follows:

1. Load a sample database into the system
2. Prepare an input posture by random
3. Recognize the posture prepared in the previous step
4. Repeat Step 2 and 3 for 1,000,000 trials
5. Record the time taken for the recognition process

As in the previous experiment, the efficiency test is repeated using different posture databases to increase the reliability. As template matching is the most efficient one among the current approaches, the experiment is also performed using a classic template matcher. Table 5.3 generalizes the result of the experiment.

As observed in Table 5.3, the proposed system is very efficient. The average CPU time taken for recognizing one posture takes only tens of micro–seconds. It

Number of postures in the database	CPU time (10^{-6}s)	
	Fuzzy Posture Recognition	Template Matching
10	13.65	27.99
20	14.53	55.65
30	15.50	83.51
40	16.35	110.97
50	16.71	138.90
60	17.46	166.82
70	18.83	194.81
80	19.75	222.94
90	20.58	249.99
100	21.35	279.62

Table 5.3. Comparison of the efficiency of the two posture recognition approaches

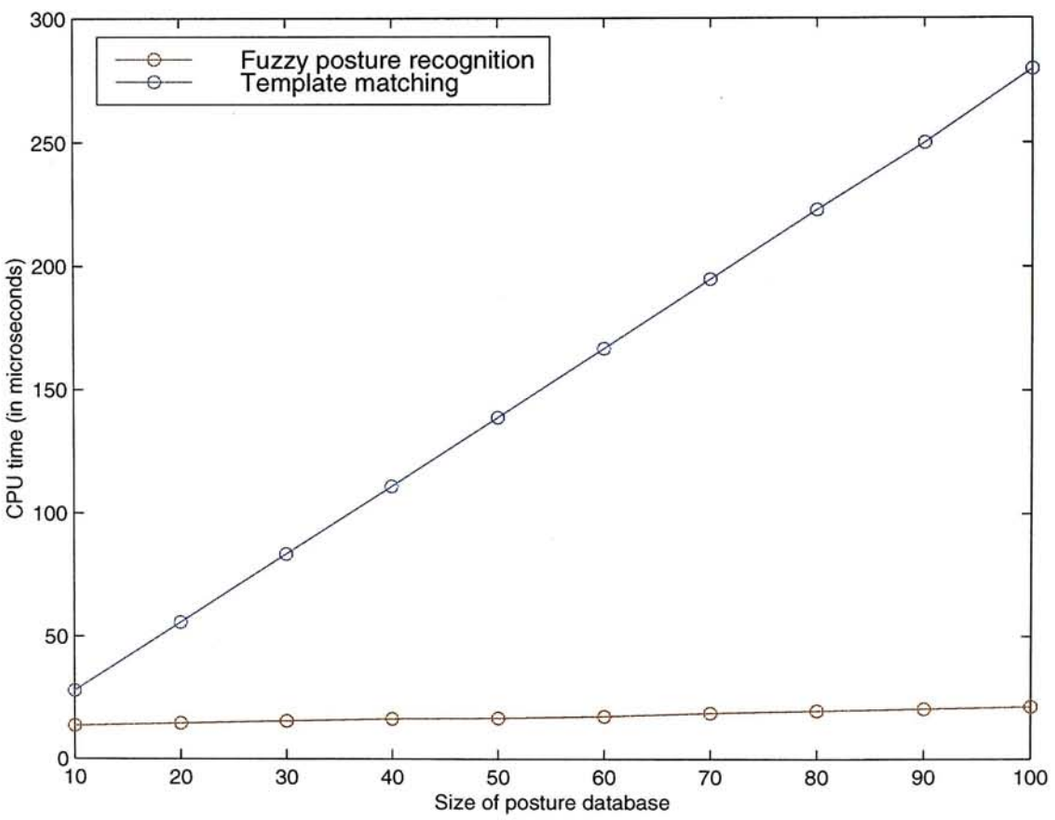


Figure 5.3. The relationship between efficiency and the size of posture database

is even more efficient than the template matching approach, especially when the size of the posture database is large.

Figure 5.3 shows the change of CPU time taken by both approaches as the size of posture database increases. The technique of classification shorten the time taken for recognition, as the number of operations needed for each recognition is greatly reduced. For small databases, our system is about 50% faster than the template matching approach. As the time taken for class indexing becomes less significant in case of larger databases, the speed–up can be further improved.

5.2 Discussion

Although we have successfully shown that the proposed recognition system is both accurate and efficient, the system is not a perfect solution to the problem. This section analyses the strengths and weaknesses of the proposed system and compares the system with other current methods. A brief summary on the proposed system will be given at the end of this section.

5.2.1 Strengths and weaknesses

Accuracy Accuracy is always the most important concern in recognition systems. In the proposed system, we tried to apply fuzzy concepts to deal with the inaccuracy factors in the problem of posture recognition. One source of error comes from the hand tracking device. According to Kessler's evaluation of CyberGlove [21], the standard deviation of the sensor data is around 10 degrees. As shown in the previous section, this level of noise can be accurately handled in the proposed system.

Efficiency As discussed in Chapter 2, the recognition system must be efficient. Like the classing template matching approach, which is well–known for its simpleness and efficiency, the calculation involved in our proposed system is very simple. In addition, the technique of classification greatly reduces the number of comparisons in the recognition procedure. These two facts explains why our system out–performs the classic template matching approach in our experiments. The recognition procedure can be finished within tens of micro–seconds even when the posture database is very large (100 postures). If a VR application needs 30 postures and runs at a frame rate of 30 frames/sec, the time interval between two frames is 1/30 second ($\sim$ 33333.33 micro–seconds). The proportion of the recognition process required is only $15.50/33333.33 \sim 1/2150$ of the total resources.

Flexibility Many existing hand recognition systems work for several pre–defined posture sets only. Some of them need fine–tuning, or re–training to run with other posture sets. Our system does not have this problem. It works properly with all testing posture databases (more than 100 distinct posture sets) in our experiments. This is an advantage for system developers, since our system allows them to design a specific posture command set for their application. A graphical user interface for creating and editing posture database will be described in Chapter 6.

Training In some recognition approaches, such as neural network and HMM, extensive training is needed before these system could work properly. A large number of labeled training examples are also required. These processes consume much human and computation resource. On the contrary, our system is much simpler. Our system needs only one example for each sample posture and no extensive training is required.

Gesture recognition The major limitation of the proposed recognition system is that it is not applicable to dynamic patterns, i.e. gestures. Current approaches that work for gestures include HMM, statistical classification and discontinuity

matching. This limitation affects the expression power of the use of whole–hand input. The proposed system cannot be applied in some specific applications, such as sign language interpretation.

5.2.2 Summary

The proposed fuzzy posture recognition system is evaluated in this chapter. In conclusion, we have shown that the proposed system possesses several advantages over some existing approaches. First, our system is accurate and efficient. The system successfully handles the problem of noise by using fuzzy logic. The technique of classification significantly reduced the time taken for recognition. It needs only a very small proportion of computation resources. Moreover, the system does not require fine–tuning or re–training for different databases. Real–time insertion and deletion of posture samples is possible. However, our current system cannot recognize dynamic pattern, which limits the expression power of the use whole–hand input.

Chapter 6

Posture Database Editor

In Chapter 4, a fuzzy posture recognition system is presented. This chapter introduces a graphical user interface designed for editing and validating the posture database used in the fuzzy posture recognition system. This posture database editor is called *"PostMan"*, which stands for "Posture Manager". The system architecture, user interface design and user functions of *PostMan* are described in the following sections.

6.1 System architecture

The posture database editor "PostMan" is designed to work on SGI workstations. A common object–oriented programming language, C++, is used to developed the system. This section describes the hardware configuration and the software tools used in developing the system.

