

**APPLYING VISION TO INTELLIGENT HUMAN-COMPUTER  
INTERACTION**

by

Guangqi Ye

A dissertation submitted to The Johns Hopkins University in conformity with the  
requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

October, 2005

© Guangqi Ye 2005

All rights reserved

# Abstract

As powerful and affordable computers and sensors become virtually omnipresent, constructing highly intelligent and convenient computation systems has never been so promising. Vision holds great promise in building advanced human-computer interaction (HCI) systems. We investigate different techniques to integrate passive vision into various interaction environments.

First, we propose a novel approach to integrate visual tracking into a haptics systems. Traditional haptic environments require that the user must be attached to the haptic device at all times, even though force feedback is not always being rendered. We design and implement an augmented reality system called VisHap that uses visual tracking to seamlessly integrate force feedback with tactile feedback to generate a “complete” haptic experience. The VisHap framework allows the user to interact with combinations of virtual and real objects naturally, thereby combining active and passive haptics. The flexibility and extensibility of our framework is promising in that it supports many interaction modes and allows further integration with other augmented reality systems.

Second, we propose a new methodology for vision-based human-computer interaction called the Visual Interaction Cues (VICs) paradigm. VICs is based on the concept of sharing perceptual space between the user and the computer. Each interaction component is represented as a localized region in the image(s). We propose to model gestures based on the streams of extracted visual cues in the local space, thus avoiding the problem of globally tracking the user(s). Efficient algorithms are proposed to capture hand shape and motion. We investigate different learning and modeling techniques including neural networks, Hidden Markov Models and Bayesian

classifiers to recognize postures and dynamic gestures.

Since gestures are in essence a language with individual low-level gestures analogous to a word in conventional languages, a high-level gesture language model is essential for robust and efficient recognition of continuous gesture. To that end, we have constructed a high-level language model that integrates a set of low-level gestures into a single, coherent probabilistic framework. In the language model, every low-level gesture is called a gesture word and a composite gesture is a sequence of gesture words, which are contextually and temporally constrained. We train the model via supervised and unsupervised learning techniques. A greedy inference algorithm is proposed to allow efficient online processing of continuous gestures. We have designed a large-scale gesture experiment that involves sixteen subjects and fourteen gestures. The experiment shows the robustness and efficacy of our system in modeling a relative large gesture vocabulary involving many users. Most of the users also consider our gesture system as comparable or more natural and comfortable than traditional user interfaces with a mouse.

Advisor: Dr. Gregory D. Hager

Readers: Dr. Gregory D. Hager

Dr. Darius Burschka

Dr. Izhak Shafran

# Contents

|   |            |
|---|------------|
| <b>Abstract</b>   | <b>ii</b>  |
| <b>List of Figures</b>  | <b>vii</b> |
| <b>List of Tables</b>   | <b>x</b>   |
| <b>Acknowledgements</b>   | <b>xii</b> |
| <b>1 Introduction</b>   | <b>1</b>   |
| 1.1 Background . . . . .  | 1          |
| 1.2 Motivation . . . . .  | 3          |
| 1.2.1 Vision In Multimodal Interfaces . . . . .   | 4          |
| 1.2.2 Vision-Based Interaction . . . . .  | 4          |
| 1.2.3 Gestures for VBI . . . . .  | 6          |
| 1.3 Thesis Statement . . . . .  | 8          |
| 1.4 Overview . . . . .  | 8          |
| 1.5 Dissertation Contributions . . . . .  | 10         |
| 1.5.1 Contribution 1: Effective Integration of Vision and Haptics . . . . .                                     | 10         |
| 1.5.2 Contribution 2: Novel Methodology for Applying Computer Vision<br>to Human-Computer Interaction . . . . . | 10         |
| 1.5.3 Contribution 3: Efficient Motion Capture for Gesture Recognition . . . . .                                | 11         |
| 1.5.4 Contribution 4: Unified Gesture Modeling for Multimodal Gestures . . . . .                                | 11         |
| 1.6 Notation . . . . .  | 13         |
| 1.7 Relevant Publications . . . . .   | 13         |
| <b>2 Vision for Natural Interaction: A Review</b>   | <b>15</b>  |
| 2.1 Vision and Multimodal Interface . . . . .   | 15         |
| 2.1.1 Mutltimodal Interface . . . . .   | 15         |
| 2.1.2 Integrating Vision into Multimodal Interface . . . . .  | 17         |
| 2.1.3 Vision and Post-WIMP Interfaces . . . . .   | 18         |
| 2.2 Vision-Based HCI . . . . .  | 19         |
| 2.2.1 Full Body Motion . . . . .  | 20         |
| 2.2.2 Head and Face Motion . . . . .  | 20         |
| 2.3 Visual Modeling of Gestures for HCI . . . . .   | 21         |

