Whole-Hand and Speech Input in Virtual Environments

Joseph J. LaViola Jr., Master's Thesis

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-99-15
December 1999

Whole-Hand and Speech Input in Virtual Environments

by
Joseph J. LaViola Jr.

B.S., Computer Science, Florida Atlantic University, 1996

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Master of Science
in the Department of Computer Science at Brown University

Providence, Rhode Island
December 1999

This dissertation by Joseph J. LaViola Jr. is accepted in its present form by the Department of Computer Science as satisfying the thesis requirement for the degree of Master of Science.

Date ______________ ____

Andries van Dam, Director

Recommended to the Graduate Council

Date ______________ ____

David H. Laidlaw, Reader

Date ______________ ____

Robert C. Zeleznik, Reader

Date ______________ ____

William A. S. Buxton, Reader
Alias—Wavefront, Inc.

Approved by the Graduate Council

Date ______________ ____

Peder J. Estrup
Dean of the Graduate School and Research

Abstract

Recent approaches to providing users with a more natural method of interacting with computer applications have shown that more than one mode of input can be both beneficial and intuitive as a communication medium between humans and computers. Two modalities in particular, whole-hand and speech input, represent a natural form of communication that has been ingrained in our physical and mental makeup since birth. In this thesis, we investigate the use of whole-hand and speech input in virtual environments in the context of two applications domains: scientific visualization and interior design. By examining the two modalities individually and in combination, and through the creation of two application prototypes (Multimodal Scientific Visualization Tool and Room Designer), we present a number of contributions including a set of interface guidelines and interaction techniques for whole-hand and speech input.

Acknowledgements

I wish to thank the members of my thesis committee, Robert Zeleznik, David Laidlaw, Andries van Dam, and William Buxton for their support, direction, and guidance in the development of the ideas presented in this work. In addition, I thank IBM for their financial support for the last two years.

I also would like to thank the members of the Brown University Graphics Group for their endearing support. In particular, I thank Andy Forsberg, Tim Miller, Rosemary Simpson, Steve Dollins, Tim Rowley, Christine Waggoner, Mark Oribello, Mike Legrand, Brian Perkins, Daniel Acevedo, Rebecca Sun, and Mark Zeldis.

Finally, I thank my mother, father, and brother for never letting me get down and helping me to maintain the energy and drive to finish this work when things got tough.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Interaction represents one of the most important components in virtual environment[1] (VE) applications; there have been many interface devices, techniques, and models that have been researched and analyzed for the purpose of finding usable and robust VE interfaces. One interface style that has shown potential in creating useful and robust interfaces is multimodal interaction. Although multimodal interfaces have existed in computer UI's since the early 1980's with Bolt's "Put That There" system [13], they have only recently (since the early 1990's) begun to be examined and incorporated in virtual environments and other 3D applications.[2]

There are many different types of individual modalities that can be combined to form multimodal interfaces [27, 76, 144]. Two of the most interesting are whole-hand and speech input, since these modalities represent a natural form of communication that has been ingrained in our physical and mental makeup since birth. On a person-to-person level, humans use these modalities in everyday conversation, so an interesting question arises as to the best way to use whole-hand and voice input in virtual environments on a human-to-computer level.

1.1 Objective

The main objective of this thesis is the development of a set of practical guidelines and interaction techniques for using whole-hand and speech input in virtual environment applications. We focus on two domains, 3D scientific visualization and interior design, with the

[1] Virtual environment and virtual reality are used synonymously throughout this document.

[2] One could consider Bolt's system a VE application since users are placed in front of a large rear-projected screen and interact using magnetic trackers. However, "Put That There" was a 2D application and had no stereoscopic viewing.

hope that the guidelines and techniques reported can extend into other VE applications.

In order to achieve this objective, it is important to understand not only how whole-hand and voice input can work together, but also how they can and have been used in unimodal virtual environment interfaces. An understanding how to improve upon these individual modalities inherently strengthens them when they are combined multimodally. Therefore, this work also analyzes the issues involved with using whole-hand and speech input in isolation resulting in the development of a number of solutions to problems with these individual modalities.

1.2 Contributions

The contributions in this work are presented under four categories which are:

1. **Interaction Analysis**
   - A survey into the issues involving whole-hand input with specific interest in posture and gesture recognition
   - A classification scheme for the information gathered with whole-hand input
   - A classification of speech input methods with a discussion of the problems and their solutions for using speech recognition
2. **Input Devices**
   - Flex and Pinch input – a hybrid whole-hand input device
3. **Interaction Techniques**
   - A framework for combining whole-hand input and speech into a multimodal interface
   - A discussion and implementation of interface techniques integrating both whole-hand and speech input
4. **Applications**
   - A fluid flow visualization application
   - A conceptual modeling (furniture layout) application

1.3 Reader's Guide

Since many of the chapters in this thesis contain background information and novel work, this section presents the reader with a brief description of each chapter's contents and identifies novel sections in parentheses.

Chapter 2 – Discusses the use of whole-hand input in virtual environments and prior whole-hand data taxonomies, presents the geometrical/topological whole-hand input classification scheme (Section 2.3), details on Flex and Pinch input (Section 2.4), and discusses some improvements to existing whole-hand interaction techniques (Sections 2.5).

Chapter 3 – Discusses the problems and issues associated with speech recognition in virtual environment applications and presents methods for solving these problems (Section 3.3).

Chapter 4 – Discusses the combination of whole-hand input and speech in multimodal interfaces and identifies a number of advantages (Section 4.3) for using multimodal interaction in virtual environment applications.

Chapter 5 – Presents the hardware configurations and software framework (Section 5.2) used in implementing the two applications described in Chapter's six and seven.

Chapter 6 – Discusses the motivation, features, interaction techniques (Section 6.1), and qualitative evaluation of a scientific visualization application for viewing flow around a dataset.

Chapter 7 – Discusses the motivation, features, interaction techniques (Section 7.2), and qualitative evaluation of a room layout/interior design application which allows users to populate naive environments.

Chapter 8 – Presents a set of guidelines for using whole-hand input and speech in virtual environments (Section 8.1) and a summary of the novel and improved interaction techniques (Section 8.2).

Chapter 9 – Presents conclusions and areas for future work.

Appendix A – Presents an introduction to the concepts and criteria used in describing hand postures and gestures.

Appendix B – Discusses the requirements for hand posture and gesture recognition. It describes the two main solutions for collecting the required data to perform recognition, the glove-based solution and the camera- or vision-based solution, and looks at the advantages and disadvantages of each (Section B.4).

Appendix C – Describes the various feature extraction and classification algorithms used in hand posture and gesture recognition, and discusses the advantages and disadvantages of each (Sections C.1 and C.2).

Appendix D – Describes the components and design decisions made for the Flex and Pinch electronics unit.

Chapter 2

Whole-Hand Input

Whole-hand input presents a method of interaction which allows users to directly control computer applications with their hands. In almost all cases, users interact and control these applications with some type of hand posture or gesture, where a posture is considered to be a static pose while a gesture is a dynamic movement[1]. In virtual environments, there has been a significant amount of work in using whole-hand input to control VE applications. The majority of these techniques use either pointing, grabbing, or pinching to interact in the VE.

2.1 Previous Work Using Whole-Hand Input in VEs

In virtual environments, usually the three most important types of interaction that the user can perform are navigation through the VE, object selection and manipulation, and object creation. With whole-hand input instead of other less direct interfaces, it has been shown that users can perform these tasks in a more natural, intuitive, direct, and efficient manner [134]. Many of these techniques use hand postures and gestures that humans use in the real world which would not be possible with other less direct interface devices.

Among the many techniques for navigation is the use of hand gestures for flying through the VE. Mine [95] developed a two-handed flying metaphor where users point in the direction they want to go by creating a vector between the two hands, with the distance between the hands being used to control velocity. Mine also developed the scaled-world grab navigation technique in which users grab an object in the desired location and pull themselves to that location. This technique provides two important benefits. The first is that users can

[1]See Appendix A for hand posture and gesture definitions.

reach any visible destination in one gesture, and the second allows them to view the object from all sides by simply twisting the wrist while still in grab mode. Another approach to navigation is *virtual walking* developed for Multigen's SmartScene^{TM} application [87, 98]. With this technique, users pull themselves through the environment by pinching space, one hand after the other. When they pinch with both hands simultaneously, rotation, scaling, and translation operations can be performed with one movement.

The most traditional methods for selecting and manipulating objects are pointing, reaching, grabbing, and pinching. Sturman, Zeltzer and Pieper [134] not only explored the use of pointing and grasping postures and gestures for object selection and manipulation, but also the use of hand postures as logical buttons and of hand gestures for menu operations and sliders. Davis [33], and Bryson [20] have also used pointing and grabbing for object interaction in VEs. In another approach, Pierce's image plane interaction techniques [111] used four distinct hand postures for object selection and manipulation including a lifting palm posture (see Figure 2.1), a two-handed framing posture (see Figure 2.2), a single outstretched finger, and a head crusher posture[2]. Rehag and Kanade have also used hand posture and gesture recognition to create a 3D mouse for object manipulation in a virtual environment [120].

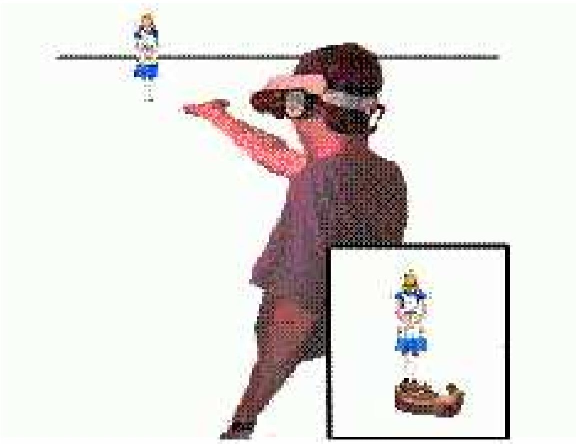

Figure 2.1: The Lifting Palm object selection technique.

Whole-hand input also has advantages for object creation in VEs since the user can directly create objects with the hands. For example, Krueger's VIDEODESK system allows users to create 2D and 3D objects by using the silhouette of the hand and pointing [73]. Weimer and Ganapathy use hand gestures to create B-spline-based 3D models [148], and

[2]The head crusher posture has the user place the thumb and forefinger around the desired 2D image of the 3D object to be selected.

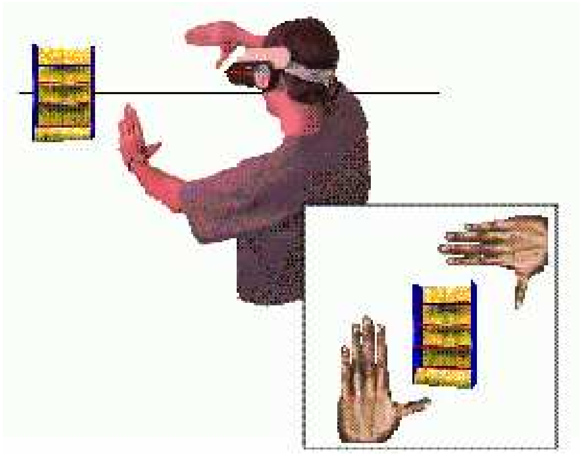

Figure 2.2: The Framing Hands object selection technique.

Utsumi uses static hand postures to create simple 3D geometric primitives [139]. In addition, Schkolne and Schröder [124] use hand motions to form conceptual 3D shapes.

2.2 Whole-Hand Input Classification Systems

Another area of work, which does not directly pertain to whole-hand input in VEs but which nevertheless is still an important topic, is the classification of various types of whole-hand input that users can perform. These classification systems and taxonomies are beneficial because they provide a basis for understanding how humans communicate with whole-hand input from psychological and physiological perspectives. This understanding facilitates mappings between human postures and gestures, and computer interaction. The mappings aid researchers in designing applications that use whole-hand input and provide a grounding for discussion.

2.2.1 Sturman's Whole Hand Input Taxonomy

Sturman's Whole Hand Input Taxonomy [132] is designed as a mapping between categories of hand actions and their interpretations. According to Sturman,

> Hand actions are defined as position, motion, and forces generated by the hand. The interpretation of hand actions are the functional interpretation made by the user and/or the applications of the hand actions.

Hand actions fall into two categories: continuous features and discrete features. Continuous features are based on the degrees of freedom of the hand and include such continuous quantities as fingertip position, joint velocities, and direction of motion. Hand gestures fall into this category, as do the forces on the pads of the fingers and palm. Discrete features are based on static values of the features of the hand. Hand postures, such as a fist or a pointing posture, fall into the discrete feature category.

Interpretation of hand actions is divided into three categories: direct, mapped, and symbolic. In direct interpretation, the user is physically interacting with the virtual world as if it were the real world (e.g., when users grab a virtual world object and place it on a virtual table in the same way they would grab a real coffee mug and place it on a real table). Direct interpretation also includes interaction in which the hand mimics the actions of the controlled object. In a mapped interpretation, data from the hand is mapped to some virtual input device such as a button or slider; such as the flexion of the index finger to manipulate a slider that changes an interocular distance parameter for stereoscopic viewing. Finally, in symbolic interpretation, users specify a hand posture or gesture that is cognitively mapped to some function or task. For example, a series of hand gestures can signify a token stream used in the recognition of American Sign Language (ASL).

Using the two categories of hand action and the three categories of interpretation, Sturman derives six categories that classify whole-hand input:

Continuous/Direct. Continuous hand data is mapped to a kinematically similar action: a graphical hand follows a user's real hand motion.

Continuous/Mapped. Continuous hand data is mapped to some logical input device: finger movements map to mouse cursor movements.

Continuous/Symbolic. The application interprets continuous hand data and determines the user's intention: in navigating through a virtual environment, waving the hand or fingers in a particular direction to indicate motion in that direction.

Discrete/Direct. Discrete hand data or a hand posture is mapped to a directly manipulative task: Sturman claims that this category is rarely used except in applications such as direct robot control.

Discrete/Mapped. Discrete hand data is mapped to a discrete activation level: an object is animated as long as the user makes a fist.

Discrete/Symbolic. Discrete hand data is used to generate commands in an application: a user makes a halt posture to make an object stop moving.

2.2.2 MIT AHIG's Gesture Classification System

The AHIG gesture classification system was first discussed in Wexelblat [150] and is also indirectly discussed in Cassell [24] and Wilson et al. [151]. AHIG's classification system starts from the idea that previous gesture classification systems, such as Efron's [38], Kendon's [66], and Nespoulous and Lecours' [101], are oriented to the psychological domain and do not necessarily apply to computer applications. The system is broken up into five major categories:

Symbolic/Modalizing. Symbolic gestures are essentially hand postures used to represent an object or concept, and are always directly mapped to a particular meaning: for instance, the 'thumbs up' posture means that everything is okay. Modalizing gestures are gestures used in conjunction with another input modality (e.g., speech). For example, in asking whether someone had seen a particular person, extending the hand out at a certain level could indicate the person's height.

Pantomimic. Pantomimic gestures involve using the hands to represent a task or interaction with a physical object. Users making this type of gesture mimic an action they would do if they were actually interacting in the real world: for example, making a swinging gesture with the hands to indicate hitting a baseball with a bat.

Iconic. Iconic gestures are gestures that represent an object. The hands become the object or objects discussed. These gestures are usually performed to act out a particular event in which the representative object is the focal point, such as someone pretending to drive a car.

Deictic/Lakoff. Deictic gesture or pointing gestures are used to indicate a particular object. The other type of gesture included in this category are Lakoff gestures [75], associated verbal utterances that specify a particular metaphor such as happiness or anger. A gesture usually accompanies these utterances to show the directionality of the metaphor.

Beat/Butterworth's/Self-adjusters. Beats are gestures used for emphasis, especially when used with speech. Beat gestures can help speakers emphasize particular words or concepts and also help direct the listener's attention. Butterworth gestures [22] are similar to beats except they are primarily used to mark unimportant events. The classic example of a Butterworth gesture is 'hand waving' as a placeholder for speaking when one is still thinking about how to say something. Finally, self-adjusters are gestures people make when they fidget: for example, when one taps a finger or moves a foot rapidly.

2.3 Geometrical/Topological Hand Data Classification

Although the classifications systems described in the previous section are useful for providing mappings between human postures and gestures and computer interfaces, they are theoretical in nature and, as a result, are not useful to UI developers when it comes to practical implementations. Therefore, what is required is a more practical classification scheme that would take into account some of the more "implementation specific" details such as input devices. Based on this argument and on the analysis of hand posture and gestures found in Appendices B and C, we developed a simple, yet practical classification scheme which categorizes whole-hand input data as shown in Figure 2.3. We found that whole-hand input can be described in two ways; through geometry and through topology, and this categorization led to the development of the Flex and Pinch input system (see Section 2.4).

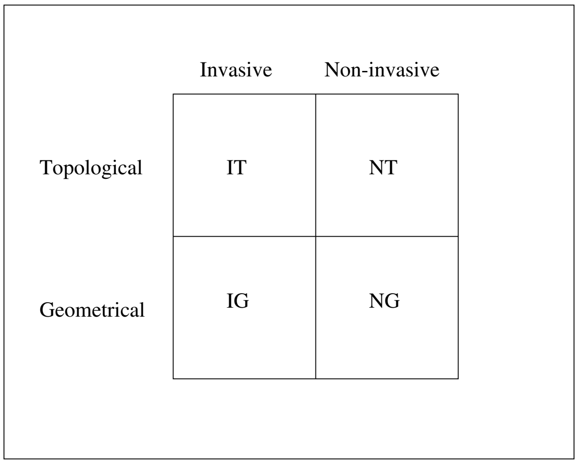

Figure 2.3: The Geometrical/Topological hand data classification scheme. The letters inside the quadrants are acronyms for a particular component. For example, NT stands for a non-invasive approach using topological data while IG-IT stands for an invasive approach which uses both geometrical and topological data.

There are two basic approaches to using whole-hand input in virtual environments. The non-invasive approach uses vision-based tracking [74] so the user is not physically attached to the computer. The invasive approach uses a glove-based device or devices to extract information from the hands. In each approach, we can extract two different types of data, namely geometrical data and topological data. Geometrical data represents information about the hand's shape, location, and orientation, while topological data provides information about how the fingers touch each other, other parts of the hand, and any structure in the physical environment. Although a non-invasive approach may be preferred, it is difficult to extract both geometrical and topological information due to problems with computer vision, such as occlusion. Therefore, we focus on the invasive approach instead, quadrants IT and IG.

With the invasive approach, two types of glove-based input devices have been developed. The first, bend-sensing gloves [103, 142, 155] (the IG quadrant), measure finger joint movement, and second, the Pinch Glove [42, 87] (the IT quadrant), detect electrical contacts between each of the finger tips. Unfortunately, both bend-sensing and pinch gloves have faults when used in isolation. Bend-sensing gloves are good at extracting geometrical information which enables them to represent the user's hands in the virtual environment. They can be used to mimic interface widgets such as sliders and dials [131], but do not have useful methods for signaling the activation or deactivation of the widget. Bend-sensing gloves are also used in conjunction with hand posture and gesture recognition, but it can be difficult to determine when one gesture begins and another ends without applying constraints to the user's gesture space [132]. Conversely, Pinch Gloves provide a series of button widgets that are placed on each finger tip which allows for the extraction of topological data for interactions, such as pinching postures. However, they have no way of determining the flexing of the fingers and they make it difficult to represent the hand in a virtual environment.

There have been few attempts to combine the two types of information that each type of data glove provides. With the exception of Grimes' Digital Data Entry Glove, which was developed specifically for entering text using the Single Hand Manual Alphabet [54], the author knows of no other work done with combining discrete and continuous whole hand input devices to extract both geometrical and topological data simultaneously.

2.4 Flex and Pinch Input

In order to develop an interface that spans the IG and IT quadrants of the geometrical/topological classification scheme, we built a hardware prototyping system for testing and evaluating different interface designs. The hardware system provides a number of benefits in that it employs a plug and play strategy for quickly adding and removing button widgets or their components. Our system enables users to incorporate up to 16 cloth sensors in a wearable interface. Conductive cloth [86] sensors[3] provide two important functions: first, each sensor knows when it comes in contact with another sensor and specifically which other sensor it contacts, second, the nature of the cloth lends itself for use on gloves or clothing.

Using our prototyping system, we constructed a device based on the Fakespace Pinch Glove [42]. As a hardware input device[4], it provides more functionality than the Pinch Glove since it uses eight cloth buttons instead of five which allows for more button combinations. In general, five of these cloth buttons can be placed around each of the finger tips, while the other three can be placed arbitrarily about the hand. This configuration represents one of many possible combinations for placement of the cloth buttons. The device could be worn with anywhere from two to 16 cloth buttons of any shape or size. In addition, the cloth buttons can also be placed on other objects in the physical environment such as a control panel for issuing commands or entering numbers. This robustness presents a clear advantage over other inflexible input devices. Using this device, we can augment existing bend-sensing gloves to create Flex and Pinch input which seamlessly integrates geometrical and topological hand data (see Figure 2.4).

2.5 Interaction Techniques Using Flex and Pinch Input

With Flex and Pinch input, we can improve on a number of existing techniques for selecting objects in virtual environments and create new techniques that could not be developed without the combination of geometrical and topological data. For example, one of the major problems with image plane interaction techniques (see Section 2.1) such as the head crusher, sticky finger, lifting palm, and framing hands object selection techniques [111] is that the user cannot activate the selection with the primary hand. As a result, the user requires an additional, separate input device for triggering the selection operation.

Flex and Pinch input provides a simple yet effective and seamless method for starting and stopping object selection by placing the cloth buttons in appropriate places on the user's primary hand. For example, with the head crusher technique, we can place the

[3] These cloth sensors were made by hand using patches of conductive cloth attached to wires which were fed into an electronics unit.

[4] See Appendix D for a description of the electronics unit.

Figure 2.4: The Flex and Pinch input system. The cloth contacts represent the "pinch" part of the device collecting discrete topological data while the glove represents the "flex" part collecting continuous geometrical data. Although a CyberGlove [142] is shown, any bend-sensing glove can be used.

cloth buttons on the thumb and middle finger so when the user positions the thumb and forefinger around the object a middle finger to thumb contact will signal the object should be selected. Another button press would signal the release of the object. The cloth contacts can be placed in other positions such as on the middle finger and on the palm by the base of the thumb or on the right side of the index finger and the left side of the middle finger (see Figure 2.5).

In a similar manner, cloth contacts are placed on the hand for the sticky finger and lifting palm (see Figure 2.6 techniques to start and stop object selection while cloth contacts are placed on both hands for the framing hands selection technique. Figure 2.7 shows the Head Crusher technique with placement of the cloth contacts between the forefinger and middle finger.

Another method that has been used for selecting objects in virtual environments is to select a given object by casting a laser into the scene from the user's hand [59]. As with the image plane techniques, the problem with laser pointing is that it is difficult to start and stop the selection with only one input device. For example, one laser pointing object selection method uses a point and clutch posturing mechanism to select objects in a virtual environment where clutching is performed by flexing the thumb [76]. The problem with using this clutching mechanism is that in order to achieve robust recognition, the user must make postures using extreme configurations of the hand which puts undo strain on the

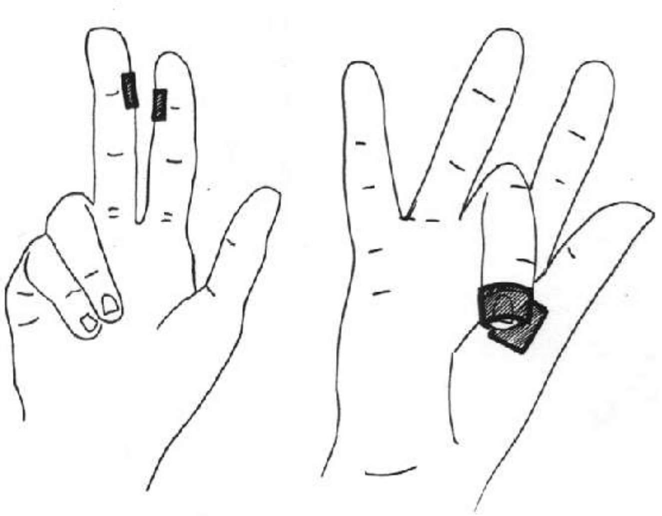

Figure 2.5: Two examples of how the cloth contacts can be placed on the hand when using the head crusher object selection technique.

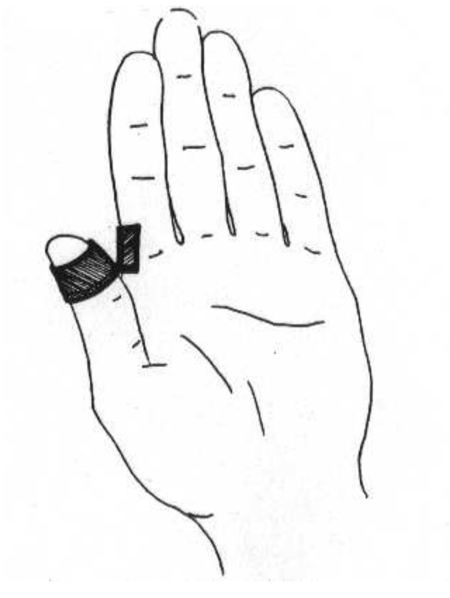

Figure 2.6: An example of how the cloth contacts can be placed on the hand using the lifting palm object selection technique.

two tendons in the thumb. Using Flex and Pinch input we can alleviate this problem by placing cloth contacts on the thumb and on the right side of the middle finger as shown in Figure 2.8. This provides a much more natural movement and puts no strain on the thumb tendons[5].

[5]One could argue that the user could make a posture that is identical to the user's hand configuration

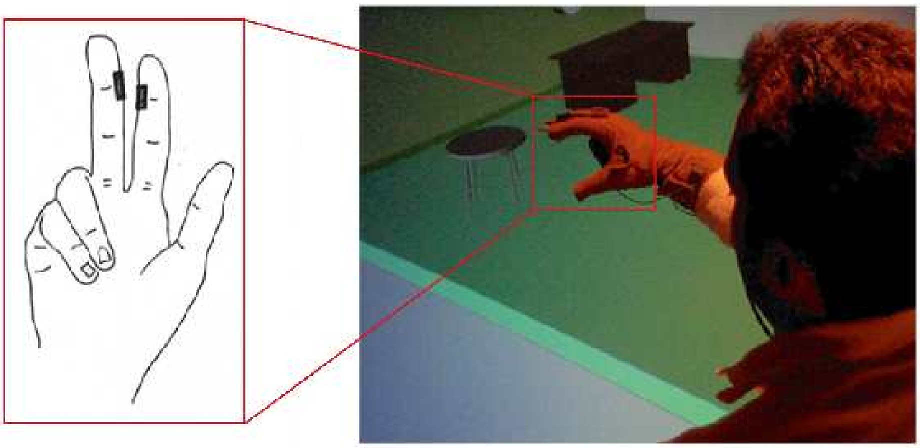

Figure 2.7: A user wearing the Flex and Pinch input device is about to invoke the head crusher object selection technique on a round table. By placing his middle and index finger together, as shown in the drawing, the user can activate the selection operation and move the table.

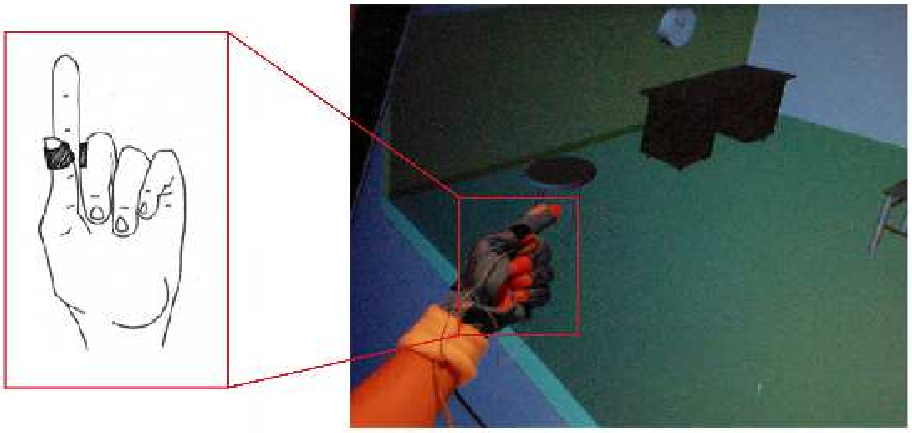

Figure 2.8: A user pointing at and selecting a desk in the virtual environment. The user makes the selection by pressing the thumb to the right side of the middle finger as shown in the drawing.