6.1.1 Hardware configuration

The hardware components of the whole system can be divided into three sub–systems: the host computer, hand–input system and the 3D tracking system. The

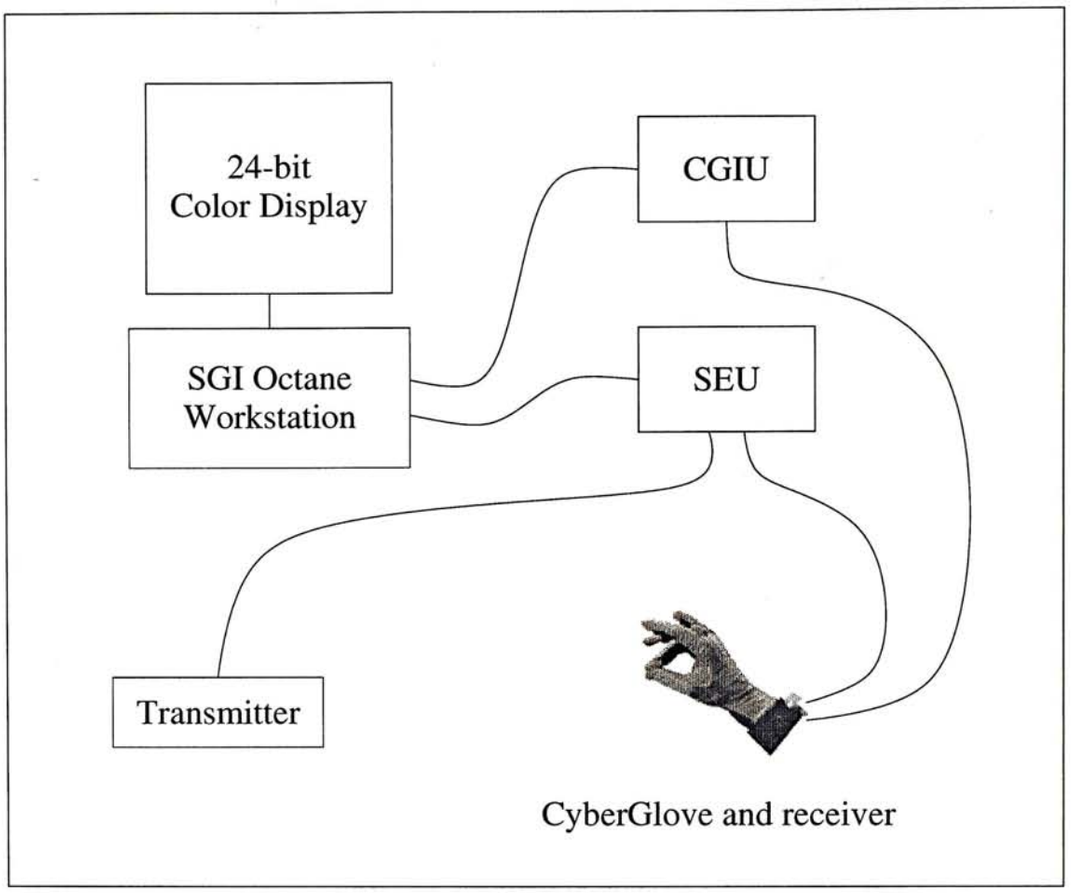


Figure 6.1. Hardware configuration of the system

configuration of these components are shown in Figure 6.1.

Host computer The host computer in the system is an *SGI Octane workstation*, equipped with one R10000 processor and 128M bytes physical memory, running under the operation system IRIX 6.4. The output device of the computer is an 24–bit color monitor, which produces high quality true–color display. The main jobs of the host computer includes collecting information from the other two sub–systems and running the application program.

Hand–input system The hand–input system has two components, including one *18–sensor CyberGlove* and one *CyberGlove Interface Unit (CGIU)*. The CyberGlove detects the sensor signals and sends them to the CGIU. The sensor signals are then amplified and digitized by the CGIU hardware. Upon request

from the host computer, these data are transmitted back to the host computer via an RS–232 serial port.

3D tracking system A *Polhemus 3SPACE FASTRAK* system is used to track the 3D position and orientation of the hand. The 3Space FASTRAK system includes a *System Electronics Unit (SEU)*, a *transmitter* and a *receiver*. The SEU drives the transmitter to generate electro–magnetic fields and collects the signals from the receiver, which is attached near the wrist of the CyberGlove. The SEU hardware computes position and orientation of the receiver, and sends these data back to the host computer via an RS–232 serial port.

6.1.2 Software tools

In addition to the fuzzy posture recognition system, a few software tools are used in the development of Postman. With the help of these tools, the system can be developed in a more efficient way. The following briefly describes the software tools used and their functions.

VirtualHand software library This software library [28] is shipped with the CyberGlove to for developing virtual–hand applications. The library contains many useful C functions, including high–level routines for the initialization of the glove, communication routines for the hand–input and the 3D tracking systems, graphics routines for drawing a virtual–hand model, and some common vector and matrix operations.

OpenGL graphics library In recent years, OpenGL [35] has become the standard graphics library and is available in many platforms. It offers a software interface for 2D and 3D graphics. The virtual hand model used in this system is drawn using OpenGL.

IRIS ViewKit toolkit The toolkit [19], developed by Silicon Graphics, is used to build graphical user interface more easily. It provides a collection of user interface components and thus the time taken for programming is greatly shortened. Most widgets in PostMan, such as windows, pull–down menus, dialog boxes, push buttons, etc. are built using this toolkit.

6.2 User interface

As mentioned at the beginning of the chapter, the main purposes of PostMan are editing and validating user–defined posture databases, which can be then saved for future use. PostMan has two operation modes: *Edit* mode and *Test* mode. User functions supported in *Edit* mode includes the insertion and deletion posture records, glove calibration and fine adjustment of the threshold value which defines the minimum certainty factor of identifying a posture. *Test* mode, on the other hand, is used to validate the current posture database. File operations are supported in both modes.

A snapshot of PostMan is captured and shown in Figure 6.2. Different components of the graphical user interface are also labeled in the figure. The GUI can be divided into several areas, including the *menu bar*, the *working frame*, the

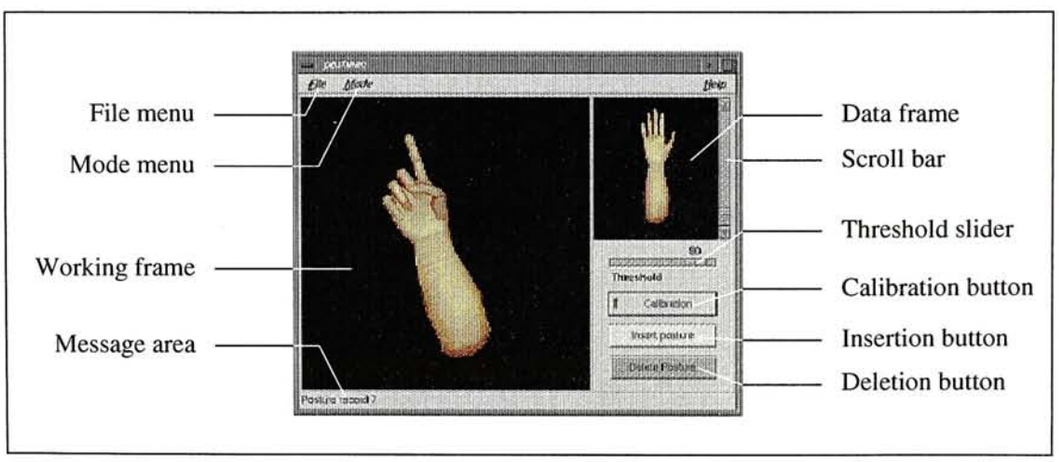
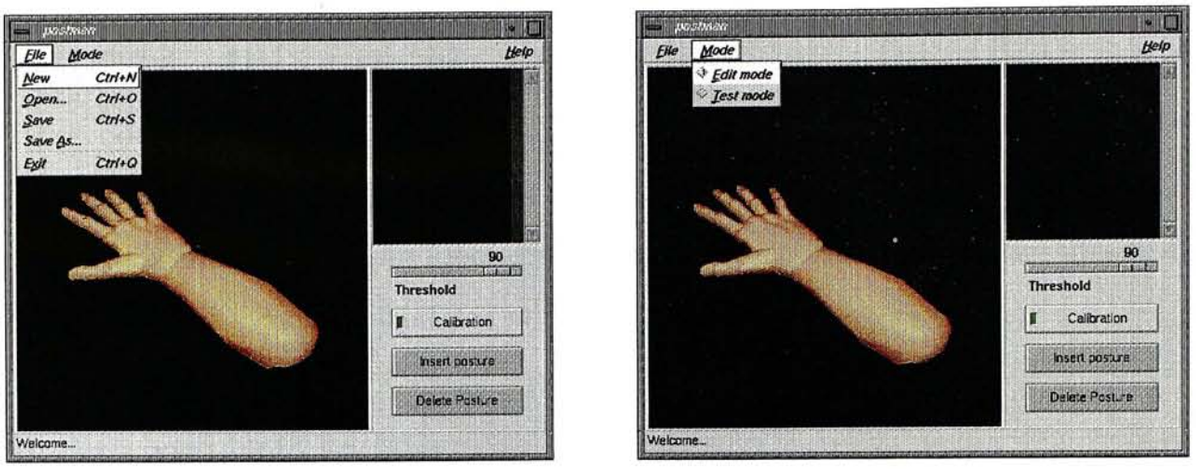


Figure 6.2. Posture database editor



(a) File menu

(b) Mode menu

Figure 6.3. The pull–down menus

data frame, the *control panel* and the *message area.* The functionalities of these components are described below.

6.2.1 Menu bar

The *menu bar* contains two pull–down menus: the *File* menu and the *Mode* menu, which are responsible for file operations and mode switching respectively. There are five menu items in the *File* menu (Figure 6.3(a)), including:

New	Create a new database
Open	Load an existing database
Save	Save the current database
Save as	Save the current database to another file
Exit	Exit PostMan

When the user choose *New*, *Open*, or *Exit* from the menu, PostMan first checks whether there is any unsaved changes. In such case, PostMan will ask the user to save the current database first.