|          |  |           |
|----------|--|-----------|
| 2.3.1    | Definition and Taxonomy of Gestures . . . . .                                | 21        |
| 2.3.2    | Modeling Temporal Structure . . . . .  | 23        |
| 2.3.3    | Modeling the Spatial Structure . . . . .                                     | 24        |
| 2.3.4    | Hand Detection and Motion Capturing . . . . .                                | 25        |
| 2.3.5    | Visual Feature Extraction . . . . .  | 26        |
| 2.3.6    | Gesture Recognition . . . . .  | 27        |
| 2.4      | Applications of Gesture Interfaces . . . . .                                 | 28        |
| 2.5      | Conclusions . . . . .  | 30        |
| <b>3</b> | <b>Augmented Reality Combining Haptics and Vision</b>                        | <b>32</b> |
| 3.1      | Introduction . . . . .   | 33        |
| 3.1.1    | Related Work . . . . .   | 35        |
| 3.2      | System Design and Implementation . . . . .                                   | 36        |
| 3.2.1    | Vision Subsystem . . . . .   | 38        |
| 3.2.2    | World Subsystem . . . . .  | 42        |
| 3.2.3    | Haptic Subsystem . . . . .   | 43        |
| 3.3      | Experimental Results . . . . .   | 47        |
| 3.4      | Conclusions . . . . .  | 50        |
| <b>4</b> | <b>The Visual Interaction Cues Paradigm</b>                                  | <b>51</b> |
| 4.1      | The VICs Framework . . . . .   | 53        |
| 4.1.1    | Modeling Interaction . . . . .   | 53        |
| 4.1.2    | The VICs Architectural Model . . . . .                                       | 55        |
| 4.1.3    | VICs Implementation . . . . .  | 58        |
| 4.1.4    | Modes of Interaction . . . . .   | 60        |
| 4.1.5    | Robustness and Applicability . . . . .                                       | 61        |
| 4.2      | A Stochastic Extension of the Parser for 2D VICs . . . . .                   | 63        |
| 4.2.1    | Hidden Markov Models . . . . .   | 64        |
| 4.2.2    | Background Modeling and Image Segmentation Based on Hue Histograms . . . . . | 65        |
| 4.2.3    | HMM-based Human Activity Recognition . . . . .                               | 67        |
| 4.2.4    | Experimental Results . . . . .   | 70        |
| 4.2.5    | Action Recognition Result . . . . .  | 71        |
| 4.3      | Conclusions . . . . .  | 72        |
| <b>5</b> | <b>Efficient Capture of 3D Gesture Motion</b>                                | <b>74</b> |
| 5.1      | 4D Touchad: A VICs Platform . . . . .  | 75        |
| 5.1.1    | Camera Calibration and Rectification . . . . .                               | 75        |
| 5.1.2    | Geometric Calibration Between the Display and Images . . . . .               | 76        |
| 5.1.3    | Color Calibration and Foreground Segmentation . . . . .                      | 77        |
| 5.1.4    | Posture Recognition on 4DT . . . . .   | 82        |
| 5.2      | Efficient Motion Capturing of 3D Gestures . . . . .                          | 86        |
| 5.2.1    | Related Work . . . . .   | 87        |
| 5.2.2    | Visual Capturing of 3D Gesture Features . . . . .                            | 87        |
| 5.2.3    | Unsupervised Learning of the Cluster Structures of 3D Features . . . . .     | 89        |