Bend-sensing gloves have the capability of being used as analog sliders since these gloves report continuous measurements of the joint angles in the hand. However, used in isolation,

when using Flex and Pinch input. However, hand gesture and posture recognition is not perfect, and if the hardware is working properly, the pinching mechanism will provide 100 percent accuracy.

it can be difficult to determine when the user wants to actually use one of the fingers as a slider widget. Using Flex and Pinch input, a seamless transition between the discrete events from the cloth contacts and the continuous updating from the bend sensors can be made which provides a mechanism for activating and deactivating the sliders when needed. For example, we can cycle through menu items with a finger[6]. A button press creates the menu and as the button is held, users can cycle through the menu items by flexing or extending a finger. If users do not wish to select an item, they need to release the button when their finger is fully extended or fully flexed. We are currently exploring how many menu items a user can easily invoke using this technique. Using the same configuration we also can change an object's scaling, translational, and rotational parameters.

Finally, an important benefit of using the pinch component of Flex and Pinch is that it gives application developers a method to test out different hand postures and gestures. In many cases, when developers want to test a new hand posture or gesture, they have to retrain their gesture recognition algorithms [43] which can be time consuming. The pinch component of Flex and Pinch input allows them to quickly move cloth contacts from one part of the hand to another without having to change any software components or restart the application. This allows application developers to quickly test the feeling and ergonomics of certain hand postures and gestures. Also, with the ability to move the cloth contacts anywhere on the hand, we can create whole-hand interfaces that could not be implemented when using either a bend-sensing glove or the Pinch Glove separately.

[6]In this case, one cloth contact is placed on the thumb while the second is placed on the left side of the forefinger between the proximal interphalangeal and metacarpophalangeal joints (see Figure B.1 for a description and the location of these joints).

Chapter 3

Speech Input

Over the past several years, speech recognition technology has advanced to the point where speech input has become a viable interaction mode in computer interfaces. This mode has important uses in command and control, telephony, dictation, and other applications. Speech interfaces are not only powerful in desktop applications, but they also show a great deal of promise in virtual environment applications since speech can be used to replace traditional desktop input devices[1] such as the keyboard and mouse buttons. Although there has been some previous work using speech input in virtual environments applications [13, 47, 87, 148], it a relatively new interaction style with respect to VEs. As a result, there are a number of problems that arise when using speech recognition systems in virtual environments. This chapter identifies some of these problems, presents solutions for solving them, and discusses their pros and cons.

3.1 Types of Speech Input Systems

There are two basic types of speech recognition systems; the speaker-dependent and the speaker-independent system [122]. A speaker-dependent system requires the user to train on a set of words so that the recognizer adapts to that particular voice. The advantages of a speaker-dependent system are that the more the user trains the system, the better the recognition accuracy (to a point), and other voices in the physical surroundings usually do not get recognized. However, the disadvantages of a speaker-dependent system are that it can be very time consuming to train to a satisfiable accuracy and, as a result of the training requirement, the ability for a user to quickly start using the system is limited.

[1] These traditional devices are not typically found in immersive virtual environments.

In contrast, a speaker-independent system requires no training for a particular user which presents an advantage in that anyone can start using the system at any time. However, since a speaker-independent system is not trained to a user's particular voice, any noise from the surrounding environment is treated as a possible speech utterance, which can cause undesirable effects.

Within speaker-independent and -dependent speech recognition systems, there are two principal recognition modes that exist. The first is called isolated word recognition and the second is continuous word spotting [122]. With isolated word recognition the user must distinctly pause between word utterances. This approach is easier to implement but is not the way humans naturally speak. The continuous word spotting mode is a much more natural way for users to issue speech commands since no distinct pauses are required. However, it is much more difficult to implement since the recognition engine must be able to detect separate words, without artificial pauses to separate them.

3.2 Practical Issues with Speech Input

The ultimate goal with speech input is to let users speak to the computer in the same way that they speak to human beings. Even with the most accurate speech recognition systems this goal is extremely difficult to achieve. Current speech recognition engines, such as the BBN Hark system or Dragon System's NaturallySpeaking^{TM} have the ability to provide recognition accuracy levels in excess of 95 percent when used in controlled settings. However, there still are a number of environmental issues that come into play when dealing with speech input that are both directly and indirectly related to the speech recognition itself.

Speech Direction. One of the most important issues is how to let the computer know that the user is speaking to it versus speaking to someone else either in the physical environment or the virtual environment. One possible method is to use a push-to-talk interface where users must somehow signal the computer they are going to speak to it (push-to-talk interfaces are discussed in the next section).

Microphone Placement. The microphone can be placed on the user via headset or lavalier or somewhere in the physical environment. Placing the microphone on the user has the advantage that its close to the user's mouth which allows for a clearer input signal to the recognition engine and allows the user to speak at a normal or even soft volume.

However, users have to wear the device and this adds to the list of input devices they are already wearing in a non-vision-based virtual environment. The alternative to wearing the microphone is to place it somewhere in the physical environment. This approach gets the microphone off the user's body but presents another set of problems. With the microphone in the physical environment, users will either have to speak at a volume higher than they want or the microphone will have to be very sensitive. Unfortunately, a sensitive microphone is much more susceptible to background noise.

External Noise. External noise represents a major obstacle in using speech input since it can distort the input signal and cause the recognition engine to recognize utterances that it was not supposed to and fail to recognize those that it should. This noise can come from a variety of sources such as talking in the physical environment and noise from running machines. In the case of virtual environments, magnetic tracking devices emit a low frequency signal which microphones can pick up. In these situations, this low frequency signal will send a continuous stream of noise to the recognition engine causing problems.

Recognition Latency. Speech recognition latency represents the time between the input utterance and output of the recognition engine. Recognition latency can severely hamper the use of the system especially when interactive virtual environments are concerned. In many cases, recognition latency is proportional to the size of the input vocabulary since the recognition engine has more possibilities to consider.

3.3 Speech Input Solutions

When dealing with solutions to these speech input issues, trade-offs must be made between a natural, humanistic style of speech and a more computerized style of interaction. The key, of course, is to find a common ground between the two so that speech input can be a natural method of interaction and work robustly and accurately in the presence of these problems.

Speech Direction Solutions. As stated in the previous section, a push-to-talk interface is an obvious solution to solving the speech direction problem, but the interface's implementation is not so obvious. There are many possible approaches to implementing a push-to-talk interface. For example, we can use a foot pedal which users depress every time they want to speak to the computer. The problem with this approach is that, first, it increases the cognitive load since users have to remember to push down the foot pedal every time they want to speak, and, second, a foot pedal will not work in all virtual environments[2]. To get around the problems with foot pedals in certain types of virtual environments, we can simply have users press a button worn on their person. However this does not solve the increased cognitive load issue.

Another approach to implementing the push-to-talk interface is to have the user's gaze tracked with either vision tracking or, more crudely, with a worn tracking device (a Polhemus tracker for example) so that an "active" region can be specified. Whenever users are looking at this active region, they can issue speech commands to the computer. While the gaze-directed approach will be transparent to the user in some situations, this solution also has a number of faults. For example, it may fail in collaborative settings when the user wishes to speak to someone else other than the computer. As in natural collaborative interaction, the user may turn to look at the collaborator before speaking or may forget the "active" zone is there and speak to the collaborator while still gazing into the zone. In this case, the user must, once again, increase his/her cognitive load by remembering to always turn towards the collaborator when speaking to that individual. Another problem with the gaze-directed approach is that it also can fail in virtual environments which use large and/or panoramic display devices. With these devices, the "active" zone must be large enough so users can look at all the parts of the display and still issue speech commands.

A fourth approach to implementing a push-to-talk interface is to let users tell the computer when to listen. With this type of interface, the user has *start* and *stop* keywords which tells the speech recognition engine to pay attention to the user's utterance's or not. Due to the nature of the implementation, it avoids many of the problems the previous approaches had due to virtual environment issues. However, it still adds to the cognitive load since users must remember to tell the computer whether or not to listen to their speech. In addition, false positive recognition can occur.[3]

The best approach to implementing a push-to-talk interface is to embed the "push" part of push-to-talk into an existing interaction technique so that it is transparent to the user. With this approach we take advantage of the fact that the user is already using some mechanism to trigger interactions in the virtual environment. We piggyback the "push" part of push-to-talk onto these mechanisms without increasing the user's cognitive load.

[2]The foot pedal fails in cases such as the Cave where a display surface exists on the floor and with head-mounted displays where the user cannot see the physical environment. Foot pedals would be appropriate for VEs that use workbench style displays.

[3]For example, the recognizer thinks the user said "Computer Start Listening" when he/she really didn't.

This approach has been used in systems such as QuickSet [27] where speech recognition activates whenever the user touches a stylus to the screen but has not been used in virtual environments. In another example, if users want to instantiate an object and place it in the VE, they can point to the place they want the object to go and, in the act of pointing, (triggered by a gesture or a button press for example) activate the speech recognition. When they stop pointing speech recognition moves into non-active mode again. This approach works well (it is used in Room Designer, described in chapter 7), gets close to the naturalistic style of interaction described in the beginning of section 3.2, and does not possess the drawbacks found with previous approaches.

Microphone Placement Solutions. In a perfect, noise-free environment having the microphone mounted somewhere so that the user does not have to wear an extra device is the best solution. However, noise-free environments are rare, so the alternative is to have the microphone as close to the user's mouth as possible. Headsets perform this function well but then users have to wear an additional wired device. This approach is even more uncomfortable with virtual environment applications since users are already wearing something on their head such as a HMD or shutter glasses. A better approach is to use a wireless lavalier since the microphone can still be placed close to the mouth, with no additional wires.

External Noise Solutions. The two basic approaches to solving the external noise problem are to either reduce external noise in the physical environment or to not let the recognition engine know that external noise exists. Reducing external noise in the environment is easier said than done since there may be noisy machines or people that cannot be moved. The second approach is somewhat more practical since microphone and speech recognition parameters can be altered to help block out the noise. For example, many speech recognition systems[4] have sensitivity adjustments which determine how much of the input signal the recognition engine will pick up. With this adjustment, the system can be tweaked so that it does not pick up external noise, but the user will have to speak in a much higher volume than normal. One can also use a unidirectional microphone or filtering algorithms to help block external noise.

Recognition Latency Solutions. The latency in a speech recognition system is often a direct reflection of vocabulary size. The larger the vocabulary, the longer it takes the recognizer to recognize utterances. The most obvious way of reducing recognition latency

[4]This work uses the BBN Hark speech recognizer which has sensitivity adjustment capability.

trimming down the vocabulary, may not always be possible. Another latency reduction approach is to increase the recognition engine's decoder speed[5] so that it does not go through as many possibilities. Latency will be reduced with this parameter change but since fewer speech input choices are considered, accuracy will definitely diminish. Finally, an alternative method is not to reduce it, but to mask it in some way. If the user is given something to do or see during the latent period, this may minimize the effects of the lag although this hypothesis has not been tested and is an area for future work.

[5]Decoder speed is another parameter which can be changed in the BBN Hark system.

Chapter 4

Combining Whole-Hand and Speech Input

We have seen in the previous two chapters that while both whole-hand and speech input can be powerful interaction styles in virtual environment applications there are numerous problems with them. We have also analyzed some of these problems and presented solutions. Although whole-hand and speech input show promise as individual interaction styles, their combination into multimodal interfaces shows even greater potential since it is this combination on which human-to-human communication is based; a communication medium that has been ingrained in the human's physical and mental makeup since birth. Therefore, in this chapter, we discuss some aspects of multimodal interaction, describe previous work in the area, and discuss the advantages that multimodal interfaces can provide the user and developer.

4.1 Multimodal Interaction

Multimodal interaction can be defined as the combination of multiple input modalities to provide the user with a richer set of interactions compared to traditional unimodal interfaces. The combination of input modalities can be divided into six basic types: complementarity, redundancy, equivalence, specialization, concurrency, and transfer [89]. In this section, we briefly define each.

Complementarity. Two or more input modalities complement each other when they combine to issue a single command. For example, to instantiate a virtual object, a user makes a pointing gesture and then speaks. Speech and gesture complement each other since the gesture provides the information on where to place the object and the speech command provides the information on what type of object to place.

Redundancy. Two or more input modalities are redundant when they simultaneously send information to the application. By having each modality issue the same command, redundant information can help resolve recognition errors and reinforce what operation the system needs to perform [106]. For example, a user issues a speech command to create a visualization tool while also making a hand gesture which signifies the creation of that tool. By providing more than one input stream, the system has a better chance of recognizing the user's intended action.

Equivalence. Two or more input modalities are equivalent when the user has a choice of which modality to use. For example, the user can create a virtual object by either issuing a voice command or picking the object from a virtual palette. The two modalities present equivalent interactions in that the end result is the same. The user can choose which modality to use based on preference (they simply like speech input over the virtual palette) or on frustration (the speech recognition is not accurate enough, thus they move to the palette).

Specialization. A particular modality is specialized when it is always used for a specific task because it is more appropriate and/or natural for that task. For example, a user wants to create and place an object in a virtual environment. For this particular task, it makes sense to have a "pointing" gesture determine the object's location since the number of possible voice commands for placing the object is too large and a voice command cannot achieve the specificity of the object placement task.

Concurrency. Two or more input modalities are concurrent when they issue different commands that overlap in time. For example, a user is navigating by gesture through a virtual environment and while doing so uses voice commands to ask questions about objects in the environment. Concurrency enables the user to issue commands in parallel; reflecting such real world tasks as talking on the phone while making dinner.

Transfer. Two input modalities transfer information when one receives information from another and uses this information to complete a given task. One of the best examples of transfer in multimodal interaction is the push-to-talk interface described in Chapter 3: the speech modality receives information from a hand gesture telling it that speech should be activated.

Although all six multimodal combination types are important to building a richer set of interactions, this work focuses only on four of them: complementarity, concurrency, specialization, and transfer.

4.2 Previous Work

Different types of input mode combinations have been used in multimodal interaction. Zeleznik uses a stylus and puck on a Wacom Tablet to interact with a conceptual 3D modeling application [154]. Cohen uses pen-based gestures and voice commands in QuickSet, a system for setup and control of distributed interactive simulations [27]. Waibel and Vo use a series of input modes that include speech, pen-based gestures, eye tracking, lip reading, handwriting recognition, and face recognition for applications such as text editing and calendar management [144]. The common thread between all of these systems is that the user's hands use 2D input to interact with the application.

In the context of whole-hand and speech input, the use of a multimodal interface that integrates the two modalities can be traced back to Bolt's "Put That There" system [13] developed in 1980. This system used pointing hand postures and voice commands to create, manipulate, and edit simple 2D primitives such as squares and circles using a large rear-projected screen. Bolt extended his earlier work in 1992 with a multimodal interface that used hand gestures along with speech for manipulating 3D objects [14]. Weimer and Ganapathy developed another system that incorporated speech and hand gestures to create B-spline based 3D models [148]. However, their system was menu driven and did not take advantage of whole hand input. Other multimodal work that uses both hand gestures and speech can be found in [4, 10, 69].

An important aspect of multimodal interaction is the integration of the different input modes, for which a number of different integration strategies have been developed. Johnston developed a unification-based integration scheme [63] based on research conducted by Oviatt [105, 107] into people's integration patterns when using more than one mode of input. This scheme uses typed feature structures [23] to represent the semantic contributions of the different modes, which allows for the individual modalities to compensate for each other's errors.

Expert systems have also been used to integrate multiple modes of input as shown in

Billinghurst's work [10]. In his system, a set of if-then production rules, which encode domain knowledge, are used to integrate speech and hand gesture. These rules map high level semantic information from the inputs to generate a somewhat intelligent response. Another approach to input integration is to use frames [143]. In this case, frames consist of slots that hold information from a single input mode. The command interpreter takes these frames and determines the appropriate action to take. An advantage of this approach is its flexibility for incorporating more than two modes of input. Note that other strategies such as agent-based approaches [26] and guided propagation networks [90] have also been developed for integrating multiple modes of input.

4.3 Advantages of Combining Whole-hand and Speech Input into Multimodal Interfaces

Multimodal interaction provides many benefits over traditional unimodal metaphors such as WIMP (Windows, Icons, Menus, Point and Click) interfaces [140]. By combining whole-hand and speech input, human computer interaction is augmented in a number of ways [1]. First, users can interact more naturally since, human-to-human interaction often occurs with combinations of speech and hand movement. Second, an application can achieve a better understanding of the user's intended action by providing it with multiple input streams because speech and whole-hand input cannot provide perfect recognition accuracy.

Combining whole-hand and speech input also has advantages of simplifying the interface not only from the user's perspective but also from the developer's perspective. From the user's perspective, the interface can be simpler since one modality does not have to account for all interactions. For example, if user have to interact solely with speech or whole-hand input, they have to remember either a complicated speech vocabulary or a complicated gesture vocabulary. However, if we combine the modes in a complementary fashion, the set of interactions remains the same as either single modality, yet their respective vocabularies are simplified, easing cognitive load. By combining these two modalities we can also reduce recognition times, increasing interaction speed since each individual recognition system has less work to do and takes less time in making decisions.

From the developer's perspective, the interface is somewhat simpler to implement in terms of algorithmic complexity. In order to provide a robust interface with either speech or whole-hand input (especially hand gestures) in isolation, the developer would have to

[1] Chapters 6 and 7 will present more advantages for combining whole-hand and speech input in the context of the two applications developed in this work.

implement rather complex recognition routines that would require many optimizations to provide fast interaction. Combining these two modalities splits the work allowing for a simpler implementation of each modal component. One could argue that the integration of the two modalities cancels out any of the gains made by having them both in the interface. This argument may be true when a multimodal combination style such as redundancy is used. However, as we will see in the next chapter, when a complementary multimodal combination style is used, the integration of the two modalities is not that complicated.

Chapter 5

Hardware and Software Frameworks

In this chapter, we present the two hardware configurations used in this work, one for a rear-projected display table and one for a surround screen environment. We also describe the software framework used in building a scientific visualization application (Chapter 6) and a room layout/interior design application (Chapter 7).

5.1 Hardware Configurations

The hardware configurations supported investigation of multimodal interfaces in two types of virtual environments: a semi-immersive table environment and a fully-immersive surround screen environment.

5.1.1 Rear-Projected Display Table Configuration

The first hardware configuration (used in the scientific visualization application) has many parts as shown in Figure 5.1. The configuration uses a SGI Octane graphics workstation as its primary computer. The Octane drives a Barco Baron Table (see Figure 5.2), a four foot by five foot rear projection display device. The table has two StereoGraphics CrystalEyes emitters placed on either side of it. These emitters (not shown in the figure) transmit an infrared signal to a pair of shutter glasses the user wears to achieve a stereoscopic view. An Ascension Flock of Birds$^{\mathrm{TM}}$ unit with an extended range transmitter is connected to the Octane through a serial interface. The flock has three trackers, one that is attached to

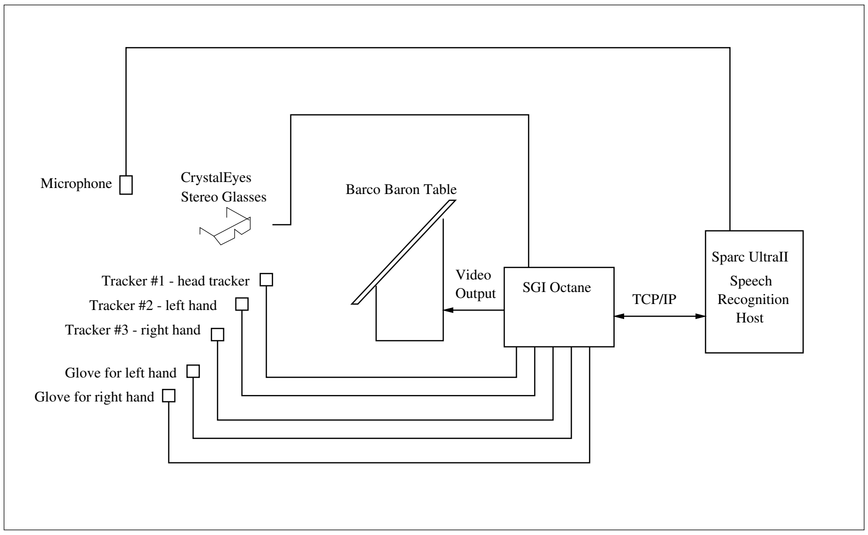

Figure 5.1: The various components that make up the rear-projected display table configuration.

the CrystalEyes for head tracked stereo viewing, and the other two are attached to glove-based input devices for position and orientation measurements. The user can wear a pair of Fakespace Pinch™ Gloves which detect electrical contact at each of the finger tips. As an alternative to the Pinch Gloves, a Nissho Electronics SuperGlove™ can be worn on the left hand, which contains a total of ten bend sensors, and on the right hand, the user can wear a Virtual Technologies CyberGlove™ which has a total of 18 bend sensors.

Figure 5.2: A Barco Baron rear-projected display device.

A second workstation, a Sun Microsystems Sparc UltraII, is used as the speech recognition server. We use the Hark Speech Recognition system, a commercially available product from BBN Corporation. This Sun workstation is physically located in close proximity to the Barco Baron, and as a result, the microphone is connected to this workstation. This configuration is advantageous since no audio signals are sent from the Octane to the UltraII though a network connection. The only data sent across the network are speech tokens from the recognizer. The other advantage of this configuration is the speech recognizer has its own CPU instead of having to share the Octane's CPU which is used for graphics and processing input device records.

5.1.2 Surround Screen Configuration

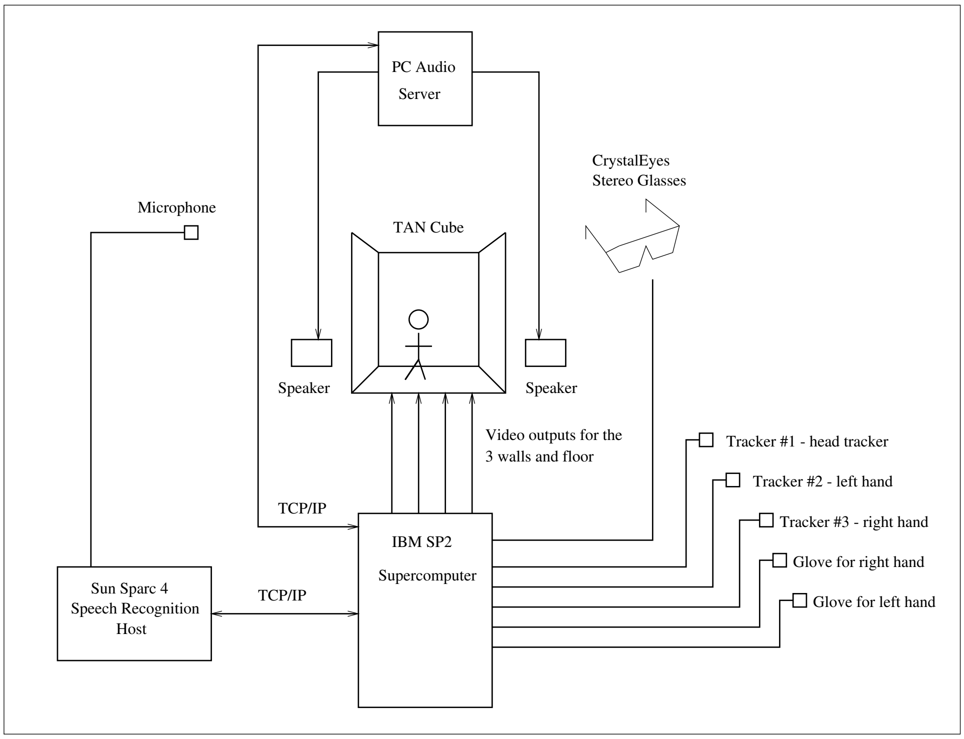

Figure 5.3: The various components that make up the surround screen hardware configuration.

The second hardware configuration (used in the Room Designer application) also has a number of components as shown in Figure 5.3. This configuration uses two nodes of an IBM SP2 as its primary computer. The SP2 drives a TAN VR-CUBE[1] (see Figure 5.4), a four-sided (three walls and a floor) surround screen display system. The VR-CUBE has a total of six StereoGraphics CrystalEyes emmitters placed on the top and bottom of each wall which allows for stereo viewing. A Polhemus FastTrak^{TM} with the Long Ranger transmitter

[1] A TAN VR-CUBE is analogous to a Cave [31].

is connected to the SP2 through a serial interface and mounted on a wood structure which sits above the VR-CUBE. The FastTrak device has four transmitters, one attached to the CrystalEyes shutter glasses, one attached to the right index finger, one attached to the left index finger, and one which can be used for a variety of things such as a wand, 3D stylus, or placed on another part of the body. The user wears a pair of Fakespace Pinch Gloves but the CyberGlove and SuperGlove discussed in the previous hardware configuration can be used if bend-angle measurements are required.

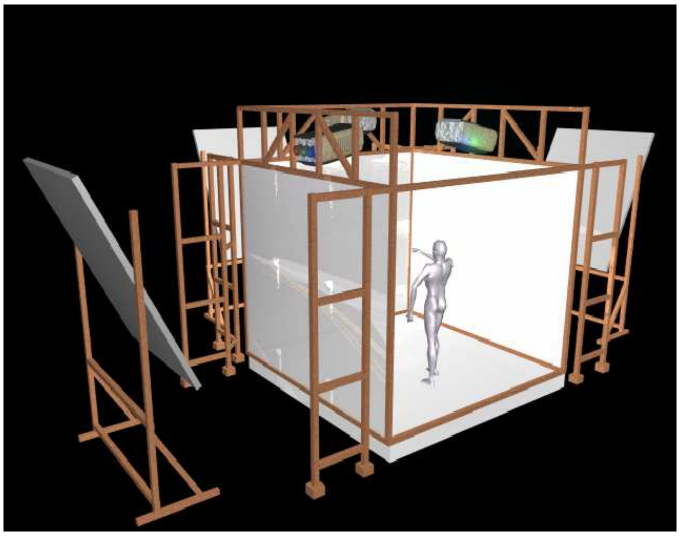

Figure 5.4: A conceptual model of the TAN VR-CUBE display device.

A second workstation, a Sun Microsystems Sparc 4, is used as the speech recognition server and it also uses the BBN Hark speech recognition engine. Finally, a PC sends audio output to two speakers and a subwoofer that surround the VR-CUBE. Both computers communicate with the SP2 via TCP/IP using a client/server model.

5.2 Software Architecture

The software architecture for our multimodal interface framework is based on an interface library called Jot [46, 80]. Jot acts as a lower level infrastructure by supplying device drivers, math routines, network communication, stereoscopic capabilities, and an OpenGL abstraction through a series of classes written in C++. The interface framework consists of a number of low level components and an integration component (see Figure 5.5). Each of the low level components perform either posture, gesture or speech recognition and send tokens to the integration component which uses this data to issue commands in the application.

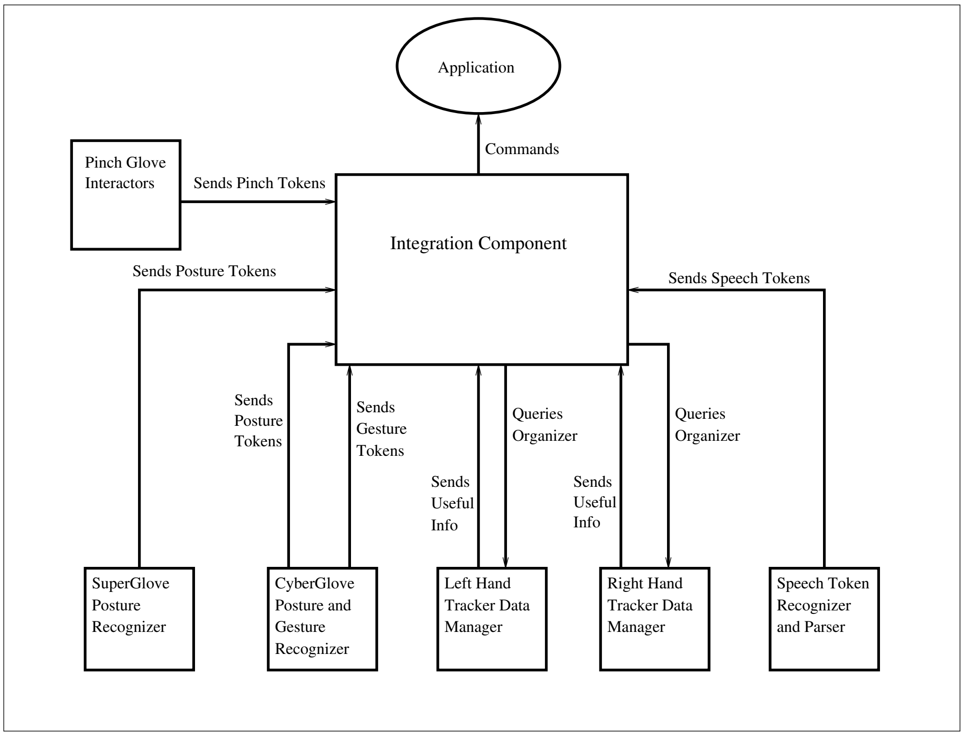

Figure 5.5: The components that make up our multimodal interface framework. Arrow indicate direction of data flow.