The second menu in the menu bar is used for mode switching. When the editor starts, it operates in *Edit* mode by default. The user can switch between two modes by choosing the corresponding item in the *Mode* menu (Figure 6.3(b)). The check box before the menu items indicates the current operation mode. When PostMan is in *Test* mode, all functions other than the menu functions are disabled. The color of the control panel turns dim and the control panel is inactive.

6.2.2 Working frame and data frame

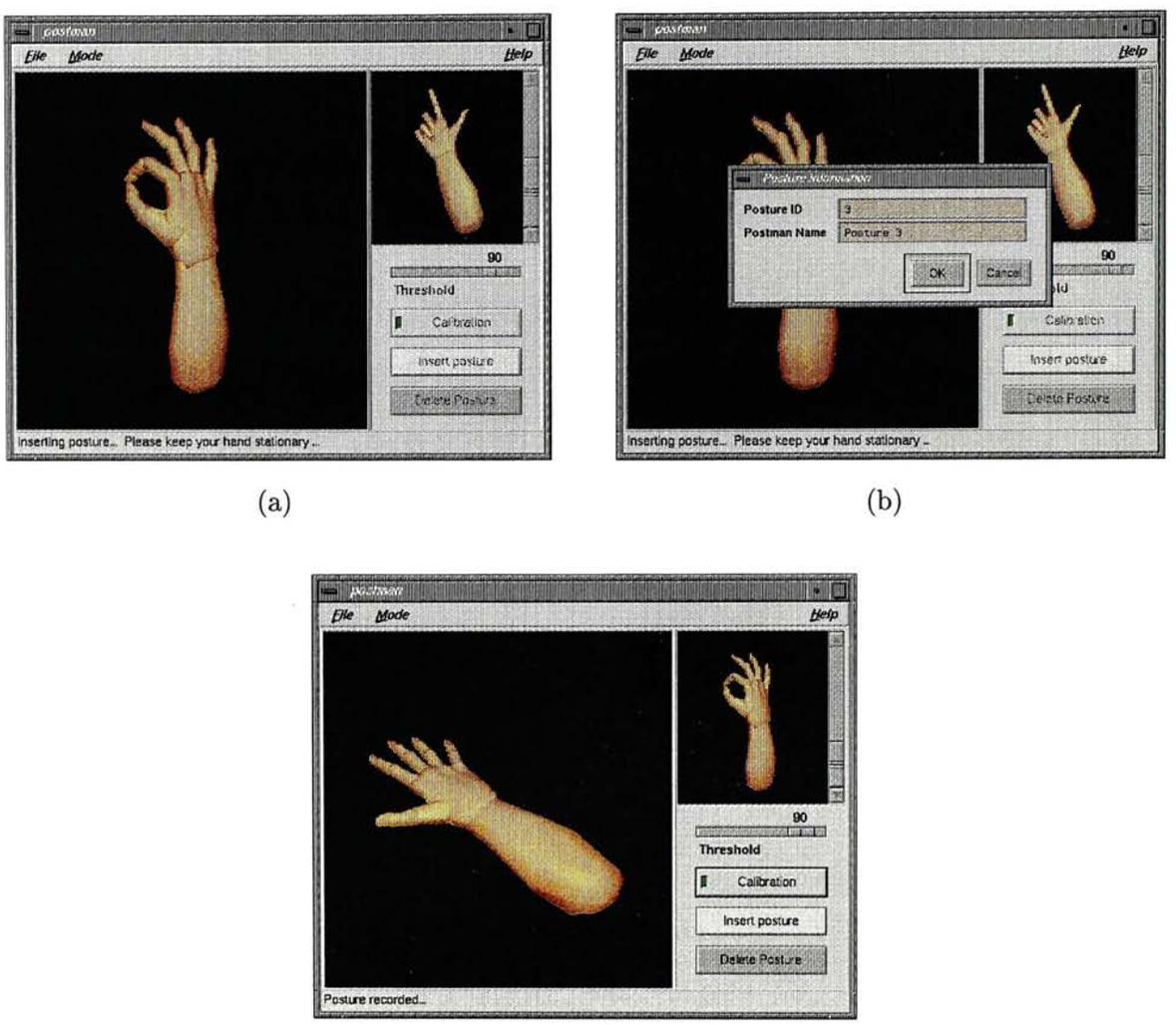
The *working frame* is an OpenGL drawing area. It is used to visualize the current hand pose detected by the CyberGlove. The virtual–hand is rendered using the function provided by the VirtualHand software library.

The *data frame* consists an OpenGL drawing area and a scroll bar. By moving the scroll bar besides the drawing area, different posture records in the database will be displayed. The name and the ID number of the posture will also be shown in the *message area* at the bottom PostMan.

6.2.3 Control panel

The *control panel* at the lower right corner consists a slider, a toggling button and two push buttons. The slider is used to adjust the threshold value, which defines the minimum certainty factor in posture identification (Section 4.2.6). The default value of this threshold is 0.90. The toggling button is use to invoke the dynamic calibration routine. When the dynamic calibration routine is activated, the "LED" on the button will be "on". The dynamic calibration routine will be deactivated automatically after about 10 seconds.

The two push buttons are used insert and delete posture records respectively. When the user presses the *Posture Insertion* button, PostMan will instruct the



(a) (b)

(c)

Figure 6.4. Procedures for inserting a posture record

user to hold the hand stationary for a few seconds (in the message area in Figure 6.4(a)). The control panel is now in inactive. PostMan will then record the user posture and ask for a name and an ID number the posture record. Figure 6.4(b) shows the dialog box for inputing these information. The insertion can be finished by pressing the "OK" button on the dialog box. The newly inserted posture record will be shown in the data frame (Figure 6.4(c)).

The procedures for deleting a posture record is simpler than insertion. First, choose the posture record to be deleted by using the scroll bar. The posture record can be then deleted by pressing the *Posture Deletion* button.

Chapter 7

An Application: 3D Virtual World Modeler

Virtual Reality Modeling Language (VRML) [18] is a popular virtual world modeling language. Users can use VRML to create 3D objects and compose interactive and immersive virtual worlds. It is also possible to distribute a virtual world over the World Wide Web and let others viewing it. Although VRML is such a powerful tool for 3D modeling, writing VRML code is a time consuming and tedious job. One has to keep a clear mind about coordinate systems, position and orientation of objects, coloring, lighting, transformations, and so on. It is very difficult for the users who are not familiar with 3D modeling, or simply have no graphics background, to build a 3D scene efficiently. A good modeling tool can certainly help. A number of 3D modeling applications, such as Alias Wavefront Animator™ and 3D Studio™, can export their data in VRML format. There are several advantages of using these modelers. First, One can build a 3D scene more efficiently. Second, any changes made to the scene can be visualized immediately. Unfortunately, these applications are all controlled by 2D input devices and do not provide direct 3D interaction for users. This chapter describes a 3D VRML modeling tools, called 3DVWM (3D Virtual World Modeler), which employs virtual–hand input.

7.1 System Design

The system is implemented using the hardware configuration shown in Figure 6.1. The software part of the 3DVWM is designed as a state machine. All state transitions are controlled by posture commands. Figure 7.1 shows the state transition diagram of the current system, and Figure 7.2 shows some common posture commands. Instead of using this default posture command set, users are allowed to define their own set using the posture database editor described in the Chapter 6