|          |  |            |
|----------|--|------------|
| 5.3      | Modeling Dynamic Gesture Based on 3D Feature . . . . .                             | 92         |
| 5.3.1    | Modeling Dynamic Gestures Using HMMs . . . . .                                     | 92         |
| 5.3.2    | Modeling Dynamic Gestures Using Neural Networks . . . . .                          | 93         |
| 5.3.3    | Gesture Recognition Experiments . . . . .  | 94         |
| 5.4      | Conclusions . . . . .  | 97         |
| <b>6</b> | <b>Robust Modeling of Multimodal Gestures With Heterogeneous Observa-<br/>tion</b> | <b>99</b>  |
| 6.1      | Introduction . . . . .   | 100        |
| 6.1.1    | Related Work in High-Level Gesture Modeling . . . . .                              | 101        |
| 6.2      | Modeling Composite Gestures . . . . .  | 102        |
| 6.2.1    | Definitions . . . . .  | 103        |
| 6.2.2    | The Gesture Language . . . . .   | 103        |
| 6.2.3    | Learning the Gesture Model . . . . .   | 105        |
| 6.2.4    | Inference on the Gesture Model . . . . .   | 107        |
| 6.2.5    | Three Low-Level Gesture Processors . . . . .                                       | 109        |
| 6.3      | Experiments with the High-Level Gesture Model . . . . .                            | 111        |
| 6.3.1    | Gesture Set . . . . .  | 111        |
| 6.3.2    | Implementation of Low-Level GWords . . . . .                                       | 114        |
| 6.3.3    | Experimental Results Using Gesture Model . . . . .                                 | 116        |
| 6.4      | Modeling Gestures Across Multiple Users . . . . .                                  | 119        |
| 6.4.1    | Gesture Vocabulary . . . . .   | 119        |
| 6.4.2    | Experimental Setup . . . . .   | 122        |
| 6.4.3    | Implementation of Low-Level Gestures . . . . .                                     | 125        |
| 6.5      | Results of Large-Scale Gesture Experiments . . . . .                               | 128        |
| 6.5.1    | Experimental Results of Modeling Low-Level Gestures . . . . .                      | 129        |
| 6.5.2    | Experimental Results of Modeling Composite Gestures . . . . .                      | 134        |
| 6.5.3    | User Feedback Regarding Our Gesture Interface . . . . .                            | 136        |
| 6.6      | Conclusions . . . . .  | 137        |
| <b>7</b> | <b>Conclusions</b>   | <b>139</b> |
| 7.1      | Future Work . . . . .  | 140        |
|          | <b>Bibliography</b>  | <b>142</b> |
|          | <b>Vita</b>  | <b>158</b> |

# List of Figures

|     |   |    |
|-----|---|----|
| 2.1 | Taxonomy of hand gestures. . . . .  | 22 |
| 3.1 | Overview of the VisHap system. . . . .  | 36 |
| 3.2 | An example of background image (left), foreground image (middle) and segmentation result (right). . . . .   | 39 |
| 3.3 | Fingertip detection result: the foreground image and detected fingertip specified with a yellow circle (left) and the candidate fingertips on the segmented foreground image (right). . . . .   | 41 |
| 3.4 | Control scheme in error space. . . . .  | 44 |
| 3.5 | Relationship of force gain $\lambda_z$ and the depth $d$ of the fingertip under a plane (left) or the surface of a button (right). . . . .  | 47 |
| 3.6 | Relationship of haptic force feedback along the normal of the plane surface and the distance of the fingertip to the plane. A negative distance indicates that the finger is under the surface. . . . .   | 48 |
| 3.7 | Relationship of haptic force feedback along the normal of the button surface and the distance of the fingertip to the button. A negative distance indicates that the finger is under the surface. . . . .   | 49 |
| 4.1 | Comparison between conventional VBI and VICs paradigm. Left figure shows the concept of traditional vision-based interface; right figure shows the VICs approach. . . . .   | 53 |
| 4.2 | The icon state model for a WIMP interface. . . . .  | 54 |
| 4.3 | The state model for a VICs-based interface component. . . . .   | 55 |
| 4.4 | Schematic explaining the principle of localized image analysis for the VICs paradigm. Here $M$ is the component mapping that yields a region of interest in the image $\mathcal{I}$ for analyzing actions on component $\mathcal{C}$ . . . . .                  | 56 |
| 4.5 | VICs system architecture and data flow. . . . .   | 59 |
| 4.6 | (left) A VICs-based calculator using a motion-color processor. (middle) Gesture-based demonstration of multiple interaction triggers for a single VI-Con. (right) VICs-based 2D-2D mirror mode interface for a <i>Breakout<sup>TM</sup></i> style game. . . . . | 62 |
| 4.7 | The parallels between a speech processing system and visual “language” processing. . . . .  | 63 |