5.2.1 Pinch Glove Finite State Automata

One of the basic building blocks in the Jot interface library is the concept of the finite state machine (FSM). Jot provides the developer with a base FSM class which is used to generate application specific finite state machine objects which allow for transitions from one event to another. These transitions are represented as the arcs in the FSM, and, when an arc is traversed, a callback is provided which the developer can use to send information to the integration component, initialize state variables, retrieve information from other parts of application, and execute components or subcomponents of an interaction technique.

With respect to our multimodal interface framework, these derived FSM's are used to determine when "button down", "button up", and "button motion" events are generated from the Pinch Gloves[2]. These events are given logical mappings to various interaction tasks or subtasks based on the application. As an example consider the FSM shown in Figure 5.6 which is used for the navigation technique found in the scientific visualization application described in Chapter 6.

[2] "Button motion" events are actually generated from the attached tracking devices. These devices are in a constant state of motion, thus they are always generating events.

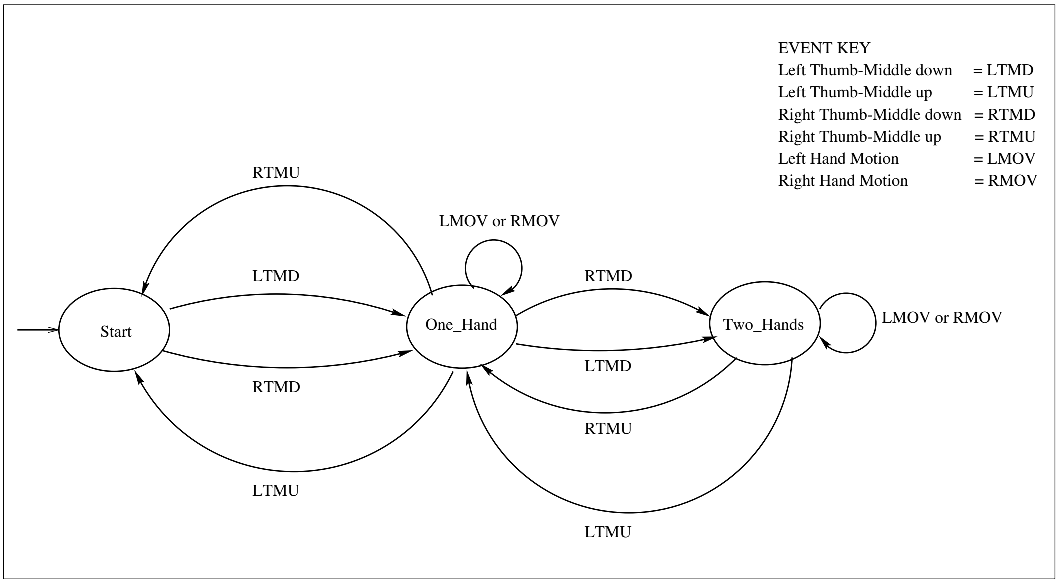

Figure 5.6: An example finite state machine created within our multimodal interface framework. The arcs are event transitions which send the interaction technique into a particular state. In this case, there are three states and a total of 10 event transitions.

This technique allows the user to navigate through the virtual environment with one or both hands based on whether the user's thumb and middle finger in each hand is touching. See Chapter 6 for more details on this technique.

5.2.2 SuperGlove Posture Recognizer

The main function of the SuperGlove posture recognizing component is to process the raw data records from the SuperGlove input device and find recognizable postures which are defined in a template file[3]. The recognizer then sends a token corresponding to the recognized posture to the integration component. The recognition method used is a sum of squares statistic (i.e. similarity statistic) to find the current data record that is most similar to a given templated data record. Postures are recognized in three stages. First a similarity statistic for each possible posture is found using the following formula:

$$ss_i = \sum_{j=1}^{n} (x_{cj} - x_{tj_i})^2 \tag{5.1}$$

where ss_i equals the ith similarity statistic, x_{cj} equals the jth bend angle of the current posture record, x_{tj_i} equals the jth bend angle of the ith templated posture record, and n

[3] A templated data record is created in a calibration file before using the application. The user makes a given posture n times (usually about 5) and an average of each data value is taken to get one record.

represents the number of bend angles measured, in this case ten. Once a similarity statistic is found for each posture[4] in the posture set, the second stage is to find the minimum value which is represented as

$$Y = \min(ss_1, ss_2, ...ss_i) \tag{5.2}$$

$$P = \begin{cases} Token & : \quad Y < \epsilon \\ Null & : \quad Y >= \epsilon \end{cases} \tag{5.3}$$

where Y equals the minimum similarity statistic value. The corresponding posture token P is then found from Y only if it is less then a threshold value ϵ otherwise no posture is recognized. Once a value for P has been determined, the third stage is to insert P into a queue which holds the last n posture tokens. The number of elements in the queue is based on informal measurements of the accuracy of the SuperGlove. In this case, we found a twenty element queue to be satisfactory for the applications we are currently developing, but this number could change for other applications. The queue is used to help alleviate false positives during recognition, and if the majority of the elements in the queue are a given posture token, then that token is sent to the integration component.

5.2.3 CyberGlove Posture and Gesture Recognizer

The CyberGlove posture and gesture recognizer has similar functionality to the SuperGlove posture recognizer except for additional gesture recognition functionality and small modifications to the posture recognition algorithm.

Posture recognition in this recognizer follows equations 5.1 thru 5.3 in the previous subsection but instead of using ten bend angles for the similarity statistic, it uses sixteen. Although the CyberGlove we use has eighteen bend sensors, posture recognition excludes the last two bend sensors which measures wrist pitch and yaw. Exclusion of wrist pitch and yaw in the recognition of postures gives the user more freedom in making them. The second modification to the algorithm is in the number of elements in the posture token queue. Based on our informal tests, the CyberGlove is more accurate than the SuperGlove and, as a result, needs a smaller queue. The number of elements in the token queue has

[4]By finding a similarity statistic for each posture in the posture set, we can sort these values and use a n-best recognition approach. The n-best approach would pass a list of posture tokens to the integration component with each token having an associated probability. These probabilities would aid in the determination of the overall task the user wanted to perform.

been reduced to ten for this recognizer which makes posture to command response time somewhat faster.

Due to the CyberGlove's improved performance over the SuperGlove, we have also included a simple gesture recognizing element to this component. The gesture recognizer looks for interesting patterns in the glove's angle measurements over time which would be difficult with the SuperGlove because it has less bend sensors. For example, we have a wrist flick gesture which can be used to delete objects in the virtual environment or change the states of the application. The wrist flick gesture is recognized if a change in the wrist yaw angle data has changed rapidly. Because both posture and gesture recognition occur in the same component, there could be a conflict with mistaking postures as part of a recognizable gesture. The posture token element queue helps to resolve this conflict by making sure that a random posture token, which may be a part of a gesture, is not sent to the integration component. As more gestures are added to the framework, we may need to modify the number of elements in the posture token queue specifically to handle posture and gesture conflicts.

5.2.4 Left and Right Hand Tracker Data Managers

The purpose of these two components is to organize the position and orientation data from the magnetic trackers so that answers to queries made by the integration component are sent back as fast as possible. Both managers have a common structure but do contain different functionality. In a similar manner to the posture recognizers described in sections 5.2.2 and 5.2.3, the tracker data managers hold a queue of the last n position and orientation records which provides them with a history of the motion of the hands[5]. Both managers use this information to communicate with the integration component.

Communication between the tracker data managers and the integration component takes place in two forms. The first form of communication allows the integration component to query the data managers for information that is useful to determine what action to take given what it already knows. Examples of the type of information requested are the current roll angle of the hand and whether one hand is closer to the display surface than the other. The second form of communication lets the tracker data managers send useful information to the integration component. The managers look for patterns in the motion history of the hand by analyzing the position and orientation data.

[5]Currently we use store the last twenty position and orientation records, but this is subject to change based on the amount of motion history we need.

The left and right tracker data managers are distinguishable by the types of information processing they do. A common action in virtual environment applications is pointing to or grabbing objects and manipulating them. Therefore, one of the tracker data managers is designed to provide information about what object has been selected for manipulation. This information is obtained by casting rays from the tracker's position into the scene and checking if any objects have been intersected. Once an object has been selected, constrained or unconstrained relative movement information can be obtained from the manager as well. The majority of the other tracker data manager's time is spent determining hand orientation which is used in conjunction with posture recognition. By knowing in what orientation the hand is, a small number of additional postures can be added to the posture set. For example, a fist posture can be turned into 3 distinct postures when the hand's orientation is at 0, 90 and 180 degrees about the z axis. Currently the left hand tracker data manager handles object selection and manipulation information while the right hand tracker data manager handles orientation information, but this configuration can be changed to accommodate user comfort.

5.2.5 Speech Token Recognizer and Parser

The speech token recognizer and parser component has two essential parts. The first part is the speech recognition engine itself which runs on either a Sun Sparc UltraII when using the rear-projected display configuration or a Sun Sparc 4 when using the surround screen configuration. These workstations act as a servers in their respective configurations and the speech itself is recognized using the BBN Hark Speech Recognizer. The recognizer is a speaker independent system based on a phonetic dictionary. Recognizable words and phrases are placed in a grammar file that is loaded into the recognizer upon start up. The recognizer has both continuous word/phrase spotting and isolated word recognition modes. We use continuous phrase spotting in both the scientific visualization application and in the Room Designer application. An advantage of the Hark system is that it allows important recognition parameters to be changed through a parameter file such as speech detection sensitivity, rejection sensitivity, and speed of the recognizer (see Figure 5.7 for an example parameter file).

Once the Hark system recognizes a word, phrase or a non-recognizable entity, the server sends a string across the network to the application. At this point, the second part of the speech token recognizer and parser takes the string, parses it and sends a one word token to the integration component. Note that one of the important goals of our project is to allow the user to speak to the computer with as much flexibility as possible. As a result, we want the user to have many different possibilities for issuing a voice command. This goal makes the parsing tedious since there are usually many possible combinations of words and phrases that can make up a single command.

5.2.6 Integration Component

The integration component combines the output from the lower level components into the framework[6]. Whenever a speech command, gesture, or posture is sent to the integration component it interprets the available data and sends an appropriate command by callback to the application. The integration component stores the current and previous posture, gesture, and speech tokens so it always has a copy of the last token received from each component which is important for determining when one series of posture tokens begins and another series ends. This component also has an individual function that gets called when a particular recognizer sends a token. This separation provides a cleaner interface.

The integration component is only one instantiated object and each of the recognizer components has a pointer to it. When a recognizer sends a token to the integration component, the specific function for that token class gets called. This function then invokes the action that the application has mapped to this token. The callback action can also check the state of the application and look at other tokens received from other components to either further refine or to reject the action.

An important part of our framework was to allow the user to issue the same command in many different ways using different modes. For example, we want the user to be able to create an object with a voice command, a hand posture, or a hand gesture and use any of these mechanisms at any time. The user may prefer a certain method of interacting with the application, but if that method fails, the user should be able to quickly and easily jump to another method for that particular command. In order to facilitate this type of interface, the integration component prints out what operation it thinks the user wants to perform. It also prints out messages when it does not understand what the user is trying to do. These messages provide the user with feedback which enables them to try the interaction again or proceed to another mode of input.

[6]The integration component currently supports complementary and equivalent styles of multimodal interaction. Incorporating redundancy into this component is an area for future work.

```
#######################################################
# Parameter file for the scientific visualization application
# Author:  Joseph LaViola
#######################################################

# Grammar file and voice model

grammar file: /map/gfx0/users/jjl/jot/jjl/grammars/vwt.bgf
voice model:  /map/gfx0/users/jjl/jot/jjl/grammars/vwt.vmd

# Params needed for doing a socket connection

Audio input channel:                    socket
Audio socket port number:               0
Produce sidetone when listening [no]:   no

# SILENCE TIMEOUT - set this to 0 so that the Recognizer will wait
# indefinitely
# for you to speak.

Maximum allowable initial silence (sec) [60.0]: 0

# DECODER SPEED - between -10.0 and 10.0, default is 0.0
#  negative numbers make the decoder go slower, but consider more
#  possibilities; positive numbers make it go faster, but consider
#  fewer possibilities.

decoder speed [0.0]: 6.0

# REJECTION SENSITIVITY - controls the ratio of false acceptance and false
#   rejection.  The value ranges from 0 (no rejection) to 100 (maximum
#   sensitivity).  Default value is set by "hcompile -r <sensitivity>".
#   If the current grammar has not been built for rejection,
#   this parameter's value will automatically be set to -1; setting this
#   parameter in this case will cause the parameter setting command to fail.

Rejection sensitivity: 50

# INTERFACE VERBOSITY - send HARK_SPEECH_START and HARK_SPEECH_END if yes

Output speech activity [no]: yes

# HEARTBEAT INTERVAL - between 0 and 10000 seconds (default is 0, no heartbeat)

Heartbeat interval (sec) [0]: 0

# ENABLE SPEECH DETECTOR - set yes (default) to search for speech

Use speech detect [yes]: yes

# SPEECH DETECT SENSITIVITY  - value 0 (insensitive) to 100 (very sensitive)
#  default is 50.  (useful range is 25-75)

Speech detect sensitivity [50]: 45
```

Figure 5.7: An example Hark parameter file.

Chapter 6

Application I - Multimodal Scientific Visualization

The first prototype application developed using the framework, described in the previous chapter, was a multimodal scientific visualization tool (MSVT) for exploring flow about a dataset (see Figure 6.1) that was based on the Virtual Windtunnel project by Bryson [20, 21]. MSVT uses the rear-projected display hardware configuration and combines speech input with pinching postures and gestures. The idea behind MSVT was not only to build a natural and intuitive interface for a scientific visualization application, but also to explore multimodal input combination types, specifically complementarity, concurrency, specialization, and transfer. Another important goal for this prototype was to determine how far we are from attaining a human-to-computer speech dialog that is equivalent to human-to-human communication using off-the-shelf technology.

6.1 Application Functionality and Interaction

MSVT gives users the ability to create, modify, drop, pick up, and delete a small set of visualization tools (streamlines, rakes, and colorplanes) for exploring the flow field about a given dataset. They also have the ability to change their viewpoint in the virtual environment, manipulate the dataset, make pictures of visualizations, and record and playback animations. Using the Fakespace Pinch Gloves, only the thumb, index, and middle finger on each hand and a set of speech commands are required to perform all the interactions in the application. The speech input component uses a vocabulary of over 20 voice commands

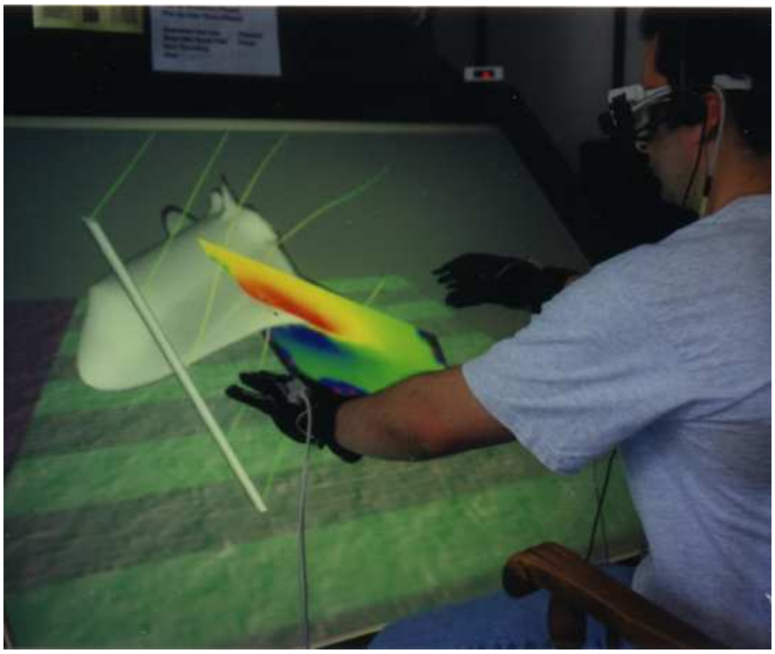

Figure 6.1: A user interacting with a dataset for visualizing a flow field around a space shuttle. The user simultaneously manipulates the streamlines with his left hand and the shuttle with his right hand while viewing the data in stereo.

shown in the grammar file in Figure 6.2. The following subsections describe the components of MSVT's interface in more detail including navigation, dataset manipulation, tool creation and manipulation, and recording and playback.

6.1.1 Navigation

Users navigate through the virtual environment with two possible interactions. Based on Multigen's SmartScene navigation techniques [87, 98], users can pull themselves through the virtual world by pinching the thumb and middle finger on either hand and grabbing a point in space. Translation is not constrained so movements in x, y, and z can be made. When the users invoke the technique with one hand after the other, they can virtually walk through the VE.

Users can also pinch the thumb and middle finger of each hand simultaneously which results in the ability to scale, rotate, and translate the virtual world in one motion. This technique can be thought of as three distinct components (see Figure 6.3). First, scaling the viewing region by moving the two hand closer or farther apart along a fixed line.

```
# This file contains a grammar for recognizing all commands
# used in the MSVT application.

<START> $main;

$main:  $create |
        $drop |
        $delete |
        $acquire |
        $remember |
        $recall |
        $start |
        $stop |
        $halt |
        $computer |
        $play |
        $quit |
        $pause;

$create: [CREATE] $object [PLEASE];

$drop: DROP $object [PLEASE];

$delete: REMOVE $object [PLEASE];

$acquire: PICK UP $object [PLEASE];

$object: RAKE | STREAMLINE | COLOR PLANE;

$remember: REMEMBER THIS VIEW;

$recall: SHOW [ME] SAVED VIEW;

$start: START RECORDING;

$stop: STOP;

$play: PLAYBACK;

$pause: PAUSE;

$halt: HALT;

$computer: COMPUTER;

$quit: QUIT APPLICATION PLEASE;
```

Figure 6.2: The grammar file which holds the speech command descriptions used in the MSVT application. Note that words in brackets are optional in issuing a given voice command.

Second, rotating the world by making arc-like motions with each hand in opposite directions or keeping one hand stationary and make arc-like motions about the stationary hand. Third, translating about the virtual world by moving both hands simultaneously keeping the distance between the hands constant throughout the operation. By combining these three components users can perform scaling, rotation, and translation in one motion. For example, moving to a specific location and facing a certain direction while zooming to a close up view of an area of interest in a dataset.

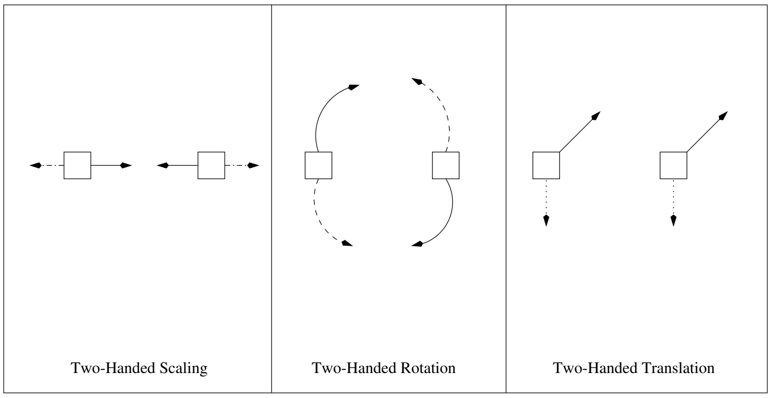

Figure 6.3: The three basic components of the two-handed navigation technique. The boxes represent the user's hands and the line styles represent possible motions. These components can be used in isolation or by combining them so the viewing region can be scaled, rotate and translated in one motion. Note that using one hand at a time also allows for translation.

Another method for navigating about the virtual environment is using the hand as a palette. In our case, the right hand acts as the palette while the left index finger is used to instantiate interactions. When users touch the left index finger to the right pinkie, middle finger, or thumb, the world rotates about the center of the dataset by 90, 180, and -90 degrees respectively. This navigation tool can be used for quickly rotating around the dataset

6.1.2 Dataset Manipulation

The task of dataset manipulation, specifically rotation and translation, is performed with the thumb and index finger on each hand. With the right hand, when users touch the thumb and index finger, they can translate and rotate the dataset by simply moving and rotating the hand. This manipulation provides six degrees of freedom. To perform rotation or translation of the dataset in isolation, and users touch the thumb and index finger of the left hand. Either dataset rotation or translation is performed based on the initial angle of the user's hand about the z axis. If the hand angle is approximately zero degrees about the z axis, (i.e. the tracker attached to the back of the hand is approximately parallel with the floor) dataset rotation is performed otherwise the dataset is only translated. Note that touching the thumb to the index finger with both hands simultaneously is analogous to the two-handed navigation technique described in the previous subsection, except scaling is omitted.

6.1.3 Tool Creation and Manipulation

MSVT provides three visualization tools; streamlines, rakes (see Figure 6.4), and colorplanes for exploring the fluid flow around the dataset. In order to create a given tool, users simply extends their arm to the display device and ask for the appropriate tool as shown in Figure 6.5. The hand that has the greatest distance from the tracking device's transmitter is the hand the object attaches to. For example, to put a colorplane in the right hand, simply extend the right arm and say "COLORPLANE". The colorplane is then instantiated and attached to the right hand where it can be moved through and around the dataset. This type of interface presents a natural way to instantiate tools by using a "show and ask" metaphor where users ask the application for some object and show where the object is to go.

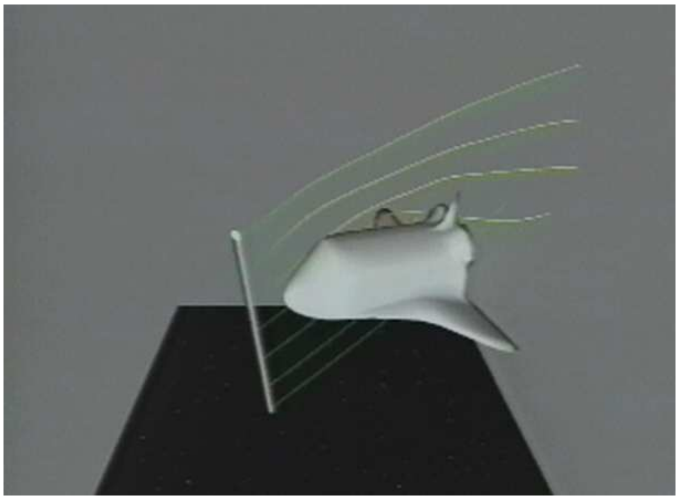

Figure 6.4: The rake visualization tool which is made is made up of a number of streamlines.

The visualization tools can be similarly dropped into the environment: users move the tool to where it should be located while uttering a "DROP *object*" command where *object* is equal to streamline, colorplane or rake. Users can then manipulate the dataset with the visualization tool fixed. If users want to pick up a fixed object they simply hold out the appropriate hand and utter the "PICK UP" command asking for a particular tool. Visualization tools can also be deleted from a hand by holding out that hand and asking the application to remove that tool with the "REMOVE" command.

Once a rake or colorplane has been created, users can change the number of streamlines

Figure 6.5: The user extends his right hand to the display asking for a streamline.

attached to the rake and the size of the colorplane. These parameter changes are made using a digital input slider, a slider which has no analog components. It simply consists of some type of button and the ability to determine the relative positions of itself and the entity that invokes the button press. The Pinch Gloves can make a very effective digital input slider since the conductive cloth patches attached to each fingertip extend down the back of the each finger as well. Therefore, the user can make a connection by sliding one fingertip along the back of another. Using the trackers attached to each hand we can then determine whether the user slides a finger along the back of another in a direction moving toward the wrist or away from the wrist. Knowing this direction, we can then increase or decrease a parameter value to one of the visualization tools. So, the number of streamlines attached to a rake can be increased or decreased with the left index finger/right index finger slider and the size of a colorplane can be increased or decreased with the left index finger/right middle finger slider. These increases and decreases are fixed in the application. Currently, two streamlines are added or removed from a rake depending on the direction of slider manipulation, and the colorplanes are increased or decreased in size by a factor of 0.5. Note that these values could be changed dynamically based on user preference if appropriate speech commands were in place.

A question arises as to how to determine what tool's parameter value to change if there is one in each hand; a rake in each hand, for example. This issue is resolved by holding the hand stationary that has the intended object of interest. With this approach, the hand that has moved the least during the course of the slider manipulation indicates which tool to modify. So, if a rake is in each hand and to increase the number of streamlines attached to the rake in the left hand by four, users can hold the left hand fixed and then slide the right index finger along the back of the left index finger, away from the wrist. Doing this twice would add four more streamlines to the rake in the left hand.

6.1.4 Recording and Playback

Having the ability to save and reinvestigate certain portions of a visualization session is an important part of using visualization techniques to better understand scientific data because it allows scientists to go back and reexamine interesting visualizations and show them to colleagues and collaborators. Therefore, MSVT provides two mechanisms for saving and retrieving visualizations. The first takes snapshots of a given scene by simply asking the application to "REMEMBER THIS VIEW". The view can then be retrieved with the "SHOW ME SAVED VIEW" command. The second mechanism records and plays back interaction animations. When the "START RECORDING" command is issued, the background color of the screen turns red, as shown in Figure 6.6, indicating that the application is in recording mode. Users then make the animation and say "STOP" to finish the recording. To view the animation users can issue the "PLAYBACK" command. During playback, the background color turns green, shown in Figure 6.7, indicating the mode change. These recording and playback tools not only benefit users who want to go back to previous visualizations, but also in collaborative settings when they needs to show colleagues important visualizations they have discovered.

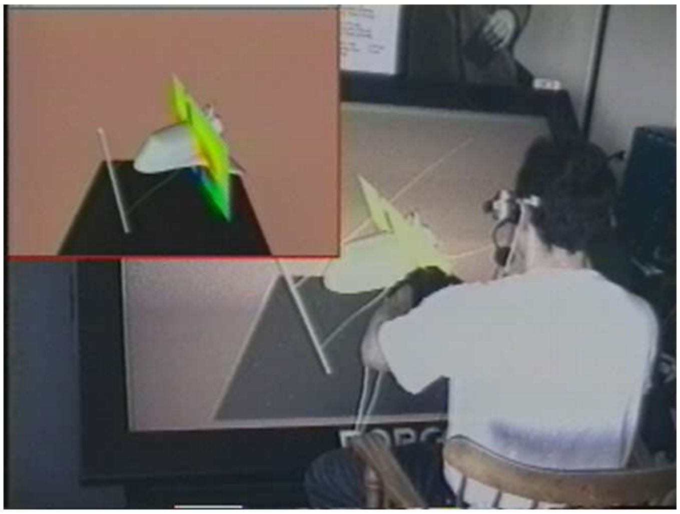

Figure 6.6: The user is in recording mode as indicated by the red background.

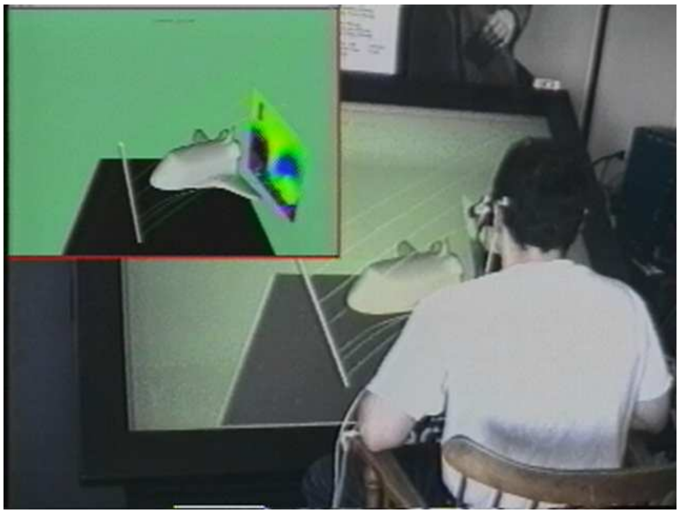

Figure 6.7: The user watching a previously recorded animation indicated by the green background.