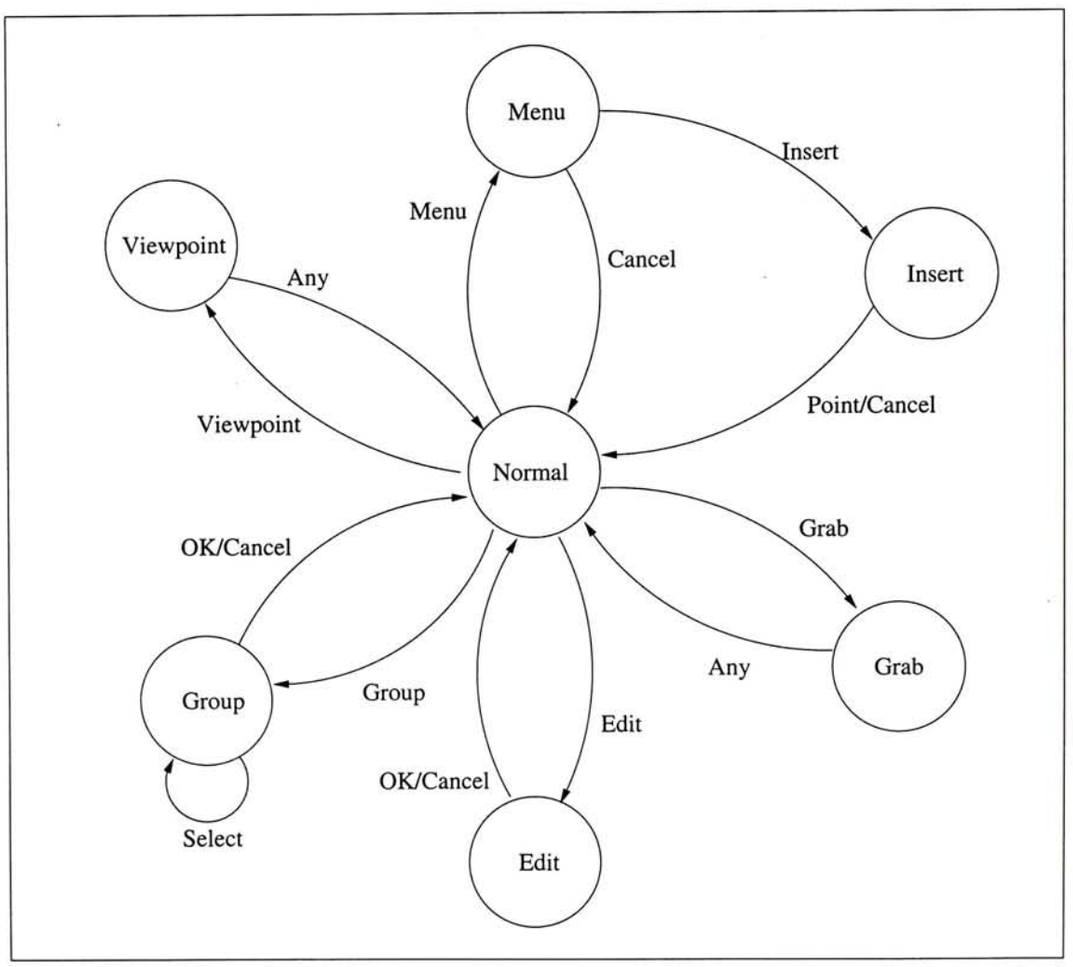
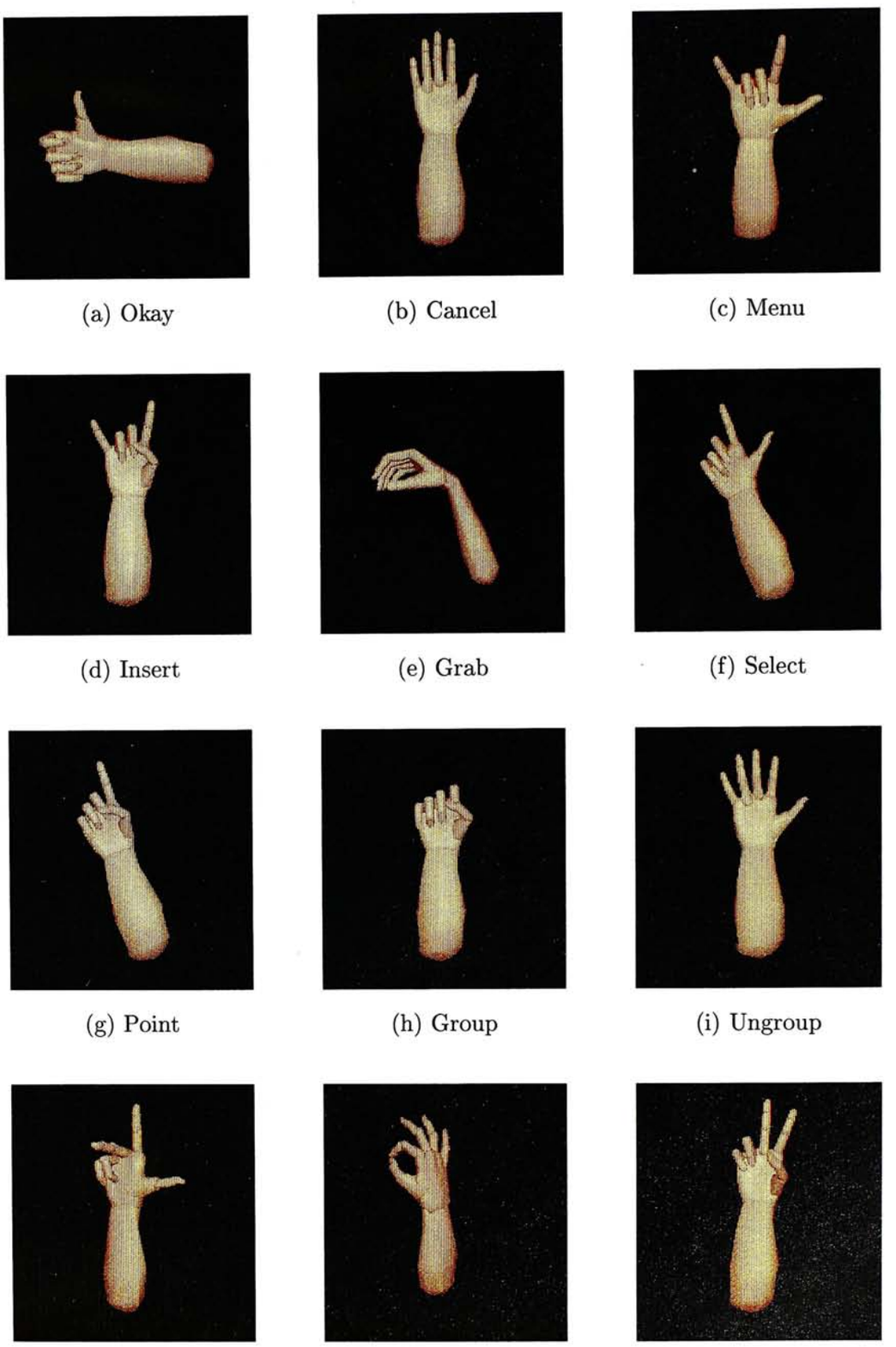


Figure 7.1. The state transition diagram of the 3D virtual world modeler.

Object–oriented approach is applied in our system design. Figure 7.3 shows the hierarchy of the virtual objects in the system. All virtual objects are divided into two classes. The first class includes the *visible* VRML nodes such as Box, Cone,



(a) Okay (b) Cancel (c) Menu

(d) Insert (e) Grab (f) Select

(g) Point (h) Group (i) Ungroup

(j) Viewpoint (k) Color (l) Delete

Figure 7.2. Some posture commands

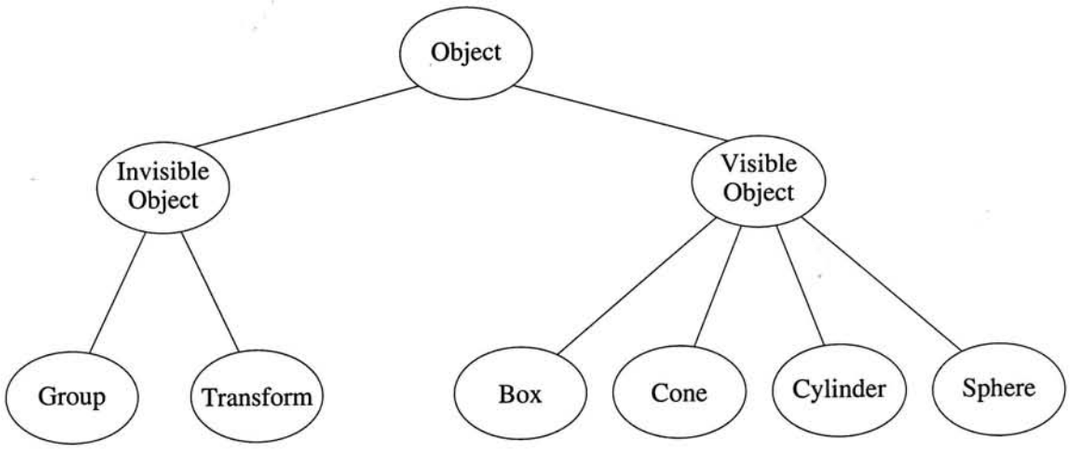


Figure 7.3. Virtual object hierarchy

Cylinder and Sphere. This kind of object hierarchy is very typical in geometric object representation. On the contrary, the second class is a set of *invisible* objects, which are all VRML nodes such as Transform, Group, and the lighting nodes.

7.2 Common operations

The following describes the common operations and the corresponding posture commands used in the modeler.

Confirming and canceling an action Like most systems, the most frequent operations in modeler are "okay" and "cancel" commands, shown in Figure 7.2(a) and Figure 7.2(b) respectively. In most cases, the modeler will return to the *Normal* state when an action is confirmed or cancelled.