|      |  |     |
|------|--|-----|
| 4.8  | An example of background image, unsegmented image and segmentation result. The leftmost image is the background. The second image shows when the hand has entered the scene. The final segmentation result is shown in the third image. . . . .  | 66  |
| 4.9  | The definition of feature space for gesture recognition. . . . .   | 68  |
| 4.10 | HMM structures used to model dynamic gestures. The left figure shows the structure of a basic singleton HMM. The right figure illustrates the composite HMM topology that concatenates singleton HMMs. . . . .   | 68  |
| 4.11 | The topology of a simple three-state HMM for gesture modeling. . . . .   | 69  |
| 4.12 | Segmentation correct ratio/false positive/false negative with different sub-image size and number of bins in the histogram. . . . .  | 71  |
| 5.1  | 4DT system with a standard flat-panel display. . . . .   | 75  |
| 5.2  | Calibrate pattern used in homography computation. . . . .  | 78  |
| 5.3  | (Left) An image pattern for color calibration. (Middle) and (Right) Images of the pattern on the display. . . . .  | 79  |
| 5.4  | An example of image segmentation based on color calibration. The upper left image is the the original pattern rendered on the screen. The upper right image is the geometrically and chromatically transformed image of the rendered pattern. The lower left image shows the image actually captured by the camera. The lower right image is the segmented foreground image. . | 80  |
| 5.5  | (Top)Pressing Gesture. (Bottom)Grasping Gesture. (Left) Original Image. (Right) Segmentation Image. . . . .  | 83  |
| 5.6  | Left picture shows the scene of when the finger is trying to trigger a button of $40 \times 40$ pixels. Right picture shows a user is controlling an X window program using finger press gestures. . . . .   | 84  |
| 5.7  | Button press experiment result on 4DT . . . . .  | 85  |
| 5.8  | Examples of the image pair and extracted appearance feature. Left and middle column display left images and right images of the scene, respectively. Right column shows the bottom layer of the feature volume (i.e., $F_{x,y,d}$ with $d = 0$ ). The z-axis of the feature figure indicates the value of each cell with corresponding x and y indices. . . . .                | 89  |
| 5.9  | An example of the motion vector. Left and middle column display the images of the scene at two different time. Right column shows the bottom layer of the motion volume. . . . .   | 90  |
| 5.10 | The average representation error against number of clusters. . . . .   | 92  |
| 5.11 | HMM structure for modeling dynamic gestures. . . . .   | 93  |
| 6.1  | Composite gesture example with a corresponding graphical model. . . . .  | 103 |
| 6.2  | Example HMM used to represent the gesture language model. . . . .  | 104 |
| 6.3  | Graphical depiction of two stages of the proposed greedy algorithm for computing the inference on the gesture model. Black nodes are not on the best path and are disregarded, and white represents past objects on the best path. Dark gray nodes represent current candidate gestures to be evaluated. . . .   | 108 |

|     |   |     |
|-----|---|-----|
| 6.4 | The gesture model we constructed for our experimental setup. The states are labeled as per the discussion in Section 6.3.1. Additionally, each node is labeled as either <b>P</b> arametric, <b>D</b> ynamic, or <b>S</b> tatic gestures. . . . . | 114 |
| 6.5 | The gesture model for our large-scale gesture experiment. . . . .   | 125 |
| 6.6 | Dual-stereo gesture system. . . . .   | 126 |