In many cases, users will have both hands occupied interacting with the dataset but may still need to issue commands. For example, if speech were not available, when manipulating a tool in one hand and the dataset in the other, users would have to free one hand and make a gesture or press a button to start a recording. MSVT solves the problem by taking advantage of the concurrent multimodal input combination style of speaking and direct manipulation.

6.2 User and Prototype Evaluation

Throughout the life of MSVT, a number of people, from academia, industry, and government, both have tried and observed the application. In general, users found the application to be compelling in terms of the interaction and the virtual environment. From an interaction perspective, the majority of users found the interface easy to use (with some training) and liked the idea of the "show and ask" metaphor for creating and selecting visualization tools. The "show and ask" metaphor is an improvement over other object creation and selection techniques such as aperture-based selection [49] and 3D menus because these techniques require users to go to the object of choice in order to select it or to navigate through many layers of 3D menus to create an object. The "show and ask" metaphor is a faster method of creating and selecting these visualization tools since users do not have to actively select the virtual objects; with "show and ask" they come to the user via simple voice commands and are properly positioned by showing the application which hand to place the object in.

Users also found the digital sliders for increasing or decreasing the number streamlines attached to a rake and increasing or decreasing the size of a colorplane to be a simple yet effective way to manipulate these parameters. In addition, the majority of the users found the recording and playback capabilities to be an important part of the application. From the virtual environment perspective, most of the users (especially those will little or no VR experience) found the semi-immersive display to be extremely compelling from a visual standpoint. Users seemed to enjoy the stereo display and often tried to physically touch the virtual objects which is a good indicator that the stereo effect is working well.

Users provided a number of useful suggestions and constructive criticisms for improving the application. A number of users wanted other visualization tools in addition to the ones provided. Specifically, a number of people wanted the application to provide a form of text output that showed exact values in the flow field based on user input. Other tools that users thought might make the application more robust were particle traces and isosurfaces. Another important suggestion had to do with scaling with respect to interaction within the flow field. Users wanted to scale down the movement of the visualization tools so fine grained manipulation could be performed when users were close to the dataset.

Another important comment was that even though they found the application to be visually compelling, they were uncertain as to whether it would actually help them to be better scientists and to understand the data easier and more efficiently. This question not only plagues MSVT but all virtual environment-based scientific visualization systems. Unfortunately, the question is difficult to answer and is a definite area for future work and consideration.

Finally, one of the major problems with MSVT that people commented on had to do with the speech recognition. One of the goals of MSVT was to see how effective a natural speech interface without any push-to-talk mechanisms would function using current technological components. In an isolated environment (an environment where only the user of the system is present), the speech recognition worked well and there were very few problems with false positive recognition. However, in a collaborative or demonstration scenario, the speech recognition often broke down due to environmental noise, recognizing words it wasn't supposed to causing erroneous operations. In some cases, these speech recognition problems made the application unusable. The main reason these problems occurred was that the application could not distinguish between the user speaking to it and to other people in the environment. As a result, with the current state of technology, we concluded that we could not have the user effectively interact with the application at a level of communication that mimics face-to-face human conversation (see Chapter 3). Therefore, an intermediary in the form of a push-to-talk interface is required when users are in collaborative or demonstration settings.

Chapter 7

Application II - Multimodal Room Layout and Interior Design

The second prototype application developed using the framework described in Chapter 5 is a system, called Room Designer, for populating simple architectural spaces (see Figure 7.1) with furniture and other interior decorations. Room Designer uses the surround screen hardware configuration which provides the user with a fully immersive virtual environment experience. In a similar input model with MSVT, Room Designer combines speech input with pinching postures and gestures. Room Designer is loosely related to systems such as The Conceptual Design Space, which allows users to inspect, inhabit, and perform modifications on conceptual building designs [15], and the Virtual Architectural Design Tool (VADet), where users can create both architectural and interior structures in the C2 (a Cave-like device) [60]. Room Designer distinguishes itself from these systems in using more complex graphics primitives and incorporating speech input into a multimodal interface.

One of the main goals of Room Designer, besides the investigation of various multimodal input styles, is to develop a tool to let novice users prototype possible furniture layouts and interior designs in a conceptual manner, and perform these tasks with a one-to-one, life size scale. In addition, as noted in the last chapter, using a non-push-to-talk speech interface was not appropriate in collaborative or demonstration settings. As a result, another important goal of this prototype is to evaluate our efforts to improve the speech interaction with an "invisible" push-to-talk mechanism.

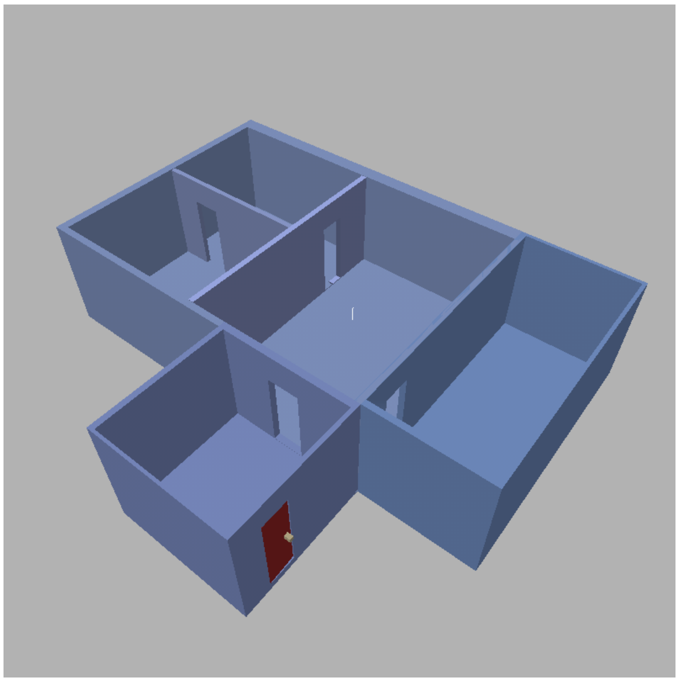

Figure 7.1: A simple conceptual model of the first floor of a house used in the Room Designer application.

7.1 Furniture Database Description

In order to provide some flexibility in making room layout and interior design decisions, we developed a database of complex graphics primitives. These primitives, represented as polygonal meshes, include various types of furniture such as chairs, tables, desks, lamps, and so on. Interior decorations such as rugs and pictures, represented as textured cubes, are also included in the database. The majority of these models and textures were found on free sites using the World Wide Web, and, in total, approximately 40 objects are present in the furniture database. These objects are presented pictorially in Figure 7.2.

From an implementation perspective, the furniture database is defined as a simple holding class. This class holds a number of furniture classes, one for each furniture type, which contain n number of furniture primitives. The furniture primitives are stored in dynamic arrays which allows for easy addition of new ones[1]. Addition of new furniture types to the database is also straightforward. A developer only has to create a new furniture class and add it to the holding class. The holding class is instantiated as a single object and during construction, it reads in all of the furniture data before the application starts which

[1] In the case of pictures and rugs, textures are stored in the dynamic arrays.

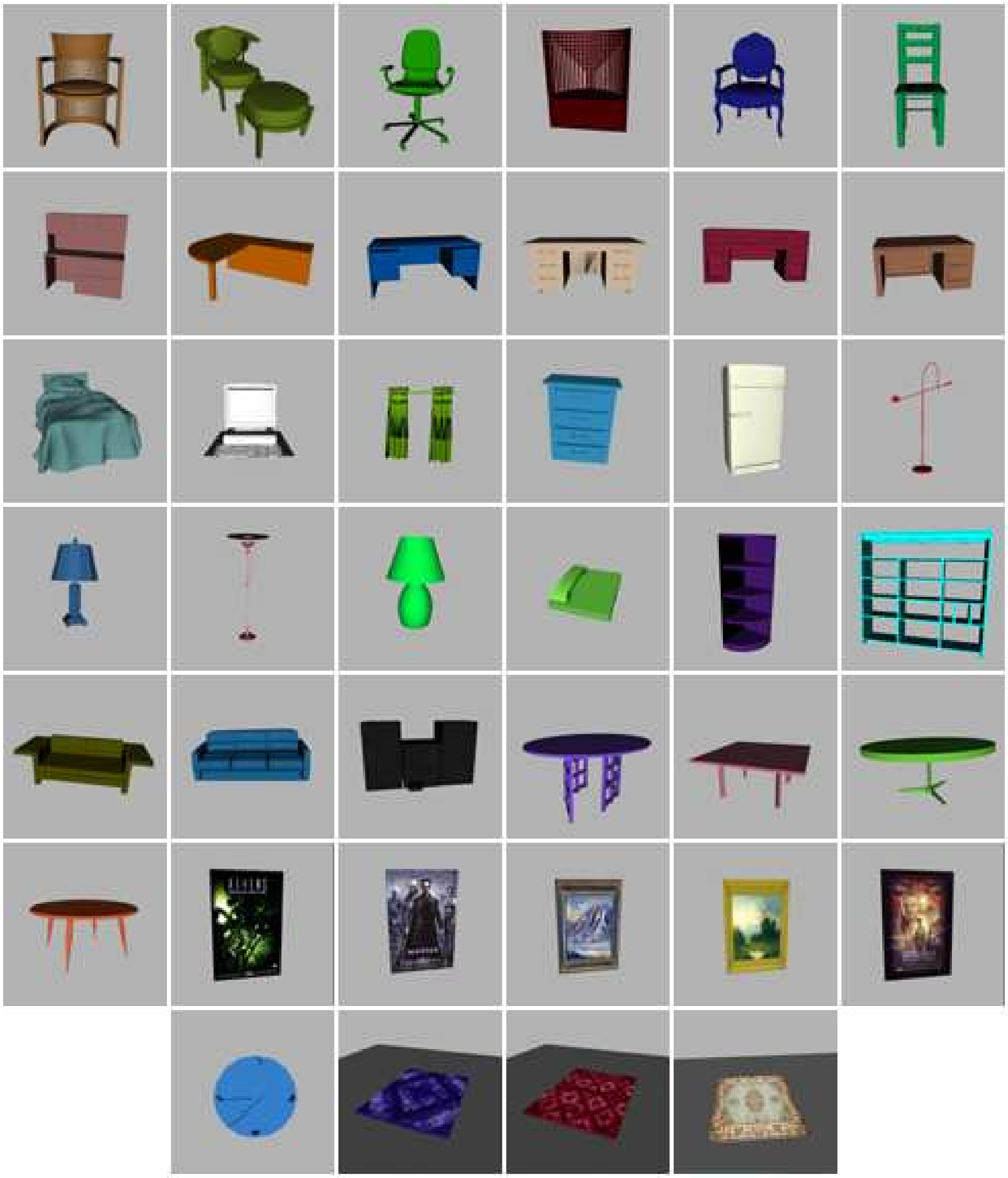

Figure 7.2: An iconic representation of 40 of the primitives present in the furniture database.

speeds up retrieval and reduces display time. Access to any particular piece of furniture is made through that furniture type's accessor method found in the holding class. These accessor methods provide a link directly to the dynamic array so objects and their initial transformations[2] can be retrieved.

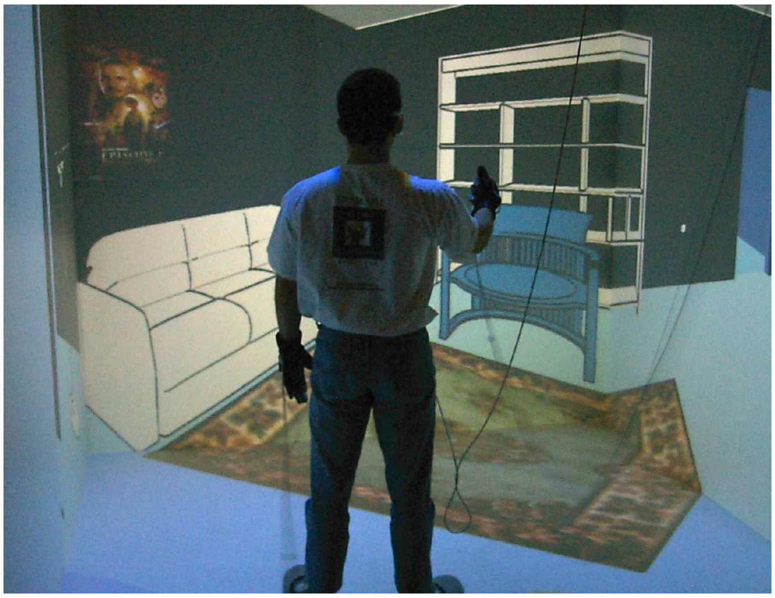

Figure 7.3: A living room constructed with Room Designer.

7.2 Application Functionality and Interaction

Room Designer allows the user to conceptually perform room layouts and interior design in an architectural space while present in that space. Some simple room layouts are shown in Figures 7.3, 7.4, and 7.5. Users create, place, modify, and manipulate furniture and interior decorations so as to quickly try our different designs that would otherwise be costly and time consuming. Users can also navigate through the space by pulling themselves through the environment. Using the Fakespace Pinch Gloves, the thumb, index, and middle finger on each hand along with the pinkie finger on the left hand and a set of speech commands are used to perform all the interactions in Room Designer. The speech input component uses a vocabulary of over 15 voice commands shown in the grammar file in Figure 7.6. The following subsections describe the components of Room Designer's interface in more detail and includes navigation, furniture and interior decoration creation, and furniture and interior decoration manipulation.

[2]All objects in the database have an initial transformation matrix which sets the size and positions the object to face the z-axis. Objects are placed in the world relative to the user upon instantiation.

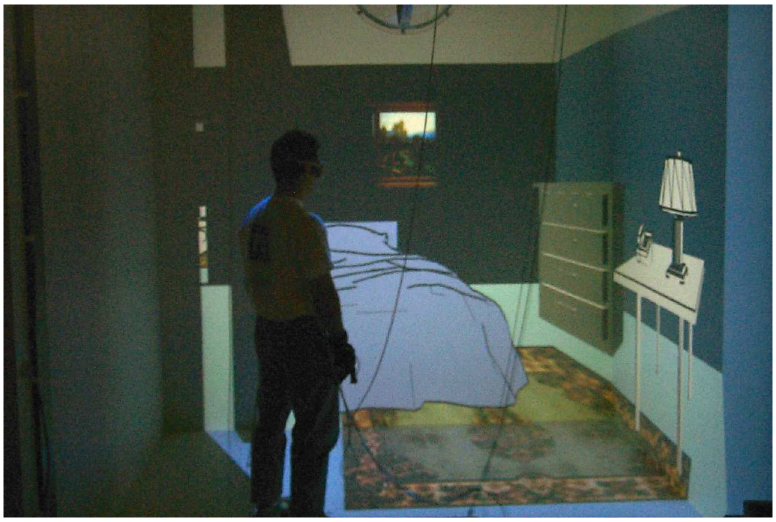

Figure 7.4: The user in a proposed layout for his bedroom.

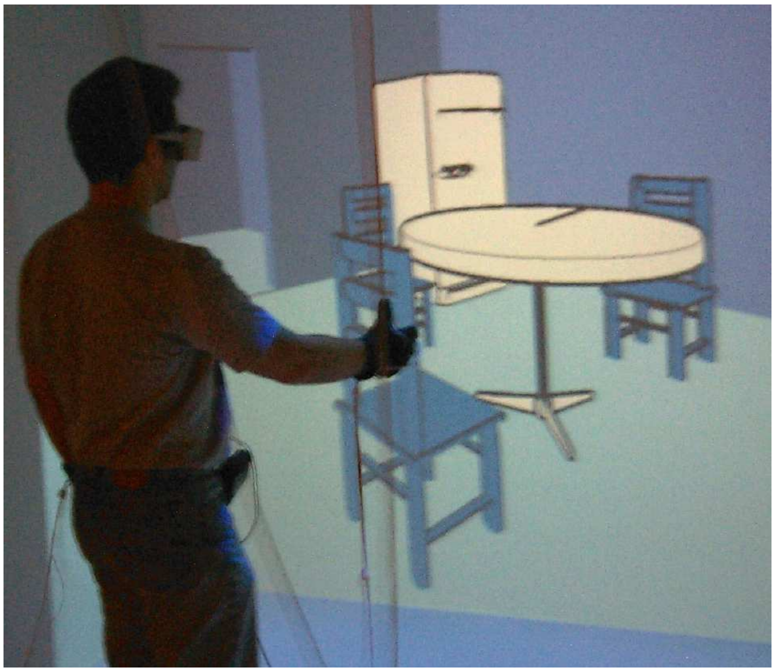

Figure 7.5: A simple kitchen created in Room Designer.

7.2.1 Navigation

Users navigate through the architectural space with a technique that is similar to one-handed navigation found in MSVT (see Chapter 6) with three exceptions. The first exception is that both translation and rotation about the user's position can be performed instead of

```
# This file contains a grammar for recognizing all commands
# used in the Room Designer application.


<START> $main;

$main:  $furniture;


$furniture : [GIVE ME A] $object [PLEASE];

$object: CHAIR |
         DESK |
         SHELF |
         LAMP |
         BED |
         CURTAIN |
         WINDOW |
         SOFA |
         TABLE |
         RUG |
         PICTURE |
         PHONE |
         DRAWER |
         COMPUTER |
         STEREO |
         FRIDGE;
```

Figure 7.6: The grammar file which holds the speech command descriptions used in the Room Designer application. Note that words in brackets are optional in issuing a given voice command.

only translation. The second exception is that translation is constrained so movement only takes place in the xz-plane since we want a walk-through style of navigation[3]. The third exception is that user movement is not based on the point in space where the hand tracker is located when the user first starts the interaction (a thumb to middle finger contact). User movement is instead based on a sphere that attaches itself to objects using a ray cast into the scene from the hand tracker. As the hand tracker's position and orientation update, a ray is cast into the scene and a sphere attaches itself to the ray intersection surface point. Users can then extend a hand and pull themselves toward the sphere. Only one motion is required no matter what the distance between the user and the sphere is because it is mapped onto the distance between the user's hand and head. Users can rotate about their position when they move their hand in a side-to-side manner.

As an example, consider a room with a chair positioned by one of the walls. If users want to move to that chair, they would simply extend their hand out to the chair. When

[3]Rotation is also constrained. Rotations are performed about the y-axis relative to the user's head position.

the sphere intersects the chair, they are notified by the creation of a bounding box around it (see Figure 7.7). They would then pinch the thumb and middle finger together on the right hand (only the right hand is used for navigation) and pull it towards them which would then translate them to the chair. If they wanted to turn around, say to go through a doorway into another room, they would make the same pinching posture but move the right hand from right to left until a 180 degree turn was made. Combinations of these forward-backward and side-to-side gestures allow users to rotate and translate simultaneously.. For example, an L-shaped gesture would move users forward and then rotate them some number of degrees. These types of gestures are good, for example, when users want to move to a door and turn to face the door at the same time.

7.2.2 Furniture and Interior Decoration Creation

Furniture and the other interior decorations are instantiated in Room Designer using a "point, speak, and cycle" interaction metaphor. When users pinch and hold the left hand's thumb and middle finger two important application states are activated. First, a virtual laser pointer is created which extends from the left index finger (the location of one of the hand trackers). Second, by creating the laser pointer the speech interface activates, thus allowing users to utter voice commands[4]. By activating the speech input in this way, we have created an invisible push-to-talk style of speech interaction because the user does not have to do anything extra to initiate the speech component. In other words, the push-to-talk part of the interface is embedded into the interaction technique for creating and placing objects in the environment, thus requiring no extra cognitive load but still providing a critical piece of functionality. In addition, the "point, speak, and cycle" metaphor requires a significantly smaller number of operations for creating furniture than traditional desktop modeling systems. For example, to create and correctly place a furniture object into a virtual room in a traditional WIMP-based modeling system, users would have to click on the "File" menu to read in a data description of the object, load the object, and then translate and rotate the object to the appropriate position. In these situations, users require several operations in order to accomplish this task. With Room Designer, users need at most two operations to complete the same task.

Using the virtual laser pointer, users point at a particular location and ask for a particular type of object, for example, pointing to the floor and saying "GIVE ME A CHAIR

[4]Analogously, when users release the thumb and middle finger, the virtual laser pointer goes away and the speech recognition deactivates by having all utterance ignored.

PLEASE." The chair then appears in the requested location facing the user. The chair that was instantiated may not be of the appropriate type. So, users can cycle through the available chairs in the furniture database by moving their left hand across their body while pinching the thumb and pinkie. A beep indicates when they have reached the end or the beginning of the list, depending on which direction the hand is moving. We use this approach over other techniques such as simply picking from a 3D menu [60] or virtual palette [28] because we do not want the user to have to continuously look from the place of object selection to the place of object instantiation. With this technique, the user can try out different types of furniture without having to visually and cognitively move away from the area of interest each time a different piece of furniture is created. Note that pictures and rugs are created in the same manner.

7.2.3 Furniture and Interior Decoration Manipulation

Already placed furniture or interior decorations can be moved about the architectural space with simple direct manipulation. Using either the left or right hand, users can grab an existing object by moving a hand to the approximate location of the object. When the hand-tracked sphere, used to provide feedback on where the user's hands are in the virtual environment is in close proximity to the given object, it's bounding box will highlight indicating that the object can be picked up (see Figure 7.7). Using a thumb to index finger pinching posture, users can pick up the object and transfer it to a different location. To move a bed into another room of a house, they could simply grab and hold onto the bed in the left hand while navigating to the other room with the right hand and then placing the bed in an appropriate location. In addition, to make things easier, furniture and the interior decorations have built in constraints attached to them. For example, chairs, tables, and desks, are constrained so that they only move in the xz-plane or only along the floor and pictures can only move in along a given wall.

Users can also delete objects in the environment. Our deletion technique is taken from Mine's proprioceptive work [95]. To delete an object, users grab and hold the object and then throw it over their shoulder (see Figure 7.8). The over-the-shoulder deletion technique has a number of advantages in that it is easy to remember and not easily invoked so unintended deletions do not normally occur.

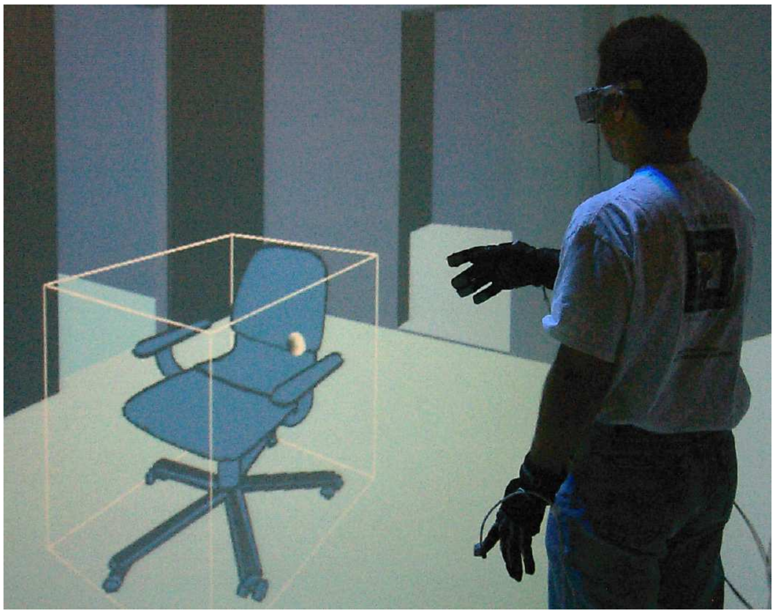

Figure 7.7: The chair's bounding box is highlighted indicating the virtual sphere has intersected it. A user can pick up the chair or move towards it.

7.3 User and Prototype Evaluation

A number of users have had a chance to try and observe Room Designer in the course of the many demonstrations that have been given at the Brown University Technology Center for Advanced Scientific Computing and Visualization[5]. On the whole, users have found the application to be a compelling virtual environment experience. They found the navigation technique allows for a much more active role in their movement over other traditional techniques such as using pointing gestures to indicate direction and speed of movement. They found the ability to point and speak to create and place the furniture simple to use and easy to remember. They found the cycling part of the interface to be somewhat difficult to use, especially for objects such as chairs which had more than a handful of different types, mainly due to the fact that they would sometimes find an object they wanted but, when they released their thumb and pinkie finger, the next object in the list would appear due to extra hand movement. Also they found it difficult to know how far they had to move their hand[6] to have another object appear. These problems are probably due to the implementation of the technique and not the technique itself. They present an important area for future work (see Chapter 9).

[5] This center is where the TAN VR-CUBE is located.

[6] In this case, haptics would be a viable solution.

Figure 7.8: A user deletes a chair by throwing it over his shoulder.

A few users, mostly from industry, commented that, with further refinement and extensions such as a larger database and more realism and detail, Room Designer could have commercial utility. It could be used in furniture show rooms so customers could plan out furniture arrangements in representations of their own homes. By having the ability to try out different designs, show room customers would be confident in their furniture selections because they would see the arrangements as if the furniture was physically in their home. In addition, the Room Designer application could be a very useful tool in estimating the costs of putting furniture and interior decorations into various architectural spaces.

Finally, only a few users commented on the speech recognition which indicates that the "invisible" push-to-talk worked well. Of the few comments that were made, most of them were in regards to simple misrecognitions. In general, our push-to-talk approach was able to solve the external noise problem since the user operating the system could talk freely to collaborators without having the application recognize utterances. Although the push-to-talk interface worked well while not increasing the cognitive load of the user, there were still some problems that occurred. The push-to-talk interface alleviates problems with external noise when the push part of the interface is not in the active state. However, when in the active state, the application is susceptible to the same external noises that a non-push-to-talk interface is. Even through the push-to-talk interface filters out about 90% of the external noise, the remaining 10% can still be problematic. This 10% represents an important area for future work.

Chapter 8

Interface Guidelines and Interaction Techniques

Based on the research described in the previous chapters, we summarize the interface guidelines and interaction techniques that have been developed for whole-hand and speech input in VEs.

8.1 Interface Guidelines

This section presents a list of interface guidelines which pertain to whole-hand and speech input in isolation and when used in multimodal interfaces. Each guideline has corresponding references back to the previous text showing where they were derived.

- Combining topological and geometrical hand data (i.e. Flex and Pinch Input) provides a number of advantages to designing and developing virtual environment interaction techniques which include the ability to quickly prototype different hand postures and gestures, providing a simple starting and stopping mechanism for them, and helping to simplify posture and gesture recognition algorithms (see Section 2.5).

- Using the backside of a finger when wearing Pinch Gloves allows for the implementation of simple digital sliders which can be used for increasing or decreasing parameters by constant factors (see page 45).

- With current speech recognition technology and work environments a push-to-talk interface is essential and, in order to minimize cognitive load, it is important to embed the push part of the push-to-talk interface into existing interaction techniques if at all possible. "Invisible" push-to-talk interaction does not increase cognitive load of the user and helps to filter out the majority of external noise (see pages 20-21 and Section 7.2.2).

- When designing speech commands for a VE application try to provide as many options for issuing a single command as possible since it gives the user more choices in the way they spontaneously utter commands. In other words, design a complete vocabulary for the task at hand (see Section 5.2.5 and pages 42 and 54).

- Although redundancy as a combination style[1] in multimodal interfaces is important and useful, other combination styles such as complementarity, equivalence, concurrency, and specialization should not be overlooked because they provide many advantages in VE interface design (see Sections 6.1 and 7.2).

- Using whole-hand and speech input in a complementary fashion simplifies both user interaction and interaction technique development because one modality does not have to account for the whole interaction in the former and the implementation can be simplified in the latter (see Sections 5.2.6, 6.1.3, and 7.2.2).

- As stated in [104], providing the user with more than one modality to issue a single command, as the multimodal equivalence combination style suggests, gives the user freedom to chose how to interact with the application (see Section 4.1).

- When both hands are occupied with a given interaction, speech input provides a natural concurrent mechanism for issuing commands to the application without having users stop what they are doing (see Section 6.1.4).