Insertion Object insertion is done through a 3D ring menu, shown in Figure 7.4. When a "menu" command (Figure 7.2(c)) is received, a ring menu containing different VRML nodes, such as sphere, cone and lighting, is drawn on the screen. The user can turn the menu by twisting the hand and select a menu item by the posture command "insert" (Figure 7.2(d)). The selected object, drawn in

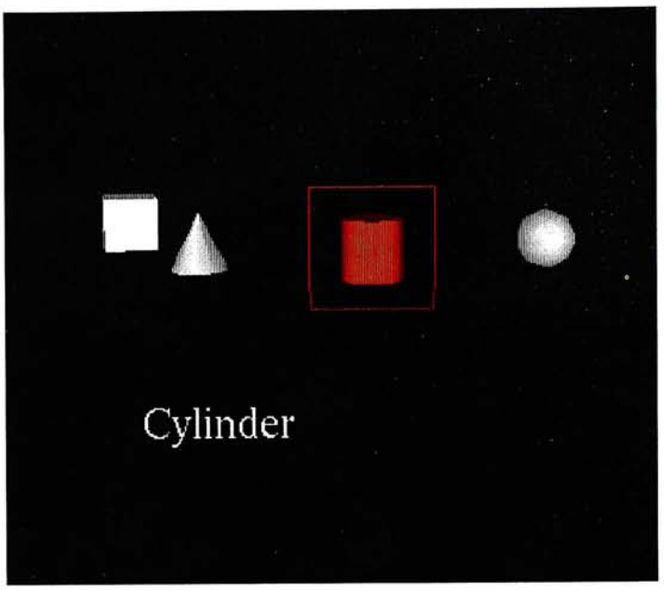


Figure 7.4. Ring menu

transparent color, will then moved with the hand until a "point" command (Figure 7.2(g)) is received. A cross forming by 3 straight lines along the 3 major axes is also drawn to help the user to place the object more easily and accurately.

Grabbing and moving When a "grab" command is detected, the modeler goes into the *Grab* state. In this state, the modeler performs collision detection between the hand and the virtual objects. If an object is found to be contacted with the hand, it will move with the hand until the modeler leaves the state. The modeler will go back to the *Normal* state when any other command is received. Bounding box algorithm is used to speedup the collision detection process.

Grouping and Ungrouping It is possible to group several primitive objects to form a compound object using the "group" operation. The grouped objects can be rotated or translated as if it is one single object. The modeler goes into the *Group* state upon receiving a "group" command (Figure 7.2(h)). To select the objects to be grouped, the "select" command (Figure 7.2(f)) is used. When the user holds this command, the modeler detects collision between the hand and the objects. The selected objects will be drawn in transparent color as an indication

of being selected. The "okay" and "cancel" commands are used to confirm or cancel the "group" operation respectively.

The reverse of this operation, "ungroup", is less complicated. First select the target object by the "select" command under the *Normal* state, and then issue an "ungroup" command (Figure 7.2(i)) to invoke the operation.

Deletion The deletion operation is invoked using the "delete" command (Figure 7.2(l)). The target objects must be selected before issuing the "delete" command. The object selection process is the same as the one in the "ungroup" operation.

Selection The user can select any objects in the 3D scene by using the "select" command. In order to differentiate the selected object from the others, the object will be drawn in transparent color. The object's center position, indicated by the intersection point of 3 axis–aligned lines, and its bounding box also be shown. More description about object selection will be given in the next section.

Duplication VRML 2.0 supports efficient object duplication by using "DEF" and "USE" [18]. Our modeler employs the same mechanism for duplicating objects. First, select an object and touch it with a "copy" command. Another instance of the selected object is then created. Locate the object by using the same method in objection insertion. If either one of the instance is modified, all other instances follow the change.

Changing viewpoint The "viewpoint" command (Figure 7.2(j)) is used to change the user's viewpoint. When the user holds this command, the modeler rotates or translates world coordinate frame according to the orientation of the hand.

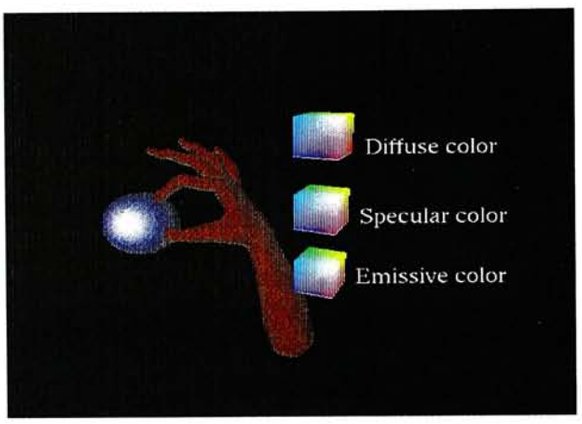
7.3 Virtual VRML Worlds

In the previous section, we mentioned some common operations with the corresponding posture commands. This section describes some operations, which are more complicated, with the help of a simple example (Figure 7.5).

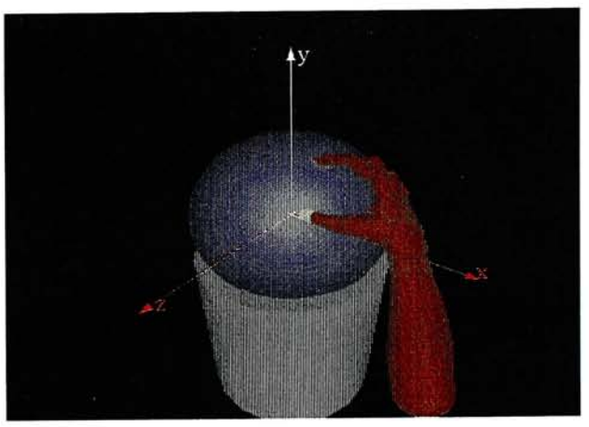
Shading an object The shading color of VRML objects are controlled by 6 components, including diffuse color, specular color, emissive color, ambient intensity, shininess and transparency. The first three components are RGB color notation and the others are real numbers between 0 and 1. By default, the modeler draws an object in gray, which is the default color of VRML objects.

The above shading models can be selected by additional recorded hand postures. The color components of each shading model can be further specified by a hand interaction with a color surrogate cube. The x, y and z coordinates of the color cube represent the three components of the RGB color model respectively. To set a parameter value, the user can move the hand to the proper color location of the cube. The tip position of the index finger is used as a reference point. The resulted appearance of the object updates accordingly to the finger's movement. We are developing the above hand functions which can match the standard VRML shading models, including color selection and texture mapping functions.

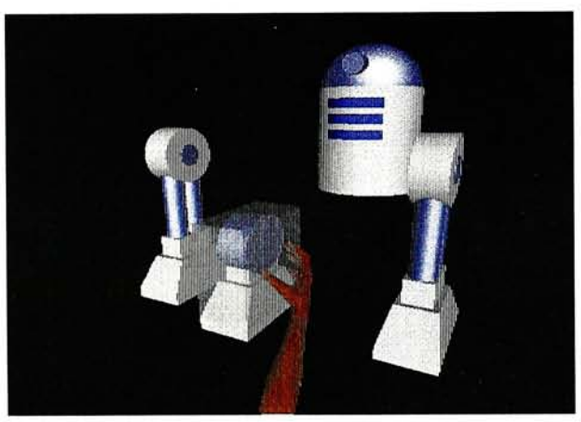
Constrained object movement In many cases, we need to move an object either along one of the major axes or rotate about one major axis in 1D or 2D, rather than a free 6D movement. In order to achieve the constrained operations easily, the modeler can be operated in two modes of movements: the *free* mode and the *constrained* mode. The *free* mode allows the free motion of the object, which follows the hand movement in space. The *constrained* mode imposes the movement to only 1D or 2D translation and rotation at a time. The modeler can switch the operation modes in the *Grab* state, using two registered hand postures.



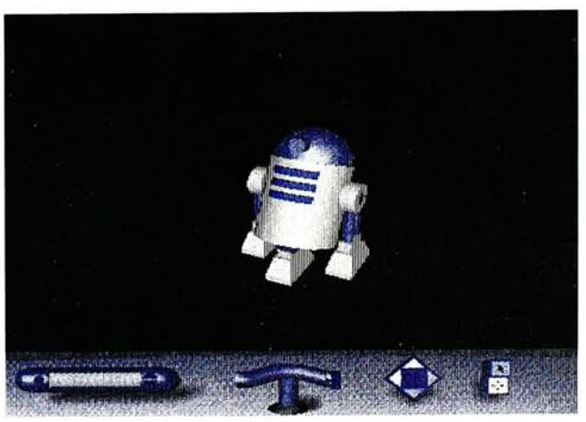
(a) Coloring an object



(b) Constrained translation



(c) Objects alignment



(d) Virtual robot model

Figure 7.5. Building a VRML world

Our current system specifies the constrained major axis by the times of selecting the same object. Initially, three axes are drawn to indicate the center position of a selected object. If the same object is selected again, the x–axis is selected as the constrained axis and highlighted. Selecting the object the third time toggles the constrained axis to the y–axis, fourth time the z–axis. The next following selection returns the operation mode to the normal state (free movement).

Currently, the modeler provides two types of constrained movement: *constrained translation* and *constrained rotation.* Under the constrained translation, the object can only be moved along the selected axis direction. Similarly, the object can only be rotated about the selected axis under the constrained rotation. These two movement functions can be identified by two recognized hand postures and executed in real–time by mapping the relative movement of hand–object coordinates. Our current system does not support the constrained translation and rotation at the same time or a constrained movement around a user–defined arbitrary direction, which will be the extended work of the current system.

Object alignment The object alignment is necessary to assemble two primitives or objects accurately. Our current system supports two types of object alignment: *alignment by axis* and *alignment by bounding box.*

In Figure 7.5(b), the main body of the robot is being assembled. The target is to combine the sphere and the cylinder concentrically. The assembly can be performed in the following way: first, select both objects with the y–axis toggled. Then issue an "alignment by axis" command, which aligns the two objects in the y–axis. Touch the sphere and move it along the axis under the constrained translation mode. The motion continues until the sphere is half–immersed into the cylinder as shown in the figure.