# List of Tables

|      |  |     |
|------|--|-----|
| 2.1  | Human sensory modalities in human-computer interaction . . . . .                                       | 17  |
| 2.2  | Example gesture systems. . . . .   | 31  |
| 4.1  | Experiment results with different length of characteristic sequence. . . . .                           | 72  |
| 5.1  | Training error of color calibration of each camera. . . . .  | 81  |
| 5.2  | On/Off neural network posture classification for the pressing and grasping gestures. . . . .           | 84  |
| 5.3  | Dynamic gesture recognition results for different feature spaces. . . . .                              | 96  |
| 6.1  | Example images of basic GWords. . . . .  | 113 |
| 6.2  | Parameters of the gesture model trained using supervised learning. . . . .                             | 117 |
| 6.3  | Parameters of the gesture model trained using unsupervised learning. . . . .                           | 117 |
| 6.4  | Recognition accuracy of the gesture model used in our experimentation. . . . .                         | 118 |
| 6.5  | Example images of our gesture vocabulary. . . . .  | 123 |
| 6.6  | Composite gestures and the corresponding gesture sequences. . . . .                                    | 124 |
| 6.7  | Posture recognition using neural networks for Higher Stereo. . . . .                                   | 129 |
| 6.8  | Posture recognition using neural networks for Lower Stereo. . . . .                                    | 130 |
| 6.9  | Posture recognition using histogram-based classifier for Higher Stereo. . . . .                        | 130 |
| 6.10 | Posture recognition using histogram-based classifier for Lower Stereo. . . . .                         | 130 |
| 6.11 | Training accuracy of dynamic gestures using standard HMMs and extended HMMs for Higher Stereo. . . . . | 131 |
| 6.12 | Testing accuracy of dynamic gestures using standard HMMs and extended HMMs for Higher Stereo. . . . .  | 132 |
| 6.13 | Training accuracy of dynamic gestures using standard HMMs and extended HMMs for Lower Stereo. . . . .  | 132 |
| 6.14 | Testing accuracy of dynamic gestures using standard HMMs and extended HMMs for Lower Stereo. . . . .   | 132 |
| 6.15 | Results of tracking parameterized gestures for Higher Stereo. . . . .                                  | 133 |
| 6.16 | Average amount of parametric changes of parameterized gestures for Higher Stereo. . . . .              | 133 |
| 6.17 | Results of tracking parameterized gestures for Lower Stereo. . . . .                                   | 134 |

|   |     |
|---|-----|
| 6.18 Average amount of parametric changes in parameterized gestures for Lower Stereo. . . . . | 134 |
| 6.19 Gesture recognition results using neural network processors. . . . .                     | 135 |
| 6.20 Gesture recognition results using the Bayesian classifier. . . . .                       | 136 |

# Acknowledgements

I would like to express my deep gratitude to Professor Gregory D. Hager, without whom the entire work would be impossible. His strong interests in scientific research, thorough understanding of the research fields, and deep insight into various problems that we met in the research are crucial for the success of our research projects. He is always available for providing all kinds of instructions, suggestions, and help. It has been a very pleasant and fruitful experience for me to work in the lab thanks to him.

I also would like to thank Dr. Jason Corso, Professor Darius Burschka, and Professor Allison Okamura for our collaboration on both VICs project and the VisHap project. It has been a pleasant experience to work with them, learn from them, and share the joy of success with them.

Dr. Izhak Shafran and Professor Darius Burschka read this dissertation and provided many good suggestions. I deeply thank them for their time and insightful recommendations. I also thank Professor Jason Eisner, Professor Ralph Etienne-cummings, Professor Darius Burschka and Professor Allison Okamura for their help on my Graduate Board Oral examination. Their suggestions are very important to my research projects on which this thesis is based.

Also, I would like to thank the rest of the Computational Interaction and Robotics Laboratory for all kinds of suggestions and help through the years. This long list of names includes Stephen Lee, Xiangtian Dai, Le Lu, Maneesh Dewan, Nicole Howie, Henry Lin, and Sharmishta Seshamani.

Deep thanks also go to the members of Haptics Explorative Lab, including Jake Abott and Panadda Marayong. Their help and recommendation on the VisHap

project is greatly appreciated.

Finally, I would like to thank my wife, Yuxia Qiu, for her strong support of my study and research. With her help, I can concentrate on my research and keep a positive attitude.

To Yuxia, and to my parents Yingkuan Ye and Changfang Li.

# Chapter 1

## Introduction

### 1.1 Background

The magic of Moore’s Law has