- The "show and ask" interaction metaphor, similar to "put that there" [13], provides users with a natural method of instantiating objects in a virtual environment because they simply have to "ask" for what object they want and "show" the application where to put it in the course of a simultaneous interaction (see sections 6.2 and 7.2.2).

- For many virtual environment applications, only grabbing, pinching, and pointing hand gestures are needed when a speech input component is present in the interface (see Chapters 6 and 7).

[1] Chapter 4 describes the different multimodal combination styles in detail.

8.2 Summary of Interaction Techniques

This section presents a summary of the interface techniques that were developed or augmented in this work. As in the previous section, references back to the text are provided so the reader can go back and get more detail.

- Using Flex and Pinch input, we augmented the image plane object selection techniques by allowing users to start and stop the interaction with the primary hand; giving them a choice of where to place the cloth contacts (see Section 2.5).
- When using bend-sensing gloves as analog sliders, Flex and Pinch input allows users to stop and start the slider easily and place the cloth contacts for doing so where they are most comfortable (see Section 2.5).
- The hand palette technique lets users quickly navigate around a dataset of interest (see Section 6.1.1) This technique could be extended to other types of interaction such as object creation and parameter adjustment.
- Depending on hand orientation, we can have specify up to three different commands using the same hand gesture. For example, using a thumb to index finger pinch allows users to either translate or rotate an object in the VE based on the orientation of the hand (see Section 6.1.2).
- The "show and ask" interaction technique allows users to "ask" the computer for a particular object and "show" the computer in what hand it is supposed to go (see Section 6.1.3).
- When using Pinch Gloves and hand trackers, we can create digital sliders used in parameter adjustment (see Section 6.1.3).
- The "point, speak, and cycle" interaction technique lets users "point" to where they want to place an object, "speak" for the type of object, and then "cycle" through a menu of choices by moving a hand across their body (see Section 7.2.2).

Chapter 9

Conclusions and Future Work

In this thesis, we have investigated the use of whole-hand and speech input in virtual environments by exploring the issues and methods of using them both in isolation and in combination forming multimodal interfaces. By identifying the problems and research issues in these areas, a number of solutions and interface guidelines have been established.

We developed Flex and Pinch, an input system based on seamlessly combining geometrical and topological hand data which has led to the improvement of a number of interaction techniques including the image plane object selection techniques [111]. We also examined the problems with using speech input technology in VEs, presented solutions to these problems, and analyzed the tradeoffs between them. In addition, we discussed the various multimodal combination styles such as redundancy, equivalence, and transfer, examined how they can be used in VEs, and presented some of the many advantages of using multimodal interfaces.

A software framework was also developed on top of the Jot interface library [46, 80] which provided a tool for creating two application prototypes, one running on a rear-projected desk display and one running in a TAN VR-CUBE, a Cave-like display device. These prototypes (MSVT and Room Designer) provided a means for testing, evaluating, and refining our solutions, interaction techniques, and interface guidelines.

Although the work presented in this thesis has formulated some solutions to a handful of interface problems involving whole-hand and speech input, there remains a large amount of work that needs to be done in whole-hand input, speech recognition technology, and multimodal interfaces in the context of virtual environments.

A number of problems exist with Flex and Pinch input that need to be addressed. Specifically, the hardware prototype is rather difficult and time consuming to put on and is rather fragile. As a result, a better design and implementation of the worn device is required. The techniques that have been developed using Flex and Pinch also need to be rigorously evaluated.

Problems still exist with speech recognition technology and an important area that needs to be addressed is getting the computer to know when a human is speaking to it and when a human is not. This issue is probably the single most important problem that needs to be solved in order to have truly robust speech input. Quantifying what users will tolerate in terms of latency and accuracy is also an interesting area of research because it lets us know where we need to be to make speech recognition engines good enough for everyday use.

As stated previously, redundancy is an important multimodal combination style yet the focus of this work was on the others presented in Chapter 4. As a result, an area that needs to be addressed is incorporating redundancy in the software framework and in our applications. Another interesting area of work is to quantitatively determine what types of combination styles users prefer and under what conditions they prefer them.

Finally, since MSVT and Room Designer are only prototypes, a large amount of work remains to make these applications rich and robust virtual environment applications. With MSVT, more visualization tools need to be added along with a push-to-talk mechanism that is similar to the one found in Room Designer. Also, user studies need to be conducted to determine whether scientific visualization in a virtual environment is actually better than with other mediums. The pivotal question here is can an application like MSVT actually help scientists do better science.

With Room Designer, a lot of work needs to be done in order to make the application more realistic in terms of having the architectural spaces and the furniture look better by adding textures, for example. The furniture database also need to be expanded so user's have more of a variety in what they can choose to place in the environment. Increasing the database size also has implications towards the instantiation of objects. Hand cycling only works well for a handful of objects so new techniques will have to be developed to handle large quantities of furniture types[1]. Issues with vocabulary will also have to be looked at since an experienced interior designer may have the vocabulary to name all the different furniture types allowing for an increased and more complicated speech command set. In contrast, a novice to interior design would not have such a rich vocabulary thus requiring a different method for creating the many different types of furniture that are possible.

[1] Extensions to the various types of marking menus maybe appropriate for this task.

Appendix A

Introduction to Hand Posture and Gesture Recognition

This appendix provides an introduction to the concepts and criteria used in describing hand postures and gestures. It acts as a starting point for Appendices B and C, a culmination of two years of research in whole hand input, which surveys the technology and techniques associated with hand posture and gesture recognition. The work surveys a majority of the problems and issues relevant to using hand postures and gestures in user interfaces, consolidating existing information in the field and organizing it in a clear and efficient manner. It also gives a critical review of the information presented so as to point out the general advantages and disadvantages of the various recognition techniques and systems. Appendix B discusses the various aspects of hand posture and gesture recognition technology by discussing a number of current glove-based input devices and aspects of vision-based recognition systems. Appendix C describes the various feature extraction and classification algorithms used in hand posture and gesture recognition, and discusses the advantages and disadvantages of each. Note that unabridged versions of these appendices can be found in [78, 79].

Hand posture and gesture recognition can be conceptually regarded as a pipeline of processing stages. At the beginning is the physical world, the actual positions and motions of the user's hand. Some kind of hardware interface is then used to transduce this information into a form usable by the computer, and features are extracted for later processing. There may be multiple feature extraction stages, and the hardware itself may perform some element of feature extraction, such as reporting joint bend angles. The features are then passed to a classifier which does the actual recognition itself, determining which posture or gesture the user is actually making. Finally, the results of this classification are used by an application of some kind. In our conception, the distinction between feature extractors and classifiers is that classifiers necessarily produce a discrete, one-dimensional output (which gesture or posture has been recognized, possibly annotated with confidence information or the like), while feature extractors generally produce multidimensional output. Thus, multiple classifiers in parallel could be used as a feature extractor if their output was collected together and processed by a later stage. As it turns out, however, in actual practice so far the division has been sharp, and we are not aware of any algorithm having been used both as producing the final output to the application and as producing input to a later processing stage.

Although hand postures and gestures are often considered identical, there is a distinction to be made between them: a gesture incorporates motion while a posture does not. For example, making a fist and holding it in a certain position is considered a posture. According to Sturman [133], the majority of useful postures have each of the finger joints either fully extended or fully flexed. Therefore, with a *simple posture*, each finger is either fully extended or flexed but not in between; examples include the fist and pointing. With a *complex posture*, the fingers can be at positions other than full flexion or full extension. Complex postures include various forms of pinching, the "okay" sign and many of the postures used in finger spelling [99].

A gesture is a dynamic movement, such as waving good-bye. Certain gestures involve significantly fewer active degrees of freedom than others and can therefore be simpler to recognize. We distinguish two types of gestures based on this: *simple gestures* involve changes in no more than six degrees of freedom, while *complex gestures* can involve more complicated motions. Thus, holding a posture while moving the hand (changing only its position and orientation) would be considered a simple gesture, as would moving the index finger back and forth to beckon someone closer. In contrast, many of the signs in American Sign Language involve more complex finger and wrist movements involving more degrees of freedom and would be considered complex gestures.

An important criterion in evaluating these kinds of interfaces is the number of postures and gestures that a given recognition system or algorithmic technique can recognize. Based on the visually apparent clustering of existing systems shown in Figure C.1, we consider 1 to 20 postures and gestures as a small set, 20 to 40 as medium-sized, and anything over 40 as a large set. Another important criterion is accuracy. Unfortunately, accuracy is difficult to define in terms of either correctly or incorrectly recognizing a particular posture or gesture. Important considerations when dealing with accuracy issues are false positive recognition, false negative recognition, and replacement recognition. False positive recognition, also known as an insertion error, is when a system recognizes a posture or gesture when one *is not* present (generally only applicable to continuous recognition systems). False negative recognition, also known as a deletion error, is when a system does not recognize a particular posture or gesture when one *is* present. Replacement recognition occurs when a system recognizes a particular posture or gesture as a different one. Since the relative consequences associated with these different errors can vary with the application, all three types of errors should be reported as part of an accuracy metric. Unfortunately, only Starner [129] and Birk and Moeslund [12] report that they have done so.

Appendix B

Hand Posture and Gesture Recognition Technology

This appendix discusses the requirements for hand posture and gesture recognition. It describes the two main solutions for collecting the required data to perform recognition, the glove-based solution and the camera- or vision-based solution, and looks at the advantages and disadvantages of each.

B.1 Data Collection for Hand Postures and Gestures

The first step in using hand posture and gestures in computer applications is gathering raw data. This raw data is then analyzed by using various recognition algorithms (see Appendix C) to extract meaning or context from the data in order to perform tasks in the application. Raw data is collected in two ways. The first is to use input devices worn by the user. This setup usually consists of one or two instrumented gloves that measure the various joint angles of the hand and a six degree of freedom (6 DOF) tracking device that gathers hand position and orientation data. The second way to collect raw hand data is to use a computer-vision-based approach by which one or more cameras collect images of the user's hands. The cameras grab an arbitrary number of images per second and send them to image processing routines to perform posture and gesture recognition as well as 3D triangulation to find the hands' position in space. In addition, the combination of the previous two methods in a hybrid approach can be used to collect raw data with the hope of achieving a more accurate level of recognition by using the two data streams to reduce each other's error. Very little work has been done on hybrid tracking for hand posture and gesture recognition, but this type of tracking has been successful in augmented reality systems like Auer [6] and State [130], and could well be applied to hand posture and gesture recognition.

B.2 Data Collection Using Trackers and Instrumented Gloves

Raw data collection using instrumented gloves and trackers requires users to physically attach computer input devices to their hands. The instrumented gloves report data values for the movement of the fingers. The trackers are attached to the back of the hand or the upper wrist, depending on the type of glove worn, and give back data on the position and orientation of the hand as a whole in 3D space.

B.2.1 Tracking Devices

A number of different tracking technologies are available to track hand position and orientation. This survey touches on the most common; for a detailed discussion see Youngblut's [153] review of virtual environment interface technology, Encarnação's [40] survey on input technology or Mulder's [97] survey on human movement technology. These three papers present a very thorough analysis of over 25 different tracking devices on the market today.

Three non-vision-based methods for tracking hand position and orientation are magnetic, acoustic, and inertial tracking. With magnetic tracking, a transmitting device emits a low-frequency magnetic field from which a small sensor, the receiver, determines its position and orientation relative to a magnetic source. The advantages of these types of systems are that they have good range, anywhere from fifteen to thirty feet away, are generally accurate to within 0.1 inches in position and 0.1 degrees in orientation, and are moderately priced [5, 112]. Their main disadvantage is that any ferromagnetic or conductive objects present in the room with the transmitter will distort the magnetic field reducing the accuracy. The distortion can be handled with filtering algorithms, but doing so introduces a more complex computational component which might increase latency. The two most commonly used magnetic trackers today are from Polhemus and Ascension Technology Corporation.

Acoustic tracking systems or ultrasonic tracking uses high-frequency sound emitted from a source component that is placed on the hand or area to be tracked. Microphones placed in the environment receive ultrasonic pings from the source components to determine their location and orientation [131]. In most cases, the microphones are placed in a triangular array and this region determines the area of tracked space. The advantages of acoustic tracking systems are that they are relatively inexpensive and lightweight. However, these devices have a short range and their accuracy suffers if acoustically reflective surfaces are present in the room. Another disadvantage of acoustic tracking is that external noises such as jingling keys or a ringing phone can cause interference in the tracking signal and thus significantly reduce accuracy. Logitech's acoustic tracking systems seem to be the most commonly used; however, some newer companies like Freepoint 3D have entered this field [40]. Acoustic tracking has also been incorporated into some glove-based devices such as the Mattel Power Glove [131] and VPL's Z-Glove [155], discussed in further detail in Section B.2.2.

Finally, inertial tracking systems use a variety of inertial measurement devices such as gyroscopes, servo-accelerometers, and even micromachined quartz tuning forks that sense angular velocity using the Coriolis principle [40]. The advantages of an inertial tracking system is speed, accuracy and range, but the major problems with these systems are that they usually only track three degrees of freedom (either position or orientation data) and they suffer from gyroscopic drift. The most commonly used inertial tracking systems are InterSense's IS-300 and IS-600. The IS-300 measures only orientation data but uses gravitometer and compass measurements to prevent accumulation of gyroscopic drift and employs a motion prediction mechanism that predicts motion up to 50 milliseconds in the future. The IS-600 tracks both position and orientation using a hybrid approach with the inertial component tracking orientation data and an ultrasonic component tracking position data [62].

A common problem with these tracking devices is that they do not have perfect accuracy. A promising way of achieving greater accuracy is to use prediction/correction techniques to filter the position and orientation data. One of the most widely used filtering techniques is the Kalman filter, a recursive mathematical procedure that uses the predictor/corrector mechanism for least-squares estimation for linear systems. Welch and Bishop [147] and Maybeck [93] both provide detailed discussions and mathematical derivations of the Kalman filter for the interested reader. Kalman filtering can be applied to tracking devices, vision tracking [9], and hybrid tracking systems as well [146].

B.2.2 Instrumented Gloves

Instrumented gloves measure finger movement through various kinds of sensor technology[1]. These sensors are embedded in a glove or placed on it, usually on the back of the hand.

[1] The exception to this definition is the Fakespace Pinch Glove and the proximity sensors on the Digital Data Entry Glove. Instead of measuring finger movement, they detect electrical contact made when the fingertips touch.

Glove-based input devices can be broadly divided into those gloves that are available in the marketplace today and those that are not, either because their respective companies have gone out of business or because they were never developed commercially. Both Sturman [133] and Kadous [65] discuss both categories of gloves, but their surveys are now out of date. Encarnação [40] and Youngblut's [153] discussions of glove input devices deal specifically with those currently available from commercial vendors. The present survey gives both a historical perspective on these devices by describing those gloves that are no longer available and a practical guide to those gloves that are on the market today.

Historical Perspectives One of the first instrumented gloves described in the literature was the 'Sayre Glove' developed by Thomas Defanti and Daniel Sandin in a 1977 project for the National Endowment of the Arts [34]. This glove used light-based sensors with flexible tubes with a light source at one end and a photocell at the other. As the fingers were bent, the amount of light that hit the photocells varied thus providing a measure of finger flexion. The glove could measure the metacarpophalangeal joints of the four fingers and thumb along with the proximal interphalangeal joints of the index and middle fingers, for a total of 7 DOF (Figure B.1 shows a diagram of the joints of the hand). It was designed for multidimensional control of sliders and other 2D widgets and did not have the sophistication or accuracy needed for hand posture or gesture recognition.

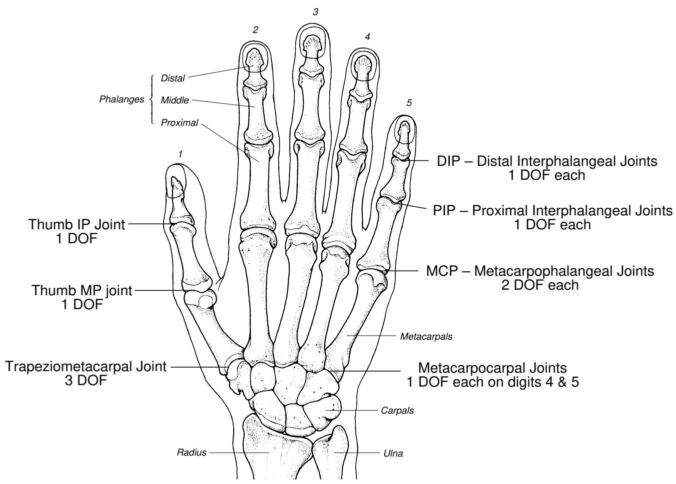

Figure B.1: The 17 joints in the hand and the associated 23 degrees of freedom (from Sturman [133]).

The Digital Data Entry Glove, designed by Gary Grimes at Bell Telephone Laboratories in 1981, was invented specifically for performing manual data entry using the Single-Hand

Manual Alphabet [54]. It used touch or proximity sensors, “knuckle-bend sensors”, tilt sensors, and inertial sensors to replace a traditional keyboard. The touch or proximity sensors determined whether the user's thumb was touching another part of the hand or fingers. They were made of silver-filled conductive rubber pads that sent an electrical signal when they made contact. The four knuckle-bend sensors measured the flexion of the joints in the thumb, index finger, and pinkie finger. The two tilt sensors measured the tilt of the hand in the horizontal plane, and the two inertial sensors measured the twisting of the forearm and the flexing of the wrist. The drawback of this glove was that it was developed for a specific task and the recognition of hand signs was done strictly in hardware. Therefore, it was not generic enough to perform robust hand posture or gesture recognition in any application other than entry of ASCII characters.

The DataGlove and Z-Glove, developed by VPL Research, were first presented at the Human Factors in Computing Systems and Graphics Interface conference in 1987 [155]. Both gloves were designed to be general-purpose interface devices for applications that required direct object manipulation with the hand, finger spelling, evaluation of hand impairment, and the like. Both gloves came equipped with five to fifteen sensors (usually ten) that measured the flexion of both the metacarpophalangeal joints and proximal interphalangeal joints of the four fingers and thumb for a total of 10 DOF. In some cases abduction sensors were used to measure angles between adjacent fingers. Both gloves used optical goniometer sensor technology patented by Zimmerman in 1985. These sensors were made up of flexible tubes with a reflective interior wall, a light source at one end and a photosensitive detector at the other that detected both direct light rays and reflected light rays. Depending on the bending of the tubes, the detector would change its electrical resistance as a function of light intensity [156]. The gloves also provided tactile feedback by putting piezoceramic benders underneath each finger which produced a tingling or numbing sensation. The main difference between the DataGlove and the Z-Glove was the position and orientation mechanisms used with each. A traditional magnetic tracking system was used with the DataGlove, while the Z-Glove had an embedded ultrasonic tracking system that placed two ultrasonic transducers on opposite sides of the metacarpals to measure the roll and yaw of the hand. Generally the Z-Glove was much more limited in application and as a result was less costly.

The DataGlove and Z-Glove were designed as general-purpose interface devices. However, their lack of accuracy limited their utility: formal testing revealed the accuracy of the sensors as no better than five to ten degrees of joint rotation [152]. The gloves could have been used for simple posture recognition and object manipulation, but they were generally not accurate enough for complex posture or gesture recognition.

The Dexterous HandMaster (DHM), first developed in 1987 was an exoskeleton that fit over the hand. Initially it was used as a master controller for the Utah/MIT Dexterous Hand, a four-digit robot hand [88]. A second version of the device later developed and marketed by Marcus [133] used a total of 20 Hall-Effect sensors as potentiometers that measured the flexion of all three joints in each finger, abduction/adduction between each finger, and four degrees of freedom for the thumb. These sensors were sampled at 200 Hz with eight bit accuracy. It was very accurate[2], with a 92 to 98 percent correlation between finger position and DHM readout [85], thus it could have been used for complex posture and gesture recognition, but it took some time to take on and off and was not suited for rapid movements because of its instability when worn.

The Power Glove was developed in 1989 by Mattel as an input device for Nintendo games and, when suitably reverse-engineered for a computer's serial port [39], became a low-cost alternative for researchers in virtual reality and hand posture and gesture recognition [65, 109]. The glove used resistive ink sensors that measured the overall flexion of the thumb, index, middle, and ring fingers for a total of four DOF. It also used ultrasonic tracking to track the hand's x, y, and z position and roll orientation of the wrist relative to a companion unit attached to the display. Because the finger sensors used two bits of precision, the Power Glove was not very accurate and useful only for a small set of simple hand postures and gestures; its big advantage was its extremely low cost.

Finally, the Space Glove, developed by W Industries in 1991, was unique in that the user placed his fingers and thumb through plastic rings that sat between the proximal interphalangeals and the metacarpophalangeal joints. The glove used sensors with twelve bit analog-to-digital converters that measured the flexion of the metacarpophalangeal joints and the interphalangeal joint of the thumb for a total of six DOF [131]. According to Sturman's personal experience [133], the Space Glove was fairly responsive but uncomfortable to wear due to the inflexibility of the plastic rings around the fingers. The glove worked only with W Industries products and, as a result, little if any work has been done with it.

Current Glove-Based Input Devices One of the least expensive gloves on the market today is the 5DT Data Glove (see Figure B.2) developed by Fifth Dimension Technologies. This glove uses five fiber optic sensors to measure the overall flexion of each of the four fingers and the thumb; according to the specifications [45], these sensors can be sampled at 200 Hz with eight bits of precision. In addition, the glove uses two tilt sensors to measure the pitch and roll of the hand. The device is currently priced at $495 for a right-handed

[2]The device was designed mainly for clinical analysis of hand impairment and robot control.

glove and $535 for a left-handed glove. Since this glove senses only the average flexion of the four fingers and the thumb, it is not suited for complex gesture or posture recognition. However, it does perform well enough for simple postures, such as pointing or making a fist, and is the supported device for General Reality Company's GloveGRASP software toolkit for hand posture recognition [52].

Figure B.2: The 5DT Data Glove developed by Fifth Dimension Technologies. The glove measures seven DOF (from Fifth Dimension Technologies [45]).

Figure B.3: Nissho Electronic's SuperGlove input device worn by the author. This glove has a minimum of 10 bend sensors and a maximum of 16.

The SuperGlove (see Figure B.3), developed by Nissho Electronics, has a minimum of 10 and a maximum of 16 bend sensors that use a special resistive ink applied to flexible boards sewn into the glove [103]. With its minimal and standard configuration, the SuperGlove measures flexion of both the metacarpophalangeal and proximal interphalangeal joints for all four fingers and the thumb. The glove comes in two different sizes and is available for both the left and right hand. A unique feature of this device is its embedded calibration procedure: three buttons on the control unit can collect data for three distinct postures to allow hardware calibration. The standard version of the SuperGlove is currently priced at around $5000. A wireless option is also available which increases the price to over $20,000; there are currently no distributors that sell the device in the United States.

From the first author's personal experience, the SuperGlove is adequate for simple posture recognition. The glove's hardware-based calibration mechanism is important and does not have to be used often, but it does not remove the need for software calibration. The glove is fairly accurate but not suited for complex gesture recognition. Unfortunately, no formal studies have been performed on the accuracy of the SuperGlove and it is not commonly discussed in the literature.

Figure B.4: Fakespace's Pinch Glove input devices worn by the author. The gloves have electrical contact points that allow users to make "pinch" postures that can be then mapped to a variety of tasks.

Pinch Gloves (see Figure B.4) take a different approach to posture recognition [42]. These gloves, originally called Chord Gloves, were prototyped by Mapes at the University of Central Florida [87]; the technology was bought by Fakespace Inc. and now the gloves are sold commercially under the Pinch Glove name. Instead of using bend sensor technology to record joint angles, Pinch Gloves have electrical contacts (similar to the Digital Data Entry

Glove's proximity sensors) on the inside of the tips of the four fingers and the thumb. Users can make a variety of postures by completing a conductive path when two or more of the electrical contacts meet. According to Encarnação [40], over 1000 postures are theoretically possible. Usually two gloves are worn to maximize the number of postures available; they are sold in pairs and are priced at $2000/pair.

The Pinch Glove system is excellent for restricted posture recognition because no elaborate posture recognition techniques are required (see Appendix C). The electrical contacts on the gloves make it easy to map postures to a variety of tasks. Since the gloves have a mount for a spatial tracking device such as a Polhemus, simple gestures can also be recognized. The drawbacks of these gloves arise from the fact that they do not use bend sensor technology. It is very difficult to provide a virtual representation of the user's hands, and such a representation is often considered important in virtual environments, although Multigen's SmartScene has gotten around this by using simple 3D cursors like spheres instead of a virtual hand [98]. Another drawback of Pinch Gloves is that the types of postures are limited since electrical contacts must be touching before a posture can be recognized. If the user makes a posture in which none of the electrical contacts create a conductive path, the posture goes unrecognized. This type of problem does not occur with a bend-sensor-based glove.

The final glove-based input device discussed here is Virtual Technologies' CyberGlove (see Figure B.6), originally developed by Kramer in his work on The "Talking Glove" [71]. Using his patented strain gauge bend sensor technology [70], he started Virtual Technologies and now sells the glove commercially. The CyberGlove can be equipped with either 18 or 22 bend sensors. With 18 sensors, the CyberGlove measures the flexion of the proximal interphalangeal and the metacarpophalangeal joints of the four fingers and the thumb, the abduction/adduction angles between the fingers, radial and palmer abduction, wrist roll, and wrist pitch [142] (Figure B.5 illustrates the various motions the hand can make). The additional four sensors in the 22 sensor model measure the flexion of the distal interphalangeal joints in the four fingers. With a six DOF tracker and the 22 sensor model, 28 degrees of freedom of the hand can be realized.

An interesting feature of the CyberGlove's interface unit is that it digitizes the voltage output of each sensor and then modifies the value using a linear calibration function. This function uses gain and offset values to represent the slope and y-intercept of the linear equation. This equation allows software calibration of the glove and thus makes it more robust for a variety of hand sizes.

The author's personal experience and an evaluation by Kessler et al. [67] suggest the

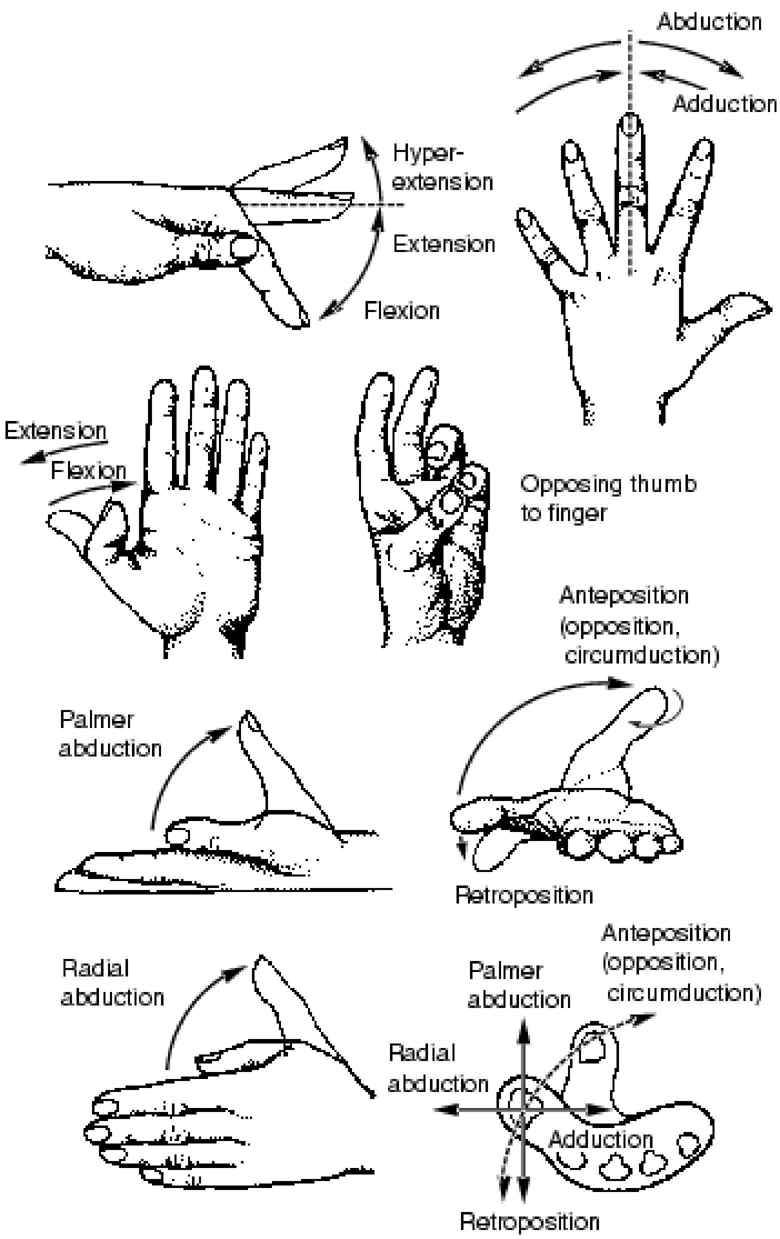

Figure B.5: The various motions that the hand and fingers can make using its 23 degrees of freedom (from Sturman [133]).