Figure 7.5(c) shows another alignment type. In the figure, all parts of the robot are built and ready for the final assembly. To make sure that all the three legs touch the ground, we can align them by the bottom face of each of the bounding

boxes. The similar idea can be applied to align the upper and lower legs. To do so, first toggle the y–axis of the objects to indicate the alignment direction. Then issue the "alignment by bounding box" command using the posture, which shows the bounding box of the object. Touch the upper leg and move it along the selected y–axis direction, until the bottom of the bound box is aligned with the top of the bounding box of the lower leg. Repeat the same hand interactions to align the other long leg.

The other parts of the robot can be assembled using the similar alignments. For instance, to assemble the small leg and the main body, we first align them by the y–axis. The grab the main body along the alignment. To adjust the orientation, we apply the constrained rotation about the x–axis on the main body. To align the two assembled long legs with the main body, we use "alignment by bounding box" and move the bounding box until they are aligned. Figure 7.5(d) shows the robot model created by the hand modeler viewed from the VRML interface of CosmoPlayer™.

Figure 7.6 shows the internal representation of the robot model in our modeler. This object tree representation is very similar to the VRML scene diagram representation. As a result, the translation from our model to VRML source code could be much easier and more efficient.

Our current system only supports the most basic primitives and operations for building VRML worlds. Its future work will be focused on extending the system to support more advanced VRML features, including complex geometry objects such as extrusion and elevation grid, texture mapping and editing, and various sensor nodes and interpolators, which make the virtual world more interactive and realistic.

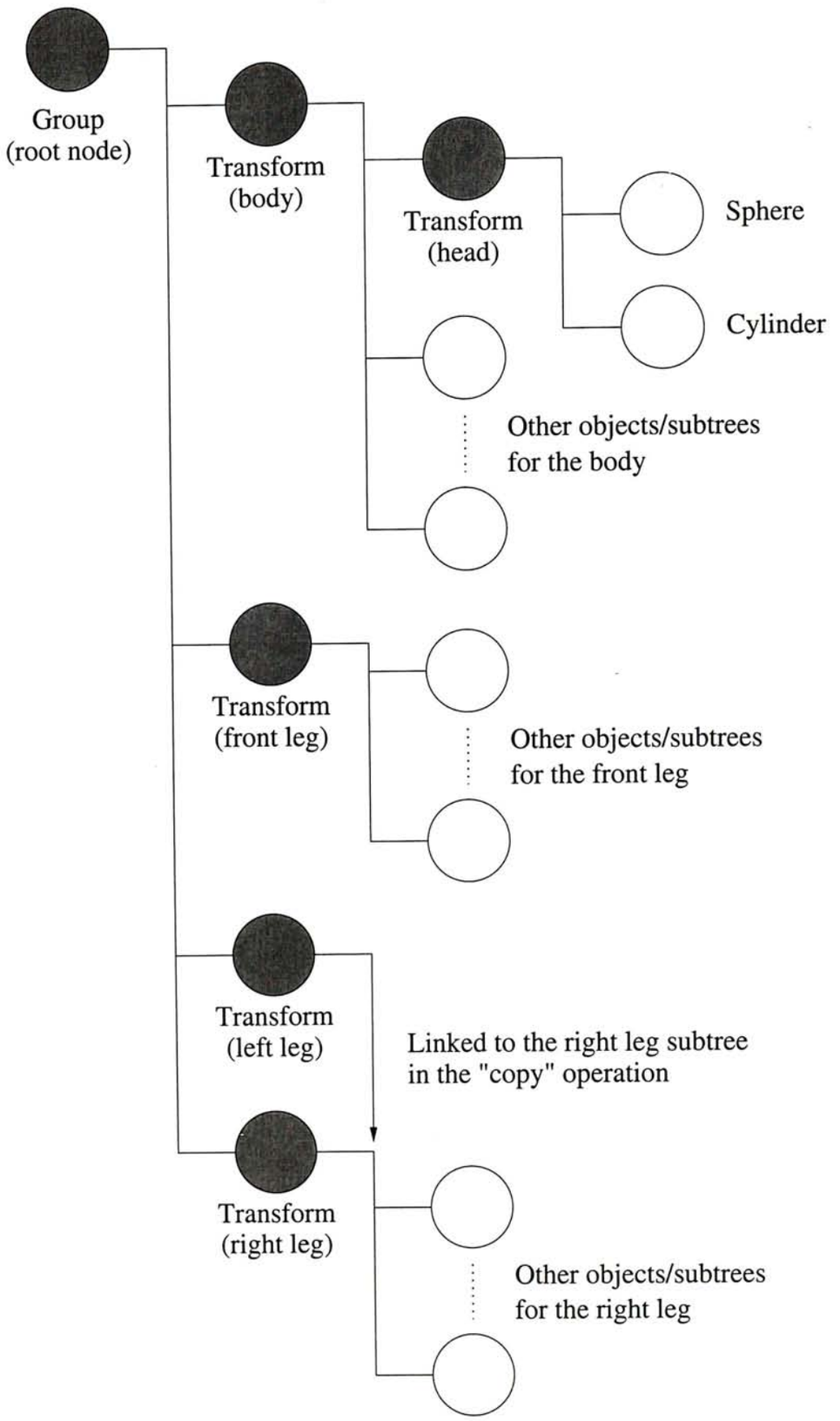


Figure 7.6. Internal representation of the robot model

Chapter 8

Conclusion

This dissertation is a research on *virtual–hand recognition*, which is defined as the utilization of recognition algorithms to identify hand postures and gestures in virtual environments. It aims at developing a virtual–hand recognition algorithm which is accurate, flexible, simple, and most importantly, efficient. In this dissertation, a wide range of background knowledges are studied. A fuzzy posture recognition system is proposed and implemented. A series of experiments are carried out to examine the behaviour of the system. Two applications are also developed to investigate the practical performance of the proposed system.

8.1 Summaries on previous work

Real–time interactivity and the *sense of immersive* are two key features of virtual reality (VR). The computer should be able to detect the user's action and generate the corresponding feedback to the user without any noticeable lag. With the appropriate equipments, virtual reality let users not only see, but also feel what the computer has done in the simulated 3D world. Unfortunately, traditional interaction technologies failed to fulfill both the above two features. Hand–input has become one of the major research directions in human–computer interaction

One enabling technology of virtual–hand recognition is the hand–tracking device, which has become a standard equipment of virtual reality. Two main streams of hand–tracking techniques are introduced, including glove–based tracking and image–based tracking. Various examples are also described. However, current hand–tracking techniques are far from ideal. Although glove–based devices are very popular, most gloves are uncomfortable. Some gloves are inaccurate and sensitive to noise. The mechanical parts sometimes inhibits the free movements of the hand. Periodic re–calibration is usually required to maintain the smoothness and stability of the glove. On the other hand, image–based tracking does not attach any mechanical device to the user, but it is slow and inaccurate. The problem of noise is even more serious then its glove–based counterparts. Moreover, it is impossible to implement tactile or force feedback on image–based device.

The main difficulty of hand–input comes from the complexity of the hand. The hand possesses up to 29 degrees of freedom in its full representation. It is inefficient and impractical to model the hand in full manner. No existing hand–tracking device measures all degrees of freedom of the hand. Some less important degrees of freedom are often ignored in hand–tracking devices and virtual–hand systems.

There are many problems in virtual–hand recognition, making the task very difficult. The first problem is the complexity of the human hand. Even some degrees of freedom can be safely ignored, about 20 input parameters are usually required to describe a posture, and a lot more for gesture. The second problem is the inaccuracy factors, such as human variation and the noise coming from hand–tracking devices. Gesture recognition faces extra difficulties. As gestures are time–varying signals, the problem space of gesture recognition involves both the space and time domains. Gesture segmentation leads to another problem. There is no obvious starting or ending of a gesture when the hand is moving continuously in the 3D space.

The most important concern in virtual–hand recognition is the efficiency. Virtual reality applications are usually very computation intensive. In order to achieve a smooth animation effect for visual feedback, it is necessary to maintain a high refresh rate and low latency. These two concerns impose a very high demand on the processing power of the computer. Therefore, practical virtual–hand recognition system must be simple and efficient, so that it would not affect the regular routines of the VR application.

Current posture and recognition approaches are carefully studied and analyzed. No existing recognition approaches is perfect for incorporating with real–time virtual reality applications. The template matching approach is simple, flexible and efficient, but its accuracy drops dramatically when more than 15 postures are stored in the system. Neural networks are high noise–tolerance and adaptive. Inaccurate or incomplete information can be handled effectively. However, the approach has a number of serious drawbacks. There is no systematic way to design the architecture, topology and the training algorithm of the network. The training phase is very long and a large number of labeled examples for each posture record is needed. The training process must be restarted after adding or removing postures records. Model–based analysis is a new technique and is very immature. Both the accuracy and efficiency of the approach are unsatisfactory.

The above approaches are more suitable for recognizing postures rather than gestures. Some other approaches are proposed to handle dynamic patterns for gesture recognition. The first one is the statistical classification approach. The major strength of the approach is that it handles dynamic patterns in a simple way. Gesture segmentation is automatic. However, it is necessary to select a feature set for each gesture database and there is no formal rule to guide this process. It does not exist a universal feature set that works for all gestures. The second approach is discontinuity matching. The gesture representation in this approach is relatively motion–oriented, which makes the recognition more effective. Moreover, the approach is robust to scaling of gesture pattern over

time, and is thus less sensible to noise. Unfortunately, the computation involved is too complex and is not suitable for real–time virtual reality systems.

Hidden Markov Models (HMM) is the most recent approach in the field. Hidden Markov Models are effective in recognizing both postures and gestures. The approach is very robust and quite efficient. Nevertheless, the approach has some serious limitations very similar to neural networks. There is no formal rule which guides the development of the system, including the model formulation, pattern representation, etc. The number of states in the HMM must be fine–tuned for each sample pattern and determined empirically. The HMM approach has a slightly edge out neural networks that the training time is shorter, and re–training is not necessary when adding or removing sample patterns. However, large number of labeled training examples is still required to build the statistic basis of the model.

8.2 Contributions

A novel fuzzy posture recognition system is proposed and implemented. The proposed system can be practically combined with VR applications. The system consists of three components: the posture database, the classifier and the identifier. The classifier roughly classifies the sample postures and stores them into different classes of the posture database. The identifier compares the input posture with the records in the class identified and finds the right match. The use of fuzzy logic successfully handles the noise problem in recognition. The system accurately recognizes noisy postures in our experiments. The recognition is very efficient, using only tens of micro-seconds. The classification technique significantly reduces the average search time of the identifier and thus the efficiency is improved. The system is also very flexible. It works with all sets of sample postures without any modification to the system. Moreover, the insertion of new postures is simple and can be done in real–time.

A posture database editor called *PostMan* (Posture Manager) is developed.

It is a graphical user interface designed for effective management of posture databases. The editor has two operation models: the *Edit* mode and the *Test* mode. In *Edit* mode, users can insert or delete postures, calibrate the glove and adjusting the threshold value which defines the minimum certainty of identifying a posture. It can also write the current database to disk, load and edit an existing database. *Test* mode allows users to test the current posture database.

A 3D modeling tool called 3DVWM (3D Virtual World Modeler) is also developed. The 3DVWM is a CAD–like modeling tool designed for creating VRML world. Instead of keyboard and mouse, a CyberGlove is used as the input and control device. The fuzzy posture recognition system is incorporated with the modeler to accept posture commands and manipulate the virtual objects directly.

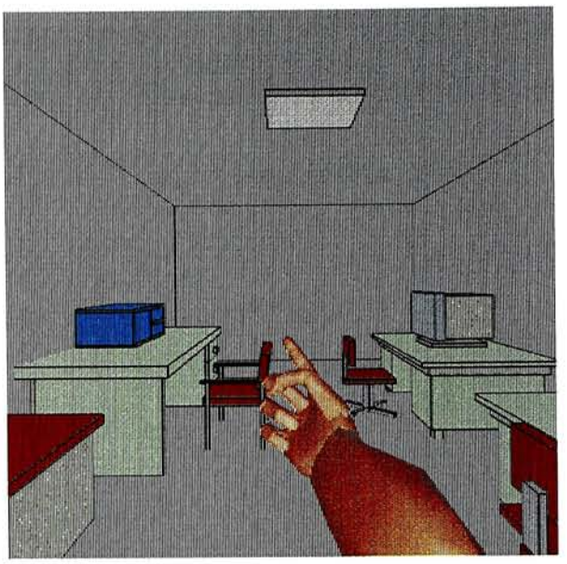
Chapter 9

Future Work

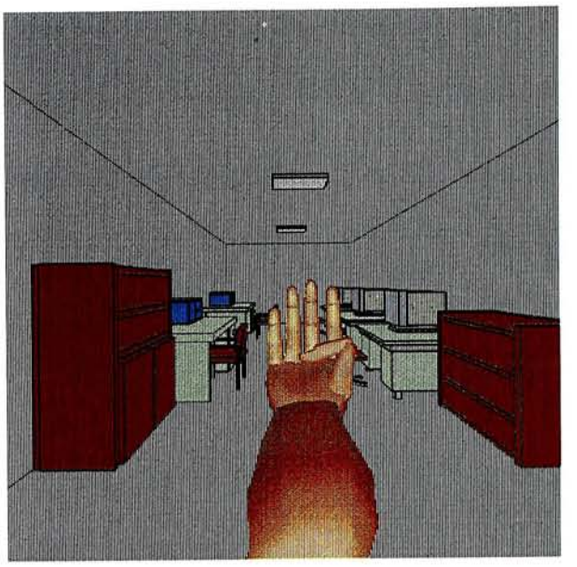
The main limitation of the virtual–hand recognition system proposed in this dissertation is that it cannot recognize dynamic pattern. In general, gestures are more intuitive and are more useful than postures. Therefore, one recommendation on the future direction of this research is gesture recognition. However, as gestures are much more complex than postures, more efficient algorithms are required. Another possible solution is to use existing recognition approaches and runs the recognition process on a separate computer. In fact, many VR systems are distributed over local networks to share the workload.

Another recommended direction is *two–handed input.* We used to work with both hands. Many actions could be simplified by using both hands. Some previous systems have already shown the possibility of using two–handed posture and gesture recognition systems. The two systems, TGSH (Two–handed Gesture environment SHell) [36] and CHINA [25], developed at the Oita University used a recurrent neural network to recognize 6 two–handed gestures.

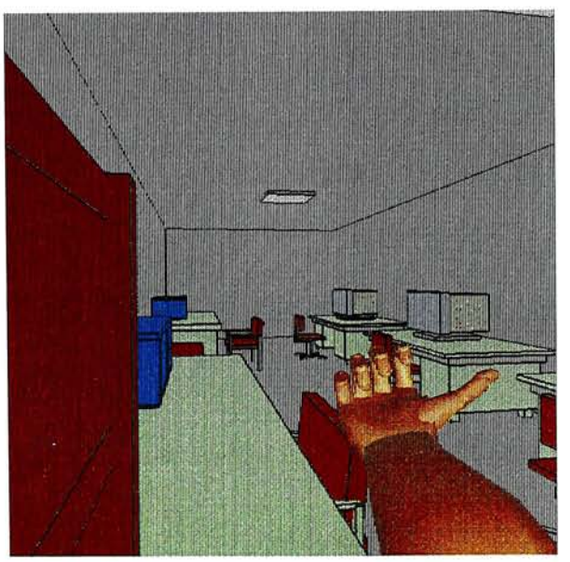
The virtual world modeler described in Chapter 7 has demonstrated one application area of hand posture recognition. Other potential application areas includes *virtual space navigation*, *interior design*, *scientific visualization*, etc. We have started the development an application which is designed for interior design



(a) Going forward



(b) Going backward



(c) Turning

Figure 9.1. Interior design and virtual world navigation

and navigation. In the current system, we can navigate in a virtual room using three postures, *forward*, *backward* and *turning*. Figure 9.1 demonstrates how these postures work. Both the *left turn* and the *right turn* command use the same posture. The turning direction is indicated by the direction of the thumb. The user action shown in Figure 9.1(c) is a right turn. Other functions for object manipulation, such as coloring and relocating of objects, etc. will be developed in the future.

Bibliography

[1] John A. Adam. Virtual Reality is for Real. *IEEE Spectrum: Virtual Reality Tools, Trends and Applications*, 30(10):22–29, October 1993.

[2] James C. Bezdek. Fuzzy Models — What Are They, and Why? *IEEE Transactions on Fuzzy Systems*, 1(1):1–5, February 1993.

[3] Eric A. Bier. Skitters and Jacks: Interactive 3D Positioning Tools. In *Proceedings on ACM Workshop on Interactive 3D Graphics*, pages 183–196, Chapel Hill, NC, October 1986. ACM.

[4] Eric A. Bier. Snap–Dragging in Three Dimensions. *Computer Graphics*, 24(2):193–204, March 1990.

[5] Mark Billinghurst, Sisinio Baldis, Lydia Matheson, and Mark Philips. 3D Palette: A Virtual Reality Content Creation Tool. In *Symposium on Virtual Reality Software and Technology*, pages 155–156. ACM, October 1997.

[6] Ronan Boulic, Serge Rezzonico, and Daniel Thalmann. Multi–Finger Manipulation of Virtual Objects. In *Symposium on Virtual Reality Software and Technology*, pages 67–74. ACM, July 1996.

[7] Grigore Burdea. Virtual Reality Systems and Applications. In *Electro'93 Internation Conference*, New Jersey, April 1993. Edison. Short Course.

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

[9] Martin Buss and Hideki Hashimoto. Dextrous Robot Hand Experiments. In *International Conference on Robotics and Automation*, pages 1680–1686. IEEE, 1995.