CyberGlove is accurate to within one degree of flexion. It works well for both simple and complex posture and gesture recognition (Wexelblat [149] and Fels [44] verify this claim). The only negative in regard to the CyberGlove is its price; the 18-sensor model is available for \$9800 and the 22-sensor model for \$14,500. But even though the glove is expensive, it is

Figure B.6: Virtual Technologies' CyberGlove, worn by the author, which can be equipped with 18 or 22 bend sensors.

the best available glove-based technology for accurate and robust hand posture and gesture recognition.

B.3 Vision-Based Technology

One of the main difficulties in using glove-based input devices to collect raw posture and gesture recognition data is the fact the gloves must be worn by the user and attached to the computer. In many cases, users do not want to wear tracking devices and computer-bound gloves since they can restrict freedom of movement and take considerably longer to set up than traditional interaction methods. As a result, there has been quite a bit of research into using computer vision to track human movement and extract raw data for posture and gesture recognition.

A vision-based solution to collecting data for hand posture and gesture recognition requires four equally important considerations. The first is the type, placement, and number of the vision device or devices used. Although almost all of the research in vision-based posture and gesture recognition has been conducted using video cameras, other devices such as the Motion Processor [138] and structured light [119] could potentially be used[3]. The Motion Processor illuminates the tracked object with infrared light sources and simultaneously detects these lights with an area image sensor while structured light uses a video camera and projector to track objects. Placing the camera is critical because the visibility

[3]Both of these methods can extract depth information without the using multiple cameras.

of the hand or hands being tracked must be maximized for robust recognition. Visibility is important because of the many occlusion problems present in vision-based tracking (see section B.4). The number of cameras used for tracking is another important issue. In general, one camera is used to collect recognition data, and it has been shown by Starner [128] and Martin [91] that one is effective and accurate in recognizing hand posture and gestures. When depth or stereo information is required for tracking hand movement, usually two or more cameras are needed. Although using more than one camera adds complications due to the algorithmic complexity of dealing with more than one image stream, they do provide more visibility and are critical in virtual environment applications which usually require depth information. Kuno [74] and Utsumi [139] use multiple vision devices effectively in the context of 3D object manipulation and 3D scene creation, respectively. Also, Rehag and Kanade [120] have shown that 27 DOF of the hand can be recovered by using two cameras.

The second consideration in a vision-based solution for hand posture and gesture recognition is to make the hands more visible to the camera for simpler extraction of hand data. One of the first ways of doing this was to place LEDs (light emitting diodes) on various points on the hand [131]. These LEDs let the camera quickly pick up feature points on the hand to aid recognition. A more common method in the literature is to simply use colored gloves. Starner [129], Davis [33], and Kuno [74] have all shown that using solid colored gloves allows faster hand silhouette extraction than simply wearing no gloves at all, but using such gloves makes it difficult to recognize finger movement and bending. In order to achieve fast silhouette extraction and track finger joint movement, Dorner developed a complex encoding scheme using sets of colored rings around the finger joints instead of solid colored gloves [36].

Even though colored gloves are wireless and simple to wear, the ideal situation for vision-based hand tracking is to track the hand with no gloves at all. Tracking a gloveless hand presents some interesting difficulties, among them skin color and background environment issues. In many cases, a solid colored screen is placed behind the user so that the natural color of the hands can be found and features extracted. One of the best vision systems for tracking the naked hand was Krueger's VIDEODESK system [73], although it required complex image-processing hardware. He was able to track hand silhouettes in order to create simple 2D and 3D shapes. Utsumi also tracked the naked hand in his 3D scene creation system [139].

The third consideration when using a vision-based solution for hand gesture and posture recognition is the extraction of features from the stream or streams of raw image data; the fourth consideration is how to apply recognition algorithms to these extracted features.

Both these considerations are discussed in Appendix C .

B.4 Advantages and Disadvantages of Glove- and Vision-Based Data Collection Systems

We can now examine the advantages and disadvantages of glove-based and vision-based technology for hand posture and gesture recognition. Kadous [65] and Sturman [133] have also discussed these issues to varying extents.

Hardware Cost

Even though glove-based technology has come down in price (under $500 for the 5DT Glove), the cost of robust and complex posture and gesture recognition is going to be high if a glove-based solution is used. The cost of a tracking device and a robust glove is in the thousands of dollars. On the other hand, a vision-based solution is relatively inexpensive, especially since modern-day workstations are often equipped with cameras.

User Comfort

With a glove-based solution, the user must wear a tracking device and glove that are connected to a computer. Putting these devices on takes time, can be quite cumbersome, and can limit one's range of motion. With a vision-based solution, the user may have to wear a glove, but the glove will be extremely lightweight, easy to put on, and not connected to the computer. Applications in which no gloves are used, give the user complete freedom of motion and provides a cleaner way to interact and perform posture and gesture recognition.

Computing Power

Depending on the algorithms used, both glove-based and vision-based solutions can require significant computing power. However, in general, the vision-based approach takes more computing power due to the image processing necessary. Glove-based solutions have a slight advantage over vision-based solutions in that the data the gloves send to the computer can easily be transformed into records that are suitable for recognition. However, with faster computers, computational power should not be an issue.

Hand Size

Human hands vary in shape and size. This is a significant problem with glove-based solutions: some users cannot wear these input devices because their hands are too big or too small. This problem is not an issue with vision-based solutions.

Hand Anatomy

Glove-based input devices may not always fit well enough to prevent their position sensors from moving relative to the joints the sensors are trying to measure. This problem reduces recognition accuracy after extended periods of use and forces users to recalibrate the devices which can be a nuisance. This problem also is not an issue with vision-based solutions.

Calibration

Calibration is important in both vision- and glove-based solutions but, due to the anatomy of the hand, it is more critical with glove-based solutions. In general, a calibration procedure or step is required for every user and, in some cases, every time a user wants to run the system. In some vision-based solutions, however, a general calibration step can be used for a wide variety of users.

Portability

In many applications, especially gesture to speech systems, freedom from being tied down to a workstation is important. With glove-based solutions, this freedom is generally available as long as hand tracking is not involved, since these input devices can be plugged right into a laptop computer. Vision-based solutions were originally quite difficult to use in a mobile environment due to camera placement issues and computing power requirements. However, with the advent of wearable computing [86] and powerful laptops with built-in cameras, mobile vision-based solutions are becoming more practical.

Noise

In glove-based solutions where hand tracking is required, some type of noise is bound to be associated with the data (it can come from a variety of sources depending on the tracking technology used). Filtering algorithms are therefore necessary to reduce noise and jitter. In some cases this can get computationally expensive when predictive techniques such as

Kalman filtering [147] are used. With a vision-based solution, noise can also be problematic but it does not seem to be as severe as with glove-based solutions.

Ease of Implementation

In general, a vision-based solution requires more complex algorithms for feature extraction and, as a result, implementation is more difficult. With glove-based solutions, feature extraction is somewhat simpler because the raw data is easier to work with.

Resolution and Sensitivity

With the exception of one study [67] which only examined the CyberGlove, the only measures available for comparing the sensitivity of glove- and vision-based solutions are bits of precision for glove-based devices and pixel resolution for vision-based solutions. Unfortunately, there is no direct way to compare these two measures. A possible metric would be to examine the smallest change in the real world that can be detected by a device; for instance, how small a finger movement can be detected. Until such a measure gains acceptance, comparing the resolution of vision- and glove-based systems will remain difficult.

Occlusion

Occlusion represents a major problem in vision-based solutions since, in many cases, the camera will not be able to pick up information about parts of the hand that are occluded by other parts. For example, it is difficult to extract information from all of the fingers when the hand is oriented in certain ways. Multiple cameras can often alleviate some of the occlusion but this increases algorithmic complexity. Occlusion is the biggest disadvantage for using vision-based solutions; for glove-based solutions it's a nonissue.

Accuracy

In both vision- and glove-based solutions for hand posture and gesture recognition, accuracy is one of the most critical components to providing robust recognition. Both these solutions provide the potential for high levels of accuracy depending on the technology and recognition algorithms used. Accuracy also depends on the complexity and quantity of the postures and gestures to be recognized. Obviously, the quantity of possible postures and gestures and their complexity greatly affect accuracy no matter what raw data collection system is used.

Appendix C

Hand Posture and Gesture Recognition Techniques

Once the raw data has been collected from a vision- or glove-based data collection system, it must be analyzed to determine if any postures or gestures can be recognized. In this appendix, various algorithmic techniques for recognizing hand postures and gestures are discussed.

The process of recognition can be generally divided into two stages: extraction of features from the raw data, and classifying the input based on those features. We then divide the classification algorithms into three categories: template matching, statistics, and miscellaneous techniques. The techniques will be discussed through a general introduction to the technique, a look at the current literature, and an analysis of advantages and disadvantages (see Tables C.2 and C.3 and Figure C.1 for a summary of both the feature extraction and classification algorithms discussed in Appendix C).

C.1 Feature Extraction Techniques

In the first phase of hand posture and gesture recognition, low-level information from the raw data is analyzed to produce higher-level semantic information which is then sent to a classification algorithm. In general, feature extraction can be as simple as extracting bend sensor values from a data glove or something more complicated such as creating useful feature vectors from images. This section contains five of the most common techniques used in hand posture and gesture recognition.

ABBREVIATION	TECHNIQUE
SFE	Simple Feature Extraction
ASM	Active Shape Models
PCA	Principal Component Analysis
LFM	Linear Fingertip Models
STVA	Spatial Temporal Vector Analysis
CTM	Classical Template Matching
IBL	Instance-based Learning
LA	Linguistic Approach
ABM	Appearance-based Motion
HMM	Hidden Markov Models
NN	Neural Networks
CA	Causal Analysis

Table C.1: The abbreviations for the feature extraction and classification algorithms discussed in Appendix C. They are referred to in Tables C.2 and C.3 and Figure C.1.

	Vision	Glove	Training	Previous Work
Feature Extraction				
SFE	No	Yes	Minimal	Moderate
ASM	Yes	No	None	Minimal
PCA	Yes	Yes	Moderate	Moderate
LFM	Yes	No	Minimal	Minimal
STVA	Yes	No	Minimal	Minimal
Classification—Template Matching				
CTM	Yes	Yes	Minimal	Extensive
IBL	Yes	Yes	Extensive	Moderate
LA	Yes	Yes	Minimal	Minimal
ABM	Yes	No	Minimal	Minimal
Classification—Statistics				
HMM	Yes	Yes	Extensive	Extensive
Classification—Miscellaneous				
NN	Yes	Yes	Extensive	Extensive
CA	Yes	No	Minimal	Minimal

Table C.2: A summary of the feature extraction and classification algorithms found in Appendix C. The table shows information about whether a technique has been used in a glove- or vision-based solution, the extent of the training required, and how much work has been done using the technique. The key to the abbreviations is found in Table C.1.

	SFE	ASM	PCA	LFM	STVA
CTM	SCP-98%(G), SSG-96%(G)	–	MCP-99%(V)	SCP-90%(V)	–
IBL	LCP-80%(G)	–	–	–	SCG-94%(V)
LA	SSP-50%(G)	–	–	–	–
ABM	–	–	Applied to Body Motion	–	–
HMM	MCP-90%(V), SCG-96%(G)	–	–	–	–
NN	MCP-98%(G), SCG-96%(G)	–	–	–	–
CA	–	–	–	–	–

Table C.3: A correlation between the different feature extraction techniques and the classification algorithms. Each applied entry has either one or two codes associated with it. Each consists of 3 letters, a number, and then another letter. The first letter states what the posture or gesture set size is, the second letter says whether the set was simple or complex, and the third letter says whether we are dealing with postures or gestures. The number shows the highest reported accuracy number for that particular configuration and the last letter in parentheses states whether the configuration was done using a glove- or vision-based solution. The key to the abbreviations is found in Table C.1.

C.1.1 Simple Feature Extraction and Analysis

Overview. Simple feature extraction derives simple mathematical quantities from the raw data. In its most simplistic form, the data from the bend sensors in a glove-based device and the position and orientation data from a six DOF tracker are the features extracted from the physical world. In other cases, a simple mathematical transformation is applied to the raw data. These types of transformations include distance metrics, velocity and acceleration information, energy measurements, and angle information.

Details and Examples. One of the first gestural interfaces to use a feature-based system was Rubine's 2D single-stroke gesture recognizer [121]. Rubine extracted such features as the cosine and sine of the initial angle of the gesture, the distance between the first and last point, the maximum speed of the gesture, and the length and angle of the bounding box diagonal. From these features, the system used a linear classifier[1] combined with template matching (for rejecting ambiguous gestures and outliers) to recognize gestures that represented numbers and letters of the alphabet, among others. The system recognized these gestures with over 97% accuracy.

This type of feature-based approach has also been applied to recognizing hand postures and gestures. However, it is slightly more complex due to the increase in dimensions from

[1] A linear classifier is equivalent to the function implemented by a simple neural network with linear activation functions (see Section C.2.3.1).

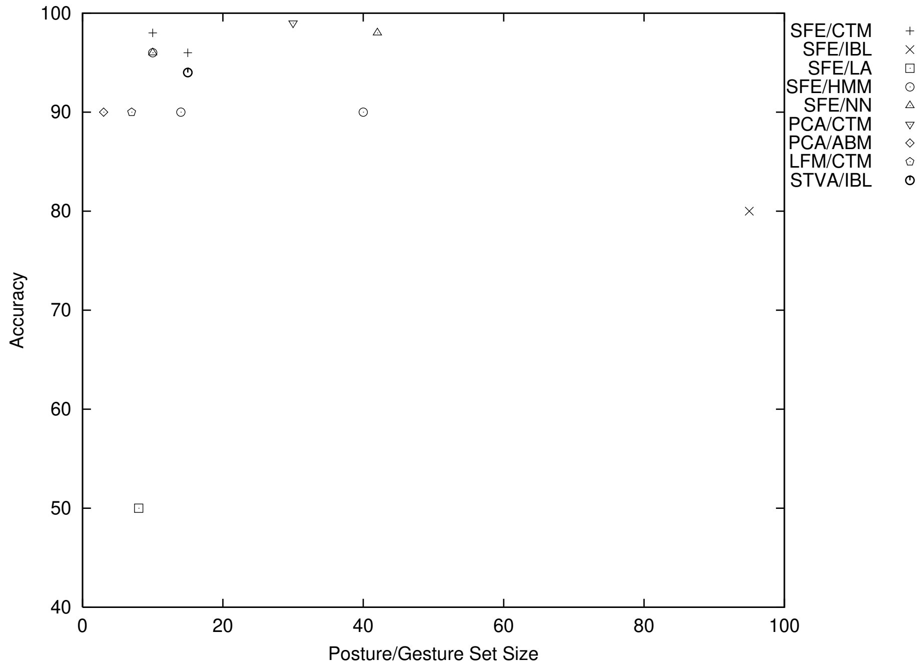

Figure C.1: A graph showing posture and gesture set sizes and accuracies for different feature extraction-classification combinations. The key to the abbreviations is found in Table C.1.

two to three and the increase in the amount of raw data produced by the input devices. Sturman [133] was the first person to extend the ideas behind Rubine's work into three dimensions and to make possible continual analysis and recognition (instead of requiring a starting and ending point in the gesture). Sturman used explicit formulations for each gesture that do not require training by individual users; however, manual programmer intervention was necessary to add new gestures. Both position data for a tracker and flex-sensor data were kept for feature extraction. The features used were similar to Rubine's but also included such inherently 3D features as cross product and bounding volume.

Using the work by Rubine and Sturman, Wexelblat developed a robust hand gesture and posture analyzer useful in a variety of applications [149]. Wexelblat's system uses a hierarchical structure that moves raw data from two CyberGloves and a set of trackers through a series of layers, each layer analyzing the data to provide higher level semantic meaning [150]. The first layer above the input devices uses entities called segmenters to watch the raw data. The segmenters look for significant changes[2] over time in the raw data

[2] In this case, significant changes are changes in bend angle, position and orientation data. Each segmenter has an associated minimum, maximum, and epsilon value which is used to determine what "significant" is.

values. Once a significant change is found, the segmenter sends a data record consisting of information that represents the data change to the next layer in the analyzer.

The next layer in Wexelblat's analyzer has a set of proto-feature detectors that extract (from the data records of one or more segmenters) information such as the extension of a finger or curvature in the palm. This information is sent to the higher-level feature detectors, which use the information from one or more proto-feature detectors to derive more global features like a fist, flat hand posture or a waving gesture. The path analysis module then takes data from all the feature detectors and puts the information into a frame to hold the completed description of the gesture. Finally, the frames are sent to an integration module that handles the interaction and performs temporal integration over them.

Analysis. Simple feature extraction and analysis is a robust way to recognize hand postures and gestures. It can be used not only to recognize simple hand postures and gestures but complex ones as well. Its main drawback is that it can become computationally expensive when a large number of features are being collected, which slows down system response time. This slowdown is extremely significant in virtual environment applications, but should diminish with the advent of faster machines. Finally, note that Wexelblat's feature extraction and analysis method could be used in vision-based solutions by altering how the features are extracted.

- *Strengths*
  - Handles postures and gestures equally well
  - Uses layered architecture to analyze postures and gestures
- *Weaknesses*
  - Can be computationally expensive depending on how many features are extracted
  - Similar postures or gestures can have similar features which makes it difficult to distinguish between them

C.1.2 Active Shape Models

Overview. Active shape models, or "smart snakes" as they are sometimes called, are a technique for locating a feature within a still image [30]. The features extracted, in this case, are the edges of the hand (see Figure C.2).

Figure C.2: The user's hand is being tracked with an active shape model (Taken from Heap [57]).

Details and Examples. The technique places a contour on the image that is roughly the shape of the feature to be tracked. The contour is then manipulated by moving it iteratively toward nearby edges that deform the contour to fit the feature. Heap and Samaria extend this technique to extract features for recognizing hand postures and gestures using computer vision [58]. In their system, they apply an active shape model to each frame and use the position of the feature in that frame as an initial approximation for the next frame. They also use a point distribution model[3] [29] to find a suitable model for the tracked object which aids in approximating the position of it in the next frame.

Analysis. Heap and Samaria's system runs in real time (25 frames per second) and is applicable only to vision-based solutions. The main disadvantages with this technique are that it has not been used as input to any classification algorithms (see Section C.2) and that currently it can track only the open hand, which would severely limit the number of hand postures and gestures that can be recognized. Also, there is very little empirical evidence in the literature to support its validity. Open areas of research using active shape models include extending them to the 3D domain (i.e. using multiple cameras) so that the number of possible postures and gestures can be increased and determining how well they work with various classification algorithms.

- *Strengths*
  - Allows real time recognition

[3] The Point Distribution model (PDM) is a shape description technique that can be used to find similar shapes in other images.

    - Can handle both hand postures and gestures

- *Weaknesses*
    - Tracks only the open hand
    - Unclear whether 3D information can be extracted

C.1.3 Principal Component Analysis

Overview. Principal component analysis (PCA) is a statistical technique for reducing the dimensionality of a data set in which there are many interrelated variables, while retaining as much of the variation in the data set as possible [64]. The data set is reduced by transforming the old data to a new set of variables (principal components) that are ordered so that the first few variables contain most of the variation present in the original variables. The original data set is transformed by computing the eigenvectors and eigenvalues of the data set's covariance matrix. The eigenvector with the highest eigenvalue holds the highest variance, the eigenvector with the second highest eigenvalue holds the second highest variance, and so on. If the data points corresponding to hand postures and gesture are organized into clusters, the the use of PCA assumes these clusters are distributed along the highest variance and, as a result, later classifications based on the extracted features will have greater discrimination power than other axes (see Figure C.3). PCA allows for extraction of features such as the mean and standard deviation of the principal components or clusters of pixel values which represent posture or gesture classes.

Details and Examples. PCA was first applied in the computer vision community to face recognition by Sirovich and Kirby [126] and later extended by Turk and Pentland [136]. Birk et al. and Martin independently developed the first two systems using PCA to recognize hand postures and gestures in a vision-based system [11, 91]. Birk's system was able to recognize 25 postures from the International Hand Alphabet, while Martin's system was used to interact in a virtual workspace.

Birk's system first performs PCA on sets of training images to generate a Bayes classifier [37] that is then used to classify postures in real time. Each set of training images can be considered a multivariate data set: each image consists of N pixels and represents a point in N-dimensional space. In order for PCA to work successfully, there must be little variance in at least one direction and whatever variance exists should not be meaningful. Birk's recognition system works well but there is little indication that PCA compresses the data set significantly beyond a naive approach.

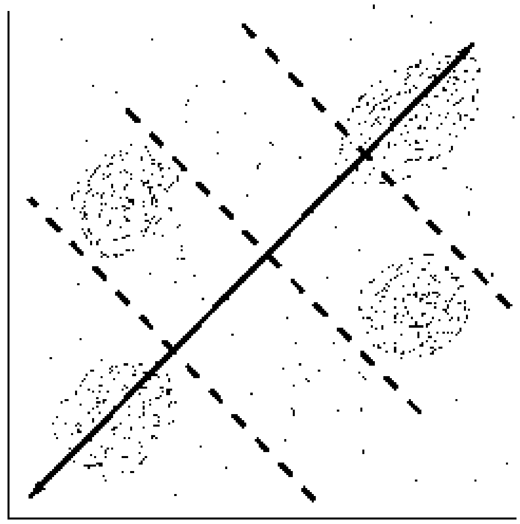

Figure C.3: The dots represent points in the data set while the solid line represents the axis of greatest variance (first principal component). The dashed lines represent potential classification divisions.

Another important issue when dealing with image data is that it is highly sensitive to position, orientation, and scaling of the hand in the image. PCA cannot transform two identical postures with different hand sizes and positions to the same point. Birk thus normalizes each image to center the hand, rotate it so that its major axis is vertical, and scale it to fit the gesture image.

The Bayes classifier, which is created off line using a technique similar to Turk and Pentland's [136] eigenfaces, is a transformation matrix containing the results of the PCA performed on all the images in the training set. After the transformation matrix has been calculated, the number of principal components is reduced by discarding the least important ones. The common approach is simply to keep the principal components with the n highest eigenvalues. However, Birk has shown that this is not effective when only a small number of principal components are to be kept [12]. Other information such as the number of posture classes, their mean, and their variance in the reduced data set can be used to choose the principal components. The Bayes classifier is then used to recognize postures from the reduced set of principal components. Since the data is assumed to have a normal distribution, the discriminant function used in the classifier is reduced to a multivariate normal distribution which can be further reduced to a Mahalanobis distance metric see Section C.2.1.1). Therefore, the classifier used in this technique is simply a form of classical template matching[4]. Note that such parameters as image resolution and number of training images are also important components of PCA in a vision-based solution and can be modified to improve recognition results [11].

Analysis. Although principal component analysis can be used in a glove-based solution [135], it has been used primarily in the computer vision community. It is accurate for specific posture sets such as the International Hand Alphabet [11] but requires training by more than one person to provide robust results. More research still needs to be done to measure the validity of this technique and to determine whether more complex hand gestures can be recognized accurately.

- *Strengths*
  - Can recognize on the order of 25 to 35 postures [11, 135]
  - Shows high levels of accuracy (greater than 95%) when used with template matching
- *Weaknesses*
  - Requires training by more than one person for accurate results and user independence [11]
  - Requires normalization to keep images consistent

C.1.4 Linear Fingertip Models

Overview. The linear fingertip model assumes that most finger movements are linear and comprise very little rotational movement. This assumption allows for a simplified hand model that uses only the fingertips as input data and permits a model that represents each fingertip trajectory through space as a simple vector. In general, the features extracted from linear fingertip models are both magnitude and direction of each fingertip.

Details and Examples. Davis and Shah use this approach in a vision-based solution that puts a glove with brightly colored fingertips on the user's hand [33]. The technique first extracts fingertip positions using histogram segmentation [53] based on the knowledge that the pixel intensity of the fingertips is known to be significantly different than the

[4]For a mathematical description of how a Bayes classifier can be reduced to a set of Mahalanobis distance measurements see Birk and Moeslund [12].

remaining regions. With this *a priori* knowledge, a histogram is created with the fingertip regions corresponding to the rightmost peak. A threshold is established (after smoothing the histogram) by finding the valley between the last two peaks. Any pixel greater than this threshold belongs to a fingertip and pixels which are less than the threshold get discarded. This segmentation creates a binary image where the only features are the fingertip regions. Note that each fingertip region is represented as a centroid point for ease of calculation.

Once the fingertips regions are detected in each image, their trajectories are calculated using motion correspondence [118] which maps points in one image to points in the next image such that no two points are mapped onto the same point (see Rangarajan and Shah [118] for the mathematics behind motion correspondence). The postures themselves are modeled from a small training set by storing a motion code, the gesture name, and direction and magnitude vectors for each of the fingertips. The postures are then recognized using classical template matching (see Section C.2.1.1) by checking if all the direction and magnitude vectors match (within some threshold) a gesture record in the training set.

Analysis. System testing showed good recognition accuracy (greater than 90%), but the system did not run in real time and the posture and gesture set should be expanded to determine if the technique is robust.

- *Strengths*
  - Simple approach
  - Concerned only with starting and ending points of fingertips
  - Has good recognition accuracy
- *Weaknesses*
  - System did not run in real time[5]
  - Recognizes only a small set of postures
  - Does not take curvilinear fingertip motion into account

C.1.5 Spatio-Temporal Vector Analysis

Overview. Spatio-temporal vector analysis is used to track the boundaries of the moving hand in a vision-based system [115]. This technique makes it possible to extract features

[5] With today's computer performance, this system should run in real time.

such as rotation invariants and normalized moments from a hand gesture which then can be used for hand gesture interpretation and recognition.

Details and Examples. Spatio-temporal vectors are computed in three steps. First, the images are processed to find the location of moving edges; second, an initial flow field describing the velocities at moving edge points is computed; finally, the flow field is refined and smoothed using a variance constraint. The location of moving edges is calculated based on the assumption that the moving hand is the fastest moving object in the static scene. The video image is modeled using a three-dimensional signal specifying an xy point and time. Partial derivatives with a varying x and y are computed from this signal using Sobel operators (a commonly used technique in image processing). The Sobel operators yield gradient images in which the image value is a measure of pixel intensity. These images are then combined with their corresponding time-derivative images to yield new images in which the pixel intensity is a measure of moving-edge intensity. The edges are extracted from these derived images by eliminating edge points whose neighbors parallel to the edge direction do not have a maximum magnitude.

After the moving edges have been found, the vector flow field of moving edges is computed. Edge velocities are calculated by first choosing dominant edge points at which to calculate velocities. Then edge point correspondences from different edge images are found by a correlation process called absolute difference correlation [1]. After the correlation process is complete, the best set of vectors, those which minimize a variance function [115], from the endpoint candidates is found. This set is calculated with a variance constraint that represents the normalized velocity change across the image's vector field[6]. The best set of vectors is then used as input to a inductive learning algorithm (see Section C.2.1.2 on instance-based learning) to recognize the predefined hand gestures.

Analysis. Spatio-temporal vector analysis has shown to be an accurate technique for vision-based solutions when combined with inductive learning. Quek and Zhao's [114] system achieves up to 94% accuracy on a small sized gesture set. However, the technique is mathematically complex and requires a significant amount of computation in order to track the hands and extract features.

- *Strengths*
  - Provides unobtrusive recognition

[6]For details on the mathematics behind this technique see Quek [115].

- Can recognize a small-sized set of hand postures and gestures accurately

- *Weaknesses*
  - Requires significant computation to track the hands
  - Complex mathematical implementation

C.2 Hand Posture and Gesture Classifiers

Once the features are collected, the second phase of hand posture and gesture recognition is to find the best classification for these features. This category contains a number of classification techniques, which we group into methods using template matching, statistical methods, and miscellaneous methods.