[10] Michael Clifton and Alex Pang. Cutting Planes and Beyond. *Computers & Graphics*, 21(5):563–575, 1997.

[11] Marc Ebner and Richard S. Wallace. A Direct–Drive Hand: Design, Modeling and Control. In *International Conference on Robotics and Automation*, pages 1668–1673. IEEE, 1995.

[12] S. Sidney Fels. Neural Networks for Computer–Human Interfaces: Glove–Talk II. In *Progress in Neural Information Processing Vol.2*, pages 1299–1304. ICONIP Hong Kong, Springer, September 1996.

[13] S. Sidney Fels and Geoffrey Hinton. GloveTalk: A Neural Network Interface between a Data–glove and a Speech Synthesizer. *IEEE Transactions on Neural Networks*, 4(4):2–8, July 1993.

[14] H. Fuchs. Superman Vision. In *Pixelation*, number 3, New Jersey, 1992.

[15] Carol M. Ginsberg and Delle Maxwell. Graphical Marionette. In *Proceedings on ACM Siggraph/Sigart Workshop on Motion*, pages 172–179. ACM, April 1983.

[16] Julian E. Gómez, Rikk Carey, Tony Fields, Andries van Dam, and Dan Venolia. Why is 3–D Interaction So Hard and What Can We Really Do About It? In *SIGGRAPH '94 Proceedings*, pages 492–493. SIGGRAPH, July 94.

[17] Gary J. Grimes. Digital Data Entry Glove Interface Device. Technical Report US Patent 4414537, Bell Telephone Laboratories, Novermber 1983.

[18] Jed Hartman and Josie Wernecke. *The VRML 2.0 Handbook: Building Moving Worlds on the Web.* Addison–Wesley Developers Press, August 1996.

[19] Ken Jones and Douglas B. O'Morain. *IRIS ViewKit™ Programmer's Guide.* Silicon Graphics, Inc., 1994.

[20] Ken–ichi Kameyama. Virtual Clay Modeling System. In *Symposium on Virtual Reality Software and Technology*, pages 197–200. ACM, October 1997.

[21] G. Drew Kesser, Larry F. Hodges, and Neff Walker. Evaluation of the CyberGlove Whole–hand Input Device. *ACM Transactions on Computer–Human Interaction*, 2(4):263–283, December 1995.

[22] Kiyoshi Kiyokawa, Haruo Takemura, Yoshiaki Katayama, Hidehiko Iwasa, and Naokazu Yokoya. VLEGO: A Simple Two–Handed Modeling Environment Based on Toy Blocks. In *Symposium on Virtual Reality Software and Technology*, pages 27–34. ACM, July 1996.

[23] George J. Klir and Tina A. Folger. *Fuzzy Sets, Uncertainty, and Information.* Prentice Hall, London, 1992.

[24] Byong K. Ko and Hyun S. Yang. Finger Mouse and Gesture Recognition System as a New Human Computer Interface. *Computers & Graphics*, 21(5):555–561, 1997.

[25] Kazuyoshi Korida, Hiroaki Nishino, and Kouichi Utsumiya. An Interactive 3D for a Virtual Ceramic Art Work Enivronment. In *Proceedings of 1997 International Conference on Virtual Systems and MultiMedia*, pages 227–234. International Society on Virtual Systems and MultiMedia, IEEE Computer Society, September 1997.

[26] James Kramer. *CyberGlove User's Manual.* Virtual Technologies, Inc., October 1995.

[27] James Kramer and L. Leifer. The Talking Glove: An Expressive and Receptive 'Verbal' Communication Aid for the Deaf, Deaf–Blind, and Non–vocal. Technical report, Stanford University, 1985.

[28] James Kramer, Mark Yim, and Larry Edwards. *VirtualHand Software Library Reference Manual.* Virtual Technologies, Inc., July 1995.

[29] K.S.Leung and W.Lam. Fuzzy Concepts in Expert Systems. *IEEE Transactions on Computers*, 21(9):43–56, September 1988.

[30] Jintae Lee and Tosiyasu L. Kunii. Model–Based Analysis of Hand Posture. *IEEE Transactions on Computer Graphics and Applications*, pages 77–86, September 1995.

[31] Rung Huei Liang and Ming Ouhyoung. A Sign Language Recognition System using Hidden Markov Model and Context Sensitive Search. In *Symposium on Virtual Reality Software and Technology*, pages 59–66. ACM, July 1996.

[32] Ashutosh Malaviya and Reinhard Klette. A Fuzzy Syntactic Method for On–line Handwriting Recognition. In *Advances in Structural and Syntactical Pattern Recognition*, pages 381–392. SSPR, 1996.

[33] Byung–Woo Min, Ho–Sub Yoon, Jung Soh, Yun–Mo Yang, and Toshiaki Ejima. Hand Gesture Recognition Using Hidden Markov Models. In *Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4232–4235. Systems, Man, and Cybernetics Society, IEEE, October 1997.

[34] Yanghee Nam and KwangYun Wohn. Recognition of Space–Time Hand–Gestures using Hidden Markov Model. In *Symposium on Virtual Reality Software and Technology*, pages 51–58. ACM, July 1996.

[35] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide.* Addison-Wesley Publishing Company, 1993.

[36] Hiroaki Nishino, Kouichi Utsumyia, Daisuke Kuraoka, Kenji Yoshioka, and Kazuyoshi Korida. Interactive Two–handed Gesture Interface in 3D Virtual Environments. In *Symposium on Virtual Reality Software and Technology*, pages 1–8. ACM, October 1997.

[37] Rochelle O'Hagan and Alexander Zelinsky. Finger Track — A Robust and

Real–time Gesture Interface. In *Proceedings of the 10th Australian Joint Conference on Artifical Intelligence (AI'97)*, pages 475–484. Australian Computer Society's National Committee on Artificial Intelligence and Expert Systems, Springer, November 1997.

[38] Tim Poston and Luis Serra. The Virtual Workbench: Dextrous VR. In *Proceedings of VRST 94 — Virtual Reality Software and Technology*, pages 111–122. World Scientific, August 1994.

[39] Dean Rubine. *The Automatic Recognition of Gestures.* PhD thesis, Carnegie Mellon University, December 1991.

[40] Luis Serra, Tim Poston, Wieslaw Nowinski, Chua Beng Choon, Ng Hern, and Prem Pillay. The Brain Bench Planner and Trainer for Minimal Access Surgery. In *Symposium on Virtual Reality Software and Technology*, pages 191–192. ACM, July 1996.

[41] Edward H. Shortliffe. *Computer–Based Medical Consulation: MYCIN.* Elsevier North–Holland, New York, 1976.

[42] David J. Sturman. *Whole–hand Input.* PhD thesis, Massachusetts Institute of Technology, December 1991.

[43] David J. Sturman and David Zeltzer. A Survey of Glove–Based Input. *IEEE transaction on Computer Graphics and Application*, 14(1):30–39, January 1994.

[44] T. Takahashi and F. Kishino. Hand Gesture Coding Based on Experiments Using a Hand Gesture Interface Devi. *SIGCHI Bulletin*, 23(2):67–74, April 1991.

[45] Tamio Tanikawa, Tatuo Arai, and Pasi Ojala. Two–Finger Micro Hand. In *International Conference on Robotics and Automation*, pages 1674–1679. IEEE, 1995.

[46] Elton K. H. Tsang and Hanqiu Sun. An Efficient Posture Recognition Method Using Fuzzy Logic. *Journal of Virtual Reality: Research, Development and Applications*, 3:1–8, 1998.

[47] Collin Wang and David J. Cannon. Virtual–Reality–Based Point–and–Direct Roboticc Ispection in Manufacturing. *IEEE Transactions on Robotics and Automation*, 12(4):516–530, August 1996.

[48] Richard Watson. A Survey of Gesture Recogniture Techniques. Technical Report TCD–CS–93–11, Department of Computer Science, Trinity College, Dublin2, July 1993.

[49] Richard Watson and Paul O'Neill. A Flexible Gesture Interface. In *Proceeding of Graphics Interface '95*, pages 231–238, 1995.

[50] Richard Watson and Paul O'Neill. Gesture Recognition for Manipulation in Artifical Realities. In *Proceedings of the 6th International Conference on Human–Computer Interaction*, July 1995.

[51] Lotfi A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–352, 1965.

[52] Lotfi A. Zadeh and Janusz Kacprzyk. *Fuzzy Logic for the Management of Uncertainty*. John Wiley & Sons, Inc., New York, 1992.

[53] Robert C. Zeleznik, Kenneth P. Herndom, Daniel C. Robbins, Nate Huang, Tom Meyer, Noah Parker, and John F. Hughes. An Interactive 3D Toolkit for Constructing 3D Widgets. In *Computer Graphics Proceedings, Annual Conference Series*, pages 81–84. ACM, 1993.

[54] Thomas G. Zimmerman and Jaron Z. Lanier. A Hand Gesture Interface Device. In *SIGCHI/GI*, pages 189–192. ACM, 1987.