C.2.1 Template Matching

One of the most common approaches to recognizing hand postures and gestures is template matching in its various forms. In general, a template-based approach creates data records for each posture and gesture in the set and uses them to classify new postures and gestures.

C.2.1.1 Classical Template Matching

Overview. Classical template matching is one of the simplest methods for recognizing hand postures and has been discussed frequently, with thorough contributions by Sturman [133] and Watson [145]. Templates can be used in both glove-based and vision-based solutions; the templates are sequences of sensor values (gloves) and a static or small set of images (computer vision). Here we discuss only glove-based template matching although it is used in vision-based solutions as well. In general, template matching determines whether a given data record can be classified as a member of a set of stored data records.

Details and Examples. Recognizing hand postures using template matching has two parts. The first is to create the templates by collecting data values for each posture in the posture set. Generally, each posture is performed a number of times and the average of the raw data for each sensor is taken and stored as the template. The second part is to compare the current sensor readings with the given set of templates to find the posture template most closely matching the current data record.

An example of the template matching comparison is the use of a Boolean function on the results of the explicit comparison between each sensor value in the current data record and the corresponding value in the posture templates. The comparisons are often made within a range of values, which helps to improve recognition accuracy with noisy glove devices. A problem with comparisons of angle measurements within a range of values, however, is that while in theory, these ranges usually go from zero to a power of two based on the bits of precision in each bend sensor, the actual angle measurements from the bend sensors do not follow their theoretical ranges and each bend sensor often has a different range. A way to remedy this problem is to normalize the bend sensor's measurements using the maximum and minimum value for each bend sensor. Normalization makes all the angle measurements zero for full extension and one for full flexion, thus making comparisons with ranges easier to implement. The drawback of normalizing bend angles is that maximum and minimum values can change during glove use; however dynamic calculation and updating of maximum and minimum values has been shown to combat this problem [133].

Another example of template matching comparisons is the use of distance measurements between the current data record and each of the data records in the posture set recognizing the posture with the lowest distance measurement. The distance measurement must be below some threshold value to avoid false positive recognition. Three distance measurements used for hand posture template matching are the sum of the absolute differences [155], the Euclidean distance (i.e. sum of the squares) [102], and the Mahalanobis distance[7] [12, 33]. The main advantage of computing a distance measurement is that comparison ranges need not be used. The main disadvantage is that a measurement must be made for each posture template.

Analysis. Template matching is the simplest of the hand posture and gesture recognition techniques, and for a small set of postures, it is appropriate and can be quite accurate. But the technique does have limitations. First, template matching is much more difficult for hand gestures. However, Darrell and Pentland have recognized hand gestures in a vision-based solution using sets of view models or templates that are matched with gesture patterns using dynamic time warping [35][8]. The second limitation is the small number of possible

[7]The Mahalanobis distance constructs an ellipsoid in multidimensional space where variations in the directions of the shorter axes have more weight. It takes into consideration the variance and correlation of the variables (through a covariance matrix) in measuring the distance between points. Note that if the covariance matrix is independent of the variables the Mahalanobis distance reduces to a Euclidean distance metric.

[8]In this case, two gestures were recognized [35].

postures that can be recognized. If the application requires a medium or large posture set, then template matching will not work since the posture templates will overlap [145].

- *Strengths*
  - Simplest technique to implement
  - Accurate (for small set of postures)
  - Requires only a small amount of calibration
- *Weaknesses*
  - Not suited for hand gestures
  - Does not work well for medium and large posture sets due to overlapping templates [145]

C.2.1.2 Instance-Based Learning

Overview. Instance-based learning can be considered an elaborate form of template matching that has a training component which continuously create new templates thus allowing the user's input to implicitly influence the recognizer. It stems from work done in machine learning with the main difference between instance-based learning and other learning algorithms such as neural networks (see Section C.2.3.1) and hidden Markov models (see Section C.2.2.1) is the way in which the training data is used. With supervised neural networks, for example, the training data is passed through the network and the weights at various nodes are updated to fit the training set. With instance-based learning, the training data is simply used as a database in which to classify other "instances". An instance, in general, is a vector of features of the entity to be classified. For example, in posture and gesture recognition, a feature vector might be the position and orientation of the hand and the bend values for each of the fingers.

Details and Examples. Instance-based learning methods include techniques that represent instances as points in Euclidean space, such as the K-Nearest Neighbor algorithm, and techniques in which instances have a more symbolic representation, such as case-based reasoning [96]. In the K-Nearest Neighbor algorithm, an instance is a feature vector of size n with points in n-dimensional space. The training phase of the algorithm involves storing a set of representative instances in a list of training examples. For each new record, the Euclidean distance is computed from each instance in the training example list, and the K

closest instances to the new input are returned. The new input is then classified and added as a new instance to the training example list so that the algorithm can continually train on the run-time instances. In the case of hand posture recognition, the training set would be divided into a number of categories based on the types of recognizable postures. As a new posture instance is entered, its K nearest neighbors are found and the most common posture of the ones corresponding to those neighbors is taken as the category in which the instance should be placed (thus recognizing the instance as a particular posture).

Another type of instance-based learning is case-based reasoning, in which instances have more elaborate descriptions. A typical instance in a case-based reasoning system might be a description of a mechanical part. Since the instance descriptions are more elaborate, simple approaches to determining instance similarity, such as the Euclidean distance measure, are not applicable. Other methods for determining similarity between instances must be used, such as those found in knowledge-based techniques such as first order logic[9].

Rule-based induction [113] is a form of learning which falls within but has not historically been associated with case-based reasoning. With feature vectors as input, rule-based induction creates a rule base using disjunctive normal form (DNF) formulae which can represent a particular hand posture or gesture with one formula. Each conjunction within a DNF formula is considered a single rule. Given a rule base of DNF formulae from a training set, a new posture or gesture instance is placed into one of three categories:

1. The new observation may satisfy one and only one formula in which case it is classified by that formula
2. The new observation satisfies more than one formula which means it may belong to any of the satisfiable classes
3. The new observation satisfies no DNF formula

Quek's [113] algorithm employs two matching strategies: exact match and flexible match. With exact match, only if the first category holds will there be a match. With flexible match, if category two holds, the conjunction is examined for each matching DNF. A count is kept of the number of training instances of that class which matches the rule. The class with the highest similar training examples is taken as the matching class (see Quek [113] for details on this algorithm). Note that Quek tested both exact and flexible matching on a set of 15 hand gestures. Exact matching resulted in approximately 90% accuracy while

[9] See Mitchell [96] for more detail on case-based reasoning, the K-Nearest Neighbor algorithm, and other instance-based learning algorithms.

flexible matching yielded a 94% accuracy result which indicates that the flexible approach seems to be more robust.

Instance-based learning techniques have the advantage of simplicity, but they have a number of disadvantages as well. One major disadvantage is the cost of classifying new instances. All of the computation must be redone whenever a new instance is classified, which means there will be response time issues when dealing with a large amount of training examples. Another disadvantage of these methods is that not all of the training examples may fit in main memory, and thus will also increase response time. Note that Aha has developed two instance-based learning algorithms that alleviate some of these problems [2]. The first one saves space by discarding instances that have already been correctly classified, and the second makes assumptions about the data to weed out irrelevant instances.

Aside from Quek's work, there has been little work done on instance-based learning in recognizing hand postures and gestures. One of the few systems reported in the literature was developed by Kadous [65], which recognized 95 discrete hand postures for performing sign language recognition using the three instance-based learning algorithms described by Aha [2]. An interesting feature of this system was its capability of achieving approximately 80% accuracy with the use of a Power Glove as the raw data collection unit.

Analysis. Instance-based learning shows promise as a way to recognize hand postures. However, response time may be an important factor when issuing posture commands due to the amount of computation required when each instance is generated, especially when instances are generated at 30 or more per second (based on the speed of the input devices used).

- *Strengths*
  - Except for case-based reasoning, instance-based learning techniques are relatively simple to implement
  - Can recognize a large set of hand postures with moderately high accuracy
  - Can recognize a small set of hand gestures with high accuracy
  - Provides continuous training
- *Weaknesses*
  - Requires a large amount of primary memory as training set increases under the naive approach

  - Response time issues may arise due to a large amount of computation at instance classification time
  - Only a little reported in the literature on using instance-based learning with hand postures and gestures

C.2.1.3 The Linguistic Approach

Overview. The linguistic approach uses a formal grammar to represent the hand posture and gesture set.

Details and Examples. Hand [56] used this approach to recognize a small set of postures under the constraint that postures have fingers either fully flexed or fully extended in a number of configurations. Hand postures were mapped to the grammar, which was specified by a series of tokens and production rules. The system used a Power Glove and an ultrasonic tracker to record raw data. Recognition accuracy was poor, ranging from about 15 to 75 percent depending on the posture to be recognized. Hand [56] claims the poor recognition rate may be due to the inaccuracy of the Power Glove; however, Kadous's work [65] with a Power Glove for instance-based learning (see Section C.2.1.2) achieved a much higher recognition rate using a larger posture set.

Analysis. The linguistic approach does not appear to be accurate or robust enough to handle most hand posture and gesture recognition tasks.

- *Strengths*
  - Simple approach
  - Can be used in either a vision- or glove-based solution
- *Weaknesses*
  - Poor recognition results
  - Limited to only simple hand postures
  - Little work reported in the literature using this technique to recognize hand postures and gestures

C.2.1.4 Appearance-Based Motion Analysis

Overview. Appearance-based motion analysis exploits the observation that humans can recognize actions from extremely low resolution images and with little or no information about the three-dimensional structure of the scene.

Details and Examples. Using this observation, Davis [33] developed an appearance-based recognition strategy for human motion using a hypothesize and test paradigm [55]. This strategy recognizes motion patterns from a video stream by first creating a filter called a binary motion region (BMR), which describes the spatial distribution of motion energy for a given action. In other words, the filter highlights regions in an image in which any form of motion has been present since the beginning of the action. The BMR acts as an index into a database of binary motion regions collected from training sessions. If the current BMR is close to any of the stored BMRs, then the current BMR is tested against a motion model of an action.

Once a set of possible actions has been found by the current BMR, the unknown movement is classified as one of the predefined actions. Davis developed two approaches to represent actions for classification. The first approach creates specific region-based motion parameterizations by using principal component analysis (see Section C.1.3) to reduce motion time traces of particular regions of the image to a single coefficient vector. Template matching is then performed using this coefficient vector and the training data. The second approach transforms action sequences into a single motion history image where the pixel intensity indicates the recency of motion at a given location. Features from the single image are extracted and matched (using the Mahalanobis distance metric [41]) against known movement[10].

The system was tested for a small set of motions (sitting, arm waving, and crouching) with good results (90%) for those subjects included in the training phase. Subjects who were not included in the training phase performed significantly lower (76%) which Davis claims is representative of not enough training data.

Analysis. Although it did not specifically recognize hand gestures, appearance-based motion analysis could be used to recognize very simple ones, but the technique will not work with gestures and postures that have complicated finger movement. More research is needed to determine if a more complex set of motions can be recognized with the existing technique,

[10]Note that a detailed description of the recognition technique described in this and the previous paragraph can be found in Davis [33].

since most hand gestures are more complicated than the set of test motions used in Davis's evaluation.

- *Strengths*
  - Provides unobtrusive recognition
  - Accurate recognition with a small set of motions for the trained user
- *Weaknesses*
  - Has not been used for recognizing hand posture and gestures
  - Very difficult to detect small details in finger movement

C.2.2 Statistical Methods

A common technique that has been popular with the pattern and speech recognition communities is the use of probabilistic or statistical models. One technique that is prevalent in the hand posture and gesture recognition literature is hidden Markov models.

C.2.2.1 Hidden Markov Models

Overview. In describing hidden Markov models it is convenient first to consider Markov chains. Markov chains are simply finite-state automata in which each state transition arc has an associated probability value; the probability values of the arcs leaving a single state sum to one. Markov chains impose the restriction on the finite-state automaton that a state can have only one transition arc with a given output; a restriction that makes Markov chains deterministic. A hidden Markov model (HMM) can be considered a generalization of a Markov chain without this Markov-chain restriction [25]. Since HMMs can have more than one arc with the same output symbol, they are nondeterministic, and it is impossible to directly determine the state sequence for a set of inputs simply by looking at the output (hence the "hidden" in "hidden Markov model"). For a more detailed description see Rabiner and Juang [116], Rabiner [117], Huang et al. [61], or Charniak [25].

Details and Examples. A HMM is defined as a set of states of which one state is the initial state, a set of output symbols, and a set of state transitions. Each state transition is represented by the state from which the transition starts, the state to which the transition moves, the output symbol generated, and the probability that the transition is taken [25]. In the context of hand gesture recognition, each state could represent a set of possible hand positions. The state transitions represent the probability that a certain hand position transitions into another; the corresponding output symbol represents a specific posture and a sequence of output symbols represent a hand gesture. One then uses a group of HMMs, one for each gesture, and runs a sequence of input data through each HMM. The input data, derived from pixels in a vision-based solution or from bend sensor values in a glove-based solution, can be represented in many different ways, the most common by feature vectors [129]. The HMM with the highest forward probability (described later in this section) determines the users' most likely gesture. An HMM can also be used for hand posture recognition; see Liang and Ouhyoung [82] for details.

A number of important issues arise in dealing with HMMs. As with neural networks, training HMMs is very important for increasing recognition accuracy of the model. A common approach is to adjust the HMM's transition probabilities in order to optimize it for a training data set. If the training data is an accurate representation of a particular gesture, for example, then the HMM should be able to recognize that gesture given new data. The Baum-Welch algorithm [25, 116, 117] uses the given training sequence to reestimate the probabilities of the state transitions in the HMM.

One of the components of the Baum-Welch algorithm is forward probability. Forward probability, or alpha probability as it is sometimes called, is the probability of an HMM given an output sequence and is calculated by incrementally computing the probabilities for the output on a symbol-by-symbol basis [116]. The algorithm goes through each timestep t and examines each state j in the HMM. For each state j, it computes a summation of the probability of producing the current output symbol at t and moving to j given that the algorithm was in state k, multiplied by the probability of producing the output up to t and ending up in k. Note that the summation is over all k. Performing this calculation iteratively for each output symbol yields the probability that the HMM would generate the whole output sequence. The forward probability is used to find a HMM with the highest probability of matching an output sequence. For example, if a hand gesture set contains 10 gestures, each one having its own HMM, the HMM with the highest forward probability score would determine which gesture to recognize. See Charniak [25] for a more detailed description of forward probability.

As stated previously, one of the drawbacks of HMMs is that one cannot directly determine the state sequence for a set of output symbols since a state can have more than one arc with the same output. Nevertheless, this information can be approximated by finding the most likely states in the HMM. A common technique for finding the best state sequence for a given output, the Viterbi algorithm [116], uses calculations similar to forward probability except the maximum is taken instead of taking a summation over all k. The Viterbi algorithm is useful because it is a fast method for evaluating HMMs. For more detail on the Viterbi algorithm see Charniak [25].

Hidden Markov models were first used in the recognition community for recognizing handwriting [141] and speech [61], and more recently a significant amount of research has been done on applying HMMs to hand posture and gesture recognition. Starner used HMMs in a vision-based solution to recognize a forty word subset of American Sign Language [129]. Instead of using a different model for each sign in the recognition set, Starner found the minimum and maximum number of states required for an individual HMM and then, using skip transitions (which give low weights to state transitions that are not needed) developed a general HMM topology for all models used in the system. With ample training of the HMMs (between 20 and 80 training samples for each sign) the system was able to achieve accuracies of over 90 percent.

Schlenzig also used a single HMM to recognize hand gestures in a vision-based solution [125]. The state of the HMM represents the gestures and the observation symbols represent the current static hand posture. This HMM had three possible states and nine possible observation symbols so the number of recognizable gestures was limited. The system employs a recursive filter for updating estimates of the gesture being recognized based on the current posture information. This recursive estimator allows recognition of combined gestures and requires only one HMM.

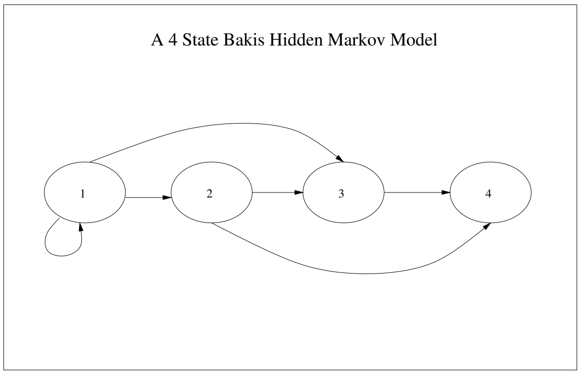

Figure C.4: A four state Bakis HMM.

Lee and Xu were able to recognize a set of 14 gestures in the context of robot teleoperation [81]. The system uses a CyberGlove and implements a Bakis hidden Markov model that restricts state transitions so that a given state can have a transition only to itself or one of the next two states; for example, if the HMM has four states, state two could have a transition arc only to itself, state three, or state four (see Figure C.4). This type of model assumes that the recognizable gestures have no cyclical properties. The system also performed iterative model updates as each gesture is recognized. Recognition accuracy of over 90 percent was achieved.

Nam and Wohn [100] and Liang and Ouhyoung [82] have also explored the use of HMMs in hand posture and gesture recognition. Nam and Wohn use a glove-based solution with a VPL DataGlove and Polhemus tracker and reduce the dimensional complexity of the tracking device from 6D (three position axes and three orientation axes) to 2D by using plane fitting. The 2D information is then sent to the HMM. After training each of the 10 HMMs, one for each gesture, with approximately 200 training samples per gesture, accuracy of over 96 percent was attained. Liang and Ouhyoung use HMMs to recognize 50 signs in Taiwanese Sign Language. The system uses a n-best approach to outputting recognition results. Unfortunately, no recognition accuracies were reported.

Analysis. Hidden Markov models provide a good way to perform hand posture and gesture recognition, and can be used in both vision-based and glove-based solutions. The literature has shown that high accuracy can be achieved and the number of possible hand gestures or postures in a posture or gesture set can be quite large. Like neural networks, HMMs must be trained and the correct number of states for each posture or gesture must be determined to maximize performance. If the number and types of hand posture and gestures are known beforehand, HMMs are a good choice for recognition. If the hand postures and gestures are determined as the system is developed, the development process can be more time-consuming due to retraining. If one HMM is used for all gestures, as in Starner's work, then the single HMM must be retrained. If each gesture has an associated HMM, then only the new gesture's HMM will have to be trained. Although HMMs require extensive training, and their hidden nature makes it difficult to understand what is occurring within them, they still may be the technique of choice since they are well covered in the literature and the accuracies reported are usually above 90 percent.

- *Strengths*
  - Can be used in either a vision- or glove-based solution
  - Can recognize medium posture or gesture sets
  - With adequate training, high accuracy can be achieved

  - Well discussed in the literature

- *Weaknesses*
  - Training can be time consuming and does not guarantee good results
  - As with multi-level neural networks, the hidden nature of HMMs makes it difficult to observe their internal behavior

C.2.3 Miscellaneous Classification Algorithms

Two classification techniques that are neither template-based nor statistically-based are neural networks and causal analysis.

C.2.3.1 Neural Networks

Overview. A neural network is an information processing system loosely based on the operation of neurons in the brain. While the neuron acts as the fundamental functional unit of the brain, the neural network uses the node as its fundamental unit; the nodes are connected by links, and the links have an associated weight that can act as a storage mechanism [123]. For a more comprehensive description, see Russell and Norvig [123], Krose and van der Smagt [72] or Anderson [3].

Details and Examples. With a neural network, each node is considered a single computational unit containing two components. The first component is the input function which computes the weighted sum of its input values; the second is the activation function, which transforms the weighted sum into a final output value. Many different activation functions can be used; the step, sign, and sigmoid functions being quite common [123] since they are all relatively simple to use. For example, using the step function, if the weighted sum is above a certain threshold, the function outputs a one indicating the node has "fired" otherwise it outputs a zero indicating the node has not fired. The other two activation functions act in a similar manner. Note that if a linear activation function is used, the neural network reduces to a simple linear classifier [37].

Neural networks generally have two basic structures or topologies, a feed-forward structure and a recurrent structure. A feed-forward network can be considered a directed acyclic graph, while a recurrent network has an arbitrary topology. The recurrent network has the advantage over a feed-forward network in that it can model systems with state transitions.

However, recurrent networks require more complex mathematical descriptions and can exhibit chaotic behavior. In both network topologies, there is no restriction on the number of layers in the network. These multilayered networks provide more representation power at the cost of more complex training. The nodes in between the input and output nodes of the multilayered network have no communication with the outside world and cannot be directly observed from the input or output behavior of the network. If the number of hidden nodes is large, it is possible to represent any continuous function of the inputs [123].

Training is an important issue in neural networks and can be classified in two different ways. First, supervised learning trains the network by providing matching input and output patterns; this trains the network in advance and as a result the network does not learn while it is running. The second learning mechanism is unsupervised learning or self-organization which trains the network to respond to clusters of patterns within the input. There is no training in advance and the system must develop its own representation of the input, since no matching output is provided [72]. Note that supervised and unsupervised learning do not have to be mutually exclusive: depending on the network, a combination of the two learning strategies can be employed. Neural network training is one of most important areas of research in neural network technology, but the many different algorithms for supervised and unsupervised learning strategies are beyond the scope of this survey; for a thorough discussion see Mehrotra, Mohan, and Ranka [94].

Neural networks have been used principally in the artificial intelligence community to build certain types of autonomous agents and recognize patterns. One of the first systems to use neural networks in hand posture and gesture recognition was developed by Murakami [99]. Hand postures were recognized with a three-layered neural network that contained 13 input nodes, 100 hidden nodes, and 42 output nodes, one for each posture to be recognized. The network used back propagation, a learning mechanism that minimizes the error between target output and the output produced by the network [123], and achieved 77% accuracy with an initial training set. Accuracy increased to 98% for participants in the original training set when the number of training patterns was increased from 42 to 206. Hand gesture recognition was done with a recurrent three-layered network with 16 input units, 150 hidden units, and 10 output units, one for each of the 10 possible gestures recognized. Recurrency in the network allowed for processing of time variant data. Recognition accuracy was initially 80%, but 96% recognition rates were achieved with a filtering mechanism for the raw data[11].

[11] See Murakami for details on this mechanism [99].

Fels did extensive work with neural networks with his Glove-TalkII system [43] in which three back propagation neural networks are used to translate hand gestures into speech. The hand gesture recognition process was broken up into three networks in order to increase speed and reduce training time. The first network, the vowel/consonant network, determined if the user wanted to produce a vowel or a consonant. This network employed a feed-forward topology with 12 input nodes, 10 hidden nodes and one output node. The second network was used to generate consonant sounds. It also employed a feed-forward topology but used 12 input nodes, 15 hidden nodes, and nine output nodes. The third network used to generate vowel sounds, had a feed-forward structure and employed two input nodes, 11 hidden nodes, and eight output nodes. The consonant and vowel networks used normalized radial basis activation functions [19] for the hidden inputs that solved problems arising from similar-sounding consonants and vowels (see Fels [44]). With these networks, a well-trained user (100 hours of training, including training the networks) was able to make intelligible speech that sounded somewhat natural.

Another system using neural networks developed by Banarse [7] was vision-based and recognized hand postures using a neocognitron network, a neural network based on the spatial recognition system of the visual cortex of the brain (for a detailed description of the neocognitron see Fukushima [51]). However, the system was limited and recognized only a handful of postures. More extensive research is needed to determine the validity of the neocognitron network as a hand posture and gesture recognition technique.

Analysis. Neural networks are a useful method for recognizing hand postures and gestures, yield increased accuracy conditioned upon network training, and work for both glove-based and vision-based solutions. However, they have distinct disadvantages. First, different configurations of a given network can give very different results, and it is difficult to determine which configuration is best without implementing them: Fels [44] reports that he implemented many different network configurations before finding ones that provided good results. Another disadvantage is the considerable time involved in training the network. Finally, the whole network must be retrained in order to incorporate a new posture or gesture. If the posture or gesture set is known beforehand this is not an issue, but if postures and gestures are likely to change dynamically as the system develops, a neural network is probably not appropriate.

- *Strengths*
  - Can be used in either a vision- or glove-based solution

  - Can recognize medium posture or gesture sets
  - With adequate training, high accuracy can be achieved

- *Weaknesses*
  - Network training can be very time consuming and does not guarantee good results
  - Requires retraining of the entire network if hand postures or gestures are added or removed

C.2.3.2 Causal Analysis

Overview. Causal analysis is a vision-based recognition technique that stems from work in scene analysis [18]. The technique attempts to extract information from a video stream by using high-level knowledge about actions in the scene and how they relate to one another and the physical environment. Examples of causal data (e.g. the underlying physics of the scene) include rigidity, mass, friction, balance, and work against gravity. Brand uses this information in his system, BUSTER, that understands and outputs information about the structure and stability of block towers [18]; In Brand [17] a more detailed description of BUSTER and other scene analysis systems is given.

Details and Examples. Brand and Essa have applied causal analysis to vision-based gesture recognition [16]. By using knowledge about body kinematics and dynamics, features recovered from the video stream can be used to identify gestures based on human motor plans. The system captures information on shoulder, elbow and wrist joint positions in the image plane. From these positions, the system extracts a feature set that includes wrist acceleration and deceleration, work done against gravity, size of gesture, area between arms, angle between forearms, nearness to body, and verticality. Gesture filters normalize and combine the features and use causal knowledge of how humans interact with objects in the physical world to recognize gestures such as opening, lifting, patting, pushing, stopping, and clutching.

Analysis. Causal analysis in gesture recognition is an interesting concept, but Brand and Essa's discussion of their implementation is cursory [16] and, as a result, it is unclear how accurate their system is. This system also has the disadvantage of not using data from the fingers. More research needs to be conducted in order to determine if this technique is robust enough to be used in any nontrivial applications.

- *Strengths*
  - Uses information about how humans interact with the physical world to help identify gestures
- *Weaknesses*
  - Uses only a limited set of gestures
  - Does not use hand orientation and position data or finger data
  - Does not run in real time
  - Implementation is unclear[16]

Appendix D

Flex and Pinch Input Electronics

D.1 Flex and Pinch Components

The Flex and Pinch input system is made up of two parts; the pinching component and the electronics unit. Conductive cloth and standard wires were used to construct the pinching component while Table D.1 shows the parts that were used in constructing the electronics unit.

D.2 Design and Implementation of Electronics

We use the Microchip PIC processor [110] as the primary means of interfacing the touch sensors with the rest of the system. The low cost and simple programming of these chips made them suitable for the task. The 16C63 [127] provided a UART for serial communications with the workstation and enough I/O pins to allow the touch sensors to be monitored

PART	USAGE
PIC16C63	8 bit microcontroller with built in UART primary interface chip
16x 20K ohm resistors	pull up resistors
16x 2K ohm resistors	protection resistors
16x 1000pF capacitors	protection capacitors
LT1081	RS232 driver/receiver converts 5 volt PIC output to RS232 levels

Table D.1: The listed parts that make up the Flex and Pinch electronics unit.

without extra glue logic. The output pins of the micro-controller were protected from electrostatic discharge with a resistor capacitor network. Additionally, an RS232 driver chip was needed to step the five volt output of the PIC to RS232 line levels.

All 163 possible non-redundant contact possibilities between pairs of wires are reported by separate keycodes. The microcode driver reports separate keycodes for wire connections while the driver on the workstation infers contacts between more than two wires. For example, if contacts one, two, and three are touching, the microcontroller reports that one and two are touching by issuing one keycode, one and three are touching by issuing another keycode, and that two and three are also touching by issuing a third keycode. The driver software determines that there are actually three wires that are all touching. This lowers the amount of memory needed on the microcontroller, and makes the software simpler and faster.

D.3 Electronics Pseudocode

This pseudocode represents the code for the PIC processor on the electronics box. Each short possibility has a byte allocated to it to represent the status (short or unshort) and a timer to determine whether the short has lasted long enough to transmit. This implementation cuts down on noise and bouncing problems.

Algorithm *1*

```
initializeMemory()
for each pin
    do set a voltage on pin;
        for each (otherpin > pin)
            do check for voltage on otherpin;
                if (pin status changed)
                        increment keycode timer;
                if (timer expired)
                        set keycode status;
                        transmit status change;
```

Bibliography

[1] Agarwal, R., and J. Sklansky. Estimating Optical Flow from Clustered Trajectories in Velocity-Time. In *Proceedings of the 11th IAPR International Conference on Pattern Recognition.* Vol. #1. Conference A: Computer Vision and Applications, 215-219, 1992.

[2] Aha, David W., Dennis Kibler, and Marc K. Albert. Instance-based Learning Algorithms. *Machine Learning*, 6, 37-66, 1991.

[3] Anderson, James A. *An Introduction to Neural Networks.* Bradford Books, Boston, 1995.

[4] Ando, H., Y. Kitahara, and N. Hataoka. Evaluation of Multimodal Interface using Spoken Language and Pointing Gesture on Interior Design System. In *International Conference on Spoken Language Processing*, 1994, 567-570.

[5] Ascension Technology Corporation. The Flock of Birds Installation and Operation Guide. Burlington, Vermont, 1996.

[6] Auer, T., A. Pinz, and M. Gervautz. Tracking in a Multi-User Augmented Reality System. In *Proceedings of the First IASTED International Conference on Computer Graphics and Imaging*, 249-253, 1998.

[7] Banarse, D. S. Hand Posture Recognition with the Neocognitron Network. School of Electronic Engineering and Computer Systems, University College of North Wales, Bangor, 1993.

[8] Baudel, Thomas, and Michel Beaudouin-Lafon. Charade: Remote Control of Objects Using Free-Hand Gestures. *Communications of the ACM*, 36(7):28-35, 1993.

[9] Blake, Andrew, and Michael Isard. 3D Position, Attitude and Shape Input Using Video Tracking of Hands and Lips. In *Proceedings of SIGGRAPH'94*, ACM Press, 185-192, 1994.

[10] Billinghurst, M., J. Savage, P. Oppenheimer, and C. Edmond. The Expert Surgical Assistant: An Intelligent Virtual Environment with Multimodal Input. In *Proceedings of Medicine Meets Virtual Reality IV*, 1995, 590-607.

[11] Birk, Henrik, Thomas B. Moeslund, and Claus B. Madsen. Real-Time Recognition of Hand Alphabet Gestures Using Principal Component Analysis. In *Proceedings of The 10th Scandinavian Conference on Image Analysis*, 1997.

[12] Birk, Henrik and Thomas B. Moeslund. *Recognizing Gestures from the Hand Alphabet Using Principal Component Analysis.* Master's thesis, Aalborg University, 1996.

[13] Bolt, R. A. Put That There: Voice and Gesture at the Graphics Interface. In *Proceedings of SIGGRAPH'80*, ACM Press, 262-270, 1980.

[14] Bolt, R. A., and E. Herranz. Two-Handed Gesture in Multi-Modal Natural Dialog. In *Proceedings of the Fifth Annual ACM Symposium on User Interface Software and Technology*, 1992, 7-14.

[15] Bowman, D. Conceptual Design Space - Beyond Walk-Through to Immersive Design, In *Designing Digital Space*, D. Bertol (ed.), Jon Wiley and Sons, New York, 1996.

[16] Brand, Matthew, and Irfan Essa. Causal Analysis for Visual Gesture Understanding. MIT Media Laboratory Perceptual Computing Section Technical Report No. 327, 1995.

[17] Brand, Matthew. *Explanation-Mediated Vision: Making Sense of the World with Causal Analysis.* Ph.D dissertation, Northwestern University, 1994.

[18] Brand, Matthew, Lawrence Birnbaum, and Paul Cooper. Sensible Scenes: Visual Understanding of Complex Structures Through Causal Analysis. In *Proceedings of the 1993 AAAI Conference*, 49-56, 1993.

[19] Broomhead, D., and D. Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321-355, 1988.

[20] Bryson, Steve. Virtual Reality in Scientific Visualization. *Communications of the ACM*, 39(5):62-71, 1996.

[21] Bryson, S., S. Johan, and L. Schlecht. An Extensible Interactive Visualization Framework for the Virtual Windtunnel. In *Proceedings of the Virtual Reality Annual International Symposium*, 1997, 106-113.

[22] Butterworth, B., and G. Beattie. Gesture and Silence as Indicators of Planning in Speech. In *Recent Advances in the Psychology of Language*, Campbell and Smith (eds.), Plenum Press, New York, 1978.

[23] Carpenter, R. *The Logic of Typed Feature Structures*, Cambridge University Press, Cambridge, England, 1992.

[24] Cassell, Justine. A Framework for Gesture Generation and Interpretation. In *Computer Vision in Human-Machine Interaction.* R. Cipolla and A. Pentland (eds.), Cambridge University Press, forthcoming.

[25] Charniak, Eugene. *Statistical Language Learning.* MIT Press, Cambridge, 1993.

[26] Cheyer, A. and L. Julia. Multimodal Maps: An Agent-based Approach. *Lecture Notes in Artificial Intelligence 1374: Multimodal Human-Computer Communication*, (eds.) H. Bunt, R. J. Beun, and T. Borghuis, 1998, 111-121.

[27] Cohen, P. R., M. Johnston, D. McGee, S. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. QuickSet:Multimodal Interaction for Distributed Applications. In *Proceedings of the Fifth Annual International Multimodal Conference*, 1997, 31-40.

[28] Coquillart, S. and G. Wesche. The Virtual Palette and the Virtual Remote Control Panel: A Device and Interaction Paradigm for the Responsive Workbench. *IEEE VR'99*, 213-217, 1999.

[29] Cootes, T.F., C.J. Taylor, D.H. Cooper, and J. Graham. Active Shape Models – Their Training and Applications. *Computer Vision and Image Understanding* , 61(2), January 1995.

[30] Cootes, T.F., and C.J. Taylor. Active Shape Models – 'Smart Snakes'. In *Proceedings of the British Machine Vision Conference*, Springer-Verlag, 266-275, 1992.

[31] Cruz-Neira, C., D. J. Sandin, T. A. Defanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Proceedings of SIGGRAPH'93*, 1993, 135-142.

[32] Davis, James William. *Appearance-Based Motion Recognition of Human Actions.* Master's thesis, Massachusetts Institute of Technology, 1996.

[33] Davis, James William, and Mubarak Shah. Gesture Recognition. Technical Report, Department of Computer Science, University of Central Florida, CS-TR-93-11, 1993.

[34] Defanti, Thomas, and Daniel Sandin. Final Report to the National Endowment of the Arts. US NEA R60-34-163, University of Illinois at Chicago Circle, Chicago, Illinois, 1977.

[35] Darrell, Trevor J., and Alex P. Pentland. Recognition of Space-Time Gestures Using a Distributed Representation. MIT Media Laboratory Vision and Modeling Group Technical Report No. 197, 1993.

[36] Dorner, Brigitte. *Chasing the Colour Glove: Visual Hand Tracking.* Master's thesis, Simon Fraser University, 1994.

[37] Duda, Richard O., and Peter E. Hart. *Pattern Classifiaction and Scene Analysis.* Wiley-Interscience Publications, Stanford Research Institute, Menlo Park, 1973.

[38] Efron, D. *Gesture and Environments.* King's Crown Press, Morningside Heights, New York, 1941.

[39] Eglowstein, Howard. Reach Out and Touch Your Data. *Byte*, 283-290, July 1990.

[40] Encarnação, M. A Survey on Input Technology for the Virtual Table Interface Device. Technical Report, Fraunhofer Center for Research in Computer Graphics, Inc., 1997.

[41] Everitt, B.S. *Cluster Analysis.* John Wiley and Sons, New York, 1974.

[42] Fakespace. Pinch^{TM} Glove System Installation Guide and User Handbook, Mountain View, California, 1997.

[43] Fels, Sidney, and Geoffrey Hinton. Glove-TalkII: An Adaptive Gesture-to-Format Interface. In *Proceedings of CHI'95 Human Factors in Computing Systems*, ACM Press, 456-463, 1995.

[44] Fels, Sidney. *Glove-TalkII: Mapping Hand Gestures to Speech Using Neural Networks – An Approach to Building Adaptive Interfaces.* Ph.D. dissertation, University of Toronto, 1994.

[45] Fifth Dimension Technologies. http://www.5dt.com/products.html, 1999.

[46] Forsberg, A., LaViola, J., and Zeleznik, R. Incorporating Speech Input into Gesture-Based Graphics Applications at The Brown University Graphics Lab, In *The CHI'99 Workshop on Designing the User Interface for Pen and Speech Multimedia Applications*, May 1999.

[47] Forsberg, A., LaViola J., and Zeleznik, R. "ErgoDesk: A Framework for Two and Three Dimensional Interaction at the ActiveDesk." In *Proceedings of the Second International Immersive Projection Technology Workshop*, Ames, Iowa, May 11-12, 1998.

[48] Forsberg, Andrew S., Joseph J. LaViola, Lee Markosian, Robert C. Zeleznik. Seamless Interaction in Virtual Reality *IEEE Computer Graphics and Applications*, 17(6), 1997, 6-9.

[49] Forsberg, Andrew S., Kenneth Herndon, and Robert C. Zeleznik. Apeture Based Selection for Immersive Virtual Environments. In *Proceedings of the 1996 ACM Symposium on User Interface Software and Technology*, 1996.

[50] Freeman, William T., and Craig D. Weissman. Television Control by Hand Gestures. Technical Report, Mitsubishi Electronic Research Laboratories, TR-94-24. 1994.

[51] Fukushima, Kunihiko. Analysis of the Process of Visual Pattern Recognition by the Neocognitron. *Neural Networks*, 2:413-420, 1989.

[52] General Reality Company. GloveGRASP^{TM} User's Guide. San Jose, California, 1996.

[53] Glassner, Andrew. *Principles of Digital Image Synthesis.* Morgan Kaufman, San Francisco, 1995.

[54] Grimes, G. Digital Data Entry Glove Interface Device. Bell Telephone Laboratories, Murray Hill, New Jersey. US Patent Number 4,414,537, 1993.

[55] Grimson, W.E. *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, Boston, 1990.

[56] Hand, Chris, Ian Sexton, and Michael Mullan. A Linguistic Approach to the Recognition of Hand Gestures. In *Proceedings of the Designing Future Interaction Conference*, University of Warwick, UK, 1994.

[57] Heap, Tony. http://www.scs.leeds.ac.uk/vislib/proj/ajh/welcome.html, 1999.

[58] Heap, A. J., and F. Samaria. Real-Time Hand Tracking and Gesture Recognition Using Smart Snakes. In *Proceedings of Interface to Real and Virtual Worlds*, Montpellier, 1995.

[59] K. Hinkley, R. Pausch, J.C. Goble, and N.F. Kassel. A Survey of Design Issues in Spatial Input. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1994, 213-222.

[60] Hill, Lewis C., Chan, Chiu-Shui Chan, and Carolina Cruz-Niera. Virtual Architectural Design Tool (VADet), In *The Third International Immersive Projection Technology Workshop*, Stuttgart, Germany, 1999.

[61] Huang, X. D., Y. Ariki, and M. A. Jack. *Hidden Markov Models for Speech Recognition.* Edinburgh University Press, Edinburgh, 1990.

[62] InterSense. http://www.isense.com, 1999.

[63] Johnston, M., P. R. Cohen, D. McGee, S. L. Oviatt, J. A. Pittman, and I. Smith. Unification-based Multimodal Integration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 1997.

[64] Jolliffe, I. T. *Principal Component Analysis.* Springer-Verlag, New York, 1986.

[65] Kadous, Waleed. *GRASP: Recognition of Australian Sign Language Using Instrumented Gloves.* Bachelor's thesis, University of New South Wales, 1995.

[66] Kendon, Adam. Current Issues in the Study of Gesture. In *The Biological Foundations of Gestures: Motor and Semiotic Aspects.* Nespoulous, Perron, and Lecours (eds.), Lawrence Erlbaum Associates, Hillsday, NJ, 1986.

[67] Kessler, G. Drew, Larry H. Hodges, and Neff Walker. Evaluation of the CyberGlove as a Whole Hand Input Device. In *ACM Transactions on Computer-Human Interaction*, 2(4):263-283, 1995.

[68] Kohler, Marcus. Special Topics of Gesture Recognition Applied to Intelligent Home Environments. In *Proceedings of the International Gesture Workshop'97*, Berlin, 285-297, 1997.

[69] Koons, D. B., C. J. Sparrell, and K. R. Thorisson. Integrating Simultaneous Input from Speech, Gaze, and Hand Gestures. *Intelligent Multimedia Interfaces*, (ed.) Mark T. Maybury, 1993, 257-279.

[70] Kramer, James. Communication System for Deaf, Deaf-blind, and Non-vocal Individuals Using Instrumented Gloves. Virtual Technologies, US Patent Number 5,047,952, 1991.

[71] Kramer, James, and Larry Leifer. The Talking Glove: An Expressive and Receptive 'Verbal' Communication Aid for the Deaf, Deaf-blind, and Non-vocal. Technical Report, Department of Electrical Engineering, Stanford University, 1989.

[72] Krose, Ben J. A., and P. Patrick van der Smagt. *An Introduction to Neural Networks.* University of Amsterdam, 1995.

[73] Krueger, Myron W. *Artificial Reality II.* Addison-Wesley Publishing Company, New York, 1991.

[74] Kuno, Yoshinori, Tomoyuki Ishiyama, Kang-Hyun Jo, Nobutaka Shimada and Yoshiaki Shirai. Vision-Based Human Interface System: Selectively Recognizing Intentional Hand Gestures. In *Proceedings of the First IASTED International Conference on Computer Graphics and Imaging*, 219-223, 1998.

[75] Lakoff, G., and M. Johnson. *Metaphors We Live By.* University of Chicago Press, Chicago, 1980.

[76] LaViola, Joseph. A Multimodal Interface Framework For Using Hand Gestures and Speech in Virtual Environment Applications. To appear in *Lecture Notes in Artifical Intelligence: The Gesture Workshop'99*, Gif-sur-Yvette, 1999.

[77] LaViola, Joseph, and Robert Zeleznik. Flex and Pinch: A Case Study of Whole Hand Input Design for Virtual Environment Interaction. In *Proceedings of the Second IASTED International Conference on Computer Graphics and Imaging*, 221-225, 1999.

[78] LaViola, Joseph, and Timothy Miller. Hand Posture and Gesture Techniques and Technology. Submitted to *Computing Surveys*, ACM Press, 1999.

[79] LaViola, Joseph. A Survey of Hand Posture and Gesture Recognition Techniques and Technology, Technical Report CS-99-11, Brown University, Department of Computer Science, Providence RI, June 1999.

[80] LaViola, Joseph, Andrew S. Forsberg, Robert C. Zeleznik. Jot: A Framework for Interface Research. *IBM's interVisions Online Magazine*, Issue No. 11, http://www.alphaworks.ibm.com, February, 1998.

[81] Lee, Christopher, and Yangsheng Xu. Online Interactive Learning of Gestures for Human/Robot Interfaces. In *1996 IEEE International Conference on Robotics and Automation*, vol. #4, 2982-2987, 1996.

[82] Liang, Rung-Huei, and Ming Ouhyoung. A Sign Language Recognition System Using Hidden Markov Model and Context Sensitive Search. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology'96*, ACM Press, 59-66, 1996.

[83] Lu, Shan, Seiji Igi, Hideaki Matsuo, and Yuji Nagashima. Towards a Dialogue System Based on Recognition and Synthesis of Japanese Sign Language. In *Proceedings of the International Gesture Workshop'97*, Berlin, 259-272, 1997.

[84] Lucente, Mark, Gert-Jan Zwart, and Andrew D. George. Visualization Space: A Testbed for Deviceless Multimodal User Interface. In *Intelligent Environments 98*, AAAI Spring Symposium Series, 87-92, 1998.

[85] Makower, J., M. Parnianpour, and M. Nordin. The Validity Assessment of the Dextrous Hand Master: A Linkage System for the Measurement of the Joints of the Hand. In *Abstracts of the First World Congress of Biomechanics.* (Volume #2), La Jolla, California, 338, 1990.

[86] Mann, Steve. Wearable Computing: A First Step Toward Personal Imaging, *IEEE Computer*, 30(2): 25-32, 1997.

[87] Mapes, Daniel J., and Michael J. Moshell. A Two-Handed Interface for Object Manipulation in Virtual Environments. In *PRESENSE: Teleoperators and Virtual Environments*, 4(4):403-416, 1995.

[88] Marcus, Beth A., and Philip J. Churchill. Sensing Human Hand Motions for Controlling Dexterous Robots. In *The Second Annual Space Operations Automation and Robotics Workshop*, Wright State University, June 20-23, 1988.

[89] Martin, J. C. TYCOON: Theoretical Framework and Software Tools for Multimodal Interfaces. *Intelligence and Multimodality in Multimedia interfaces.* (ed.) John Lee, AAAI Press, 1998.

[90] Martin, J. C., R. Veldman, and D. Beroule. Developing Multimodal Interfaces: A Theoretical Framework and Guided Propagation Networks. *Lecture Notes in Artificial Intelligence 1374: Multimodal Human-Computer Communication*, (eds.) H. Bunt, R. J. Beun, and T. Borghuis, 1998, 158-187.

[91] Martin, Jerome, and James L. Crowley. An Appearance-Based Approach to Gesture Recognition. In *Proceedings of the Ninth International Conference on Image Analysis and Processing*, 340-347, 1997.

[92] Matsuo, Hideaki, Seiji Igi, Shan Lu, Yuji Nagashima, Yuji Takata, and Terutaka Teshima. The Recognition Algorithm with Non-contact for Japanese Sign Language Using Morphological Analysis. In *Proceedings of the International Gesture Workshop'97*, Berlin, 273-284, 1997.

[93] Maybeck, Peter S. *Stochastic Models, Estimation and Control.* Volume 1, Academic Press, Inc, 1979.

[94] Mehrotra, Kishan, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of Artificial Neural Networks.* The MIT Press, Boston, 1997.

[95] Mine, Mark. Moving Objects In Space: Exploiting Proprioception In Virtual Environment Interaction. In *Proceedings of SIGGRAPH'97*, ACM Press, 19-26, 1997.

[96] Mitchell, Tom M. *Machine Learning.* McGraw-Hill, Boston, 1997.

[97] Mulder, Axel. Human Movement, Tracking Technology. Technical Report, School of Kinesiology, Simon Fraser University, 94-1, 1994.

[98] Multigen. SmartScene^{TM} Video Clip, Discovery Channel's NextStep program, 1998.

[99] Murakami, Kouichi, and Hitomi Taguchi. Gesture Recognition Using Recurrent Neural Networks. In *Proceedings of CHI'91 Human Factors in Computing Systems*, 237-242, 1991.

[100] Nam, Yanghee, and KwangYun Wohn. Recognition of Space-Time Hand-Gestures Using Hidden Markov Model. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology'96*, ACM Press, 51-58, 1996.

[101] Nespoulous, J., and A. R. Lecours. Gestures, Nature and Function. In *The Biological Foundations of Gestures: Motor and Semiotic Aspects.* Nespoulous, Perron, and Lecours (eds.), Lawrence Erlbaum Associates, Hillsday, NJ, 1986.

[102] Newby, Gregory B. Gesture Recognition Using Statistical Similarity. In *Proceedings of Virtual Reality and Persons with Disabilities*, 1993.

[103] Nissho Electronics Corporation. Introduction to SuperGlove. Tokyo, Japan, 1997.

[104] Oviatt, Sharon. Ten Myths of Multimodal Interaction. In *Communications of the ACM*, 42(11):74-81, 1999.

[105] Oviatt, S., A. DeAngeli, and K. Kuhn. Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction. In *Proceedings of CHI'97 Human Factors in Computing Systems*, 1997, 415-422.

[106] Oviatt, Sharon, and Robert VanGent. Error Resolution During Multimodal Human-Computer Interaction. In *Proceedings of the International Conference on Spoken Language Processing*, 204-207, 1996.

[107] Oviatt, S., and E. Olsen. Integration Themes in Multimodal Human-Computer Interaction. In *Proceedings of the 1994 International Conference on Spoken Language Processing*, 1994, 551-554.

[108] Papper, M., and M. Gigante. Using Gestures to Control a Virtual Arm. In *Virtual Reality Systems*, R. Earnshaw, H. Jones, and M. Gigante (eds.), Academic Press, London, 1993.

[109] Pausch, Randy. Virtual Reality on Five Dollars a Day. Technical Report CS-91-21, Department of Computer Science, University of Virginia, 1991.

[110] `http://www.microchip.com/10/Lit/ PICmicro/index.htm.`

[111] J.S. Pierce, A.S. Forsberg, M.J. Conway, S. Hong, R.C. Zeleznik, and M.R. Mine. Image Plane Interaction Techniques in 3D Immersive Environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, 1997, 39-44.

[112] Polhemus. http://www.polhemus.com, 1999.

[113] Quek, Francis K.H. Unencumbered Gestural Interaction. *IEEE Multimedia*, 4(3):36-47, 1996.

[114] Quek, Francis K.H and Meide Zhao. Inductive Learning in Hand Pose Recognition. In the *Second International Conference on Automatic Face and Gesture Recognition.*, 78-83, 1996.

[115] Quek, Francis K.H. Toward a Vision-Based Gesture Interface. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology'94*, ACM Press, 17-31, 1994.

[116] Rabiner, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE* 77(2):267-296, 1989.

[117] Rabiner, L. R., and B.H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 4-16, January 1986.

[118] Rangarajan, K., and M. Shah. Establishing Motion Correspondence. *CVGIP: Image Understanding*, 54:56-73, 1991.

[119] Raskar, Ramesh, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. In *Proceedings of SIGGRAPH'98*, ACM Press, 179-188, 1998.

[120] Rehag, James M., and Takeo Kanade. DigitEyes: Vision-Based Human Hand Tracking. Technical Report CMU-CS-93-220, School of Computer Science, Carnegie Mellon University, 1993.

[121] Rubine, Dean. Specifing Gestures by Example. In *Proceedings of SIGGRAPH'91*, ACM Press, 329-337, 1991.

[122] Rudnicky, Alexander I., Alexander G. Hauptman, and Kai-Fu Lee. Survey of Current Speech Technology. In *Communications of the ACM*, 37(3):52-57, 1994.

[123] Russell, Stuart, and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cliffs, NJ, 1995.

[124] Schkolne, Steven, and Peter Schröder. Surface Drawing. Technical Report CS-TR-99-03, Caltech Department of Computer Science, 1999.

[125] Schlenzig, Jennifer, Edward Hunter, and Ramesh Jain. Recursive Spatio-Temporal Analysis: Understanding Gestures. Technical Report VCL-95-109, Visual Computing Laboratory, University of California, San Diego, 1995.

[126] Sirovich, I., and M. Kirby. Low-dimensional Procedure for the Characterization of Human Faces. *Journal of the Optical Society of America*, 4(3):519-524, 1987.

[127] `http://www.microchip.com/10/Lit/ PICmicro/16C6X/index.htm.`

[128] Starner, Thad, and Alex Pentland. Real-Time American Sign Language Recognition from Video Using Hidden Markov Models. MIT Media Laboratory Perceptual Computing Section Technical Report No. 375, 1996.

[129] Starner, Thad. *Visual Recognition of American Sign Language Using Hidden Markov Models.* Master's thesis, Massachusetts Institute of Technology, 1995.

[130] State, Andrei, Hirota Gentaro, David T. Chen, William F. Garrett, and Mark A. Livingston. Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking. In *Proceedings of SIGGRAPH'96*, ACM Press, 429-438, 1996.

[131] Sturman, David J., and David Zeltzer. A Survey of Glove-based Input. *IEEE Computer Graphics and Applications*, 14(1):30-39, 1994.

[132] Sturman, David J., and David Zeltzer. A Design Method for 'Whole-Hand' Human-Computer Interaction. In *ACM Transactions on Information Systems*, 11(3):219-238, 1993.

[133] Sturman, David J. *Whole-hand Input.* Ph.D dissertation, Massachusetts Institute of Technology, 1992.

[134] Sturman, David J., David Zeltzer, and Steve Pieper. Hands-on Interaction with Virtual Environments. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology'89*, ACM Press, 19-24, 1989.

[135] Takahashi, Tomoichi, and Fumio Kishino. Hand Gesture Coding Based on Experiments Using a Hand Gesture Interface Device. *SIGCHI Bulletin* 23(2):67-73, 1991.

[136] Turk, M., and A. Pentland. Eigenfaces for Recognition. *Journal of Neuroscience*, 3(1):71-86, 1991.

[137] Tung, C. P., and A. C. Kak. Automatic Learning of Assembly Tasks Using a Dataglove System. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 1-8, 1995.

[138] Umeki, Naoko, Akira Morishita, Shunichi Numazaki, Yasunobu Yamauchi, Isao Mihara, and Miwako Doi.. A Motional Interface Approach Based on User's Tempo. In *Proceedings of CHI'99 Human Factors in Computing Systems, Extended Abstracts*, ACM Press, 23-24, 1999.

[139] Utsumi, Akira, Jun Kurumisawa, Takahiro Otsuka, and Jun Ohya. Direct Manipulation Scene Creation in 3D. SIGGRAPH'97 Electronic Garden, 1997.

[140] van Dam, Andries. Post-WIMP User Interfaces. *Communications of the ACM*, 40(2), 1997, 63-67.

[141] Veltman, S. R., and R. Prasad. Hidden Markov Models Applied to On-line Handwritten Isolated Character Recognition. In *IEEE Transactions on Image Processing*, 314-318, 1994.

[142] Virtual Technologies. CyberGlove^{TM} User's Manual. Palo Alto, California, 1993.

[143] Vo, M. T., and A. Waibel. A Multimodal Human-Computer Interface: Combination of Gesture and Speech Recognition. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems – Adjunct Proceedings*, 1993, 69-70.

[144] Waibel, A., M. T. Vo, P. Duchnowski, and S. Manke. Multimodal Interfaces. *Artificial Intelligence Review*, Special Volume on Integration of Natural Language and Vision Processing, Mc Kevitt, P. (Ed.), 1995, 299-319.

[145] Watson, Richard. A Survey of Gesture Recognition Techniques. Technical Report TCD-CS-93-11, Department of Computer Science, Trinity College Dublin, 1993.

[146] Welch, Greg, and Gary Bishop. SCAAT: Incremental Tracking with Incomplete Information. In *Proceedings of SIGGRAPH'97*, ACM Press, 333-345, 1997.

[147] Welch, Greg, and Gary Bishop. An Introduction to the Kalman Filter. Technical Report TR 05-041, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[148] Weimer, D. and S. K. Ganapathy. Interaction Techniques Using Hand Tracking and Speech Recognition. In *Multimedia Interface Design*, Meera M. Blattner and Roger B. Dannenberg, (eds.), Addison-Wesley Publishing Company, New York, 109-126, 1992.

[149] Wexelblat, Alan. An Approach to Natural Gesture in Virtual Environments. In *ACM Transactions on Computer-Human Interaction*, 2(3):179-200, 1995.

[150] Wexelblat, Alan. *A Feature-Based Approach to Continuous-Gesture Analysis.* Master's thesis, Massachusetts Institute of Technology, 1994.

[151] Wilson, Andrew D., Aaron F. Bobick and Justine Cassell. Recovering the Temporal Structure of Natural Gesture. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996.

[152] Wise, Sam, William Gardner, Eric Sabelman, Erik Valainis, Yuriko Wong, Karen Glass, John Drace, and Joseph Rosen. Evaluation of a Fiber Optic Glove for Semiautomated Goniometric Measurements. *Journal of Rehabilitation Research and Development*, 27(4): 411-424, 1990.

[153] Youngblut, C., R.E. Johnson, S.H. Nash, R.A. Wienclaw, and C.A. Will. Review of Virtual Environment Interface Technology. Technical Report IDA Paper P-3186, Log: H96-001239. Institute for Defense Analysis, 1996.

[154] Zeleznik, Robert C., Andrew S. Forsberg, and Paul S. Strauss. Two Pointer Input For 3D Interaction. In *Proceedings of the Symposium on Interactive 3D Graphics*, 1997, 115-120.

[155] Zimmerman, Thomas G., Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A Hand Gesture Interface Device. In *Proceedings of CHI+GI'87 Human Factors in Computing Systems and Graphics Interface*, ACM Press, 189-192, 1987.

[156] Zimmerman, Thomas G. Optical Flex Sensor. US Patent Number 4,542,291, 1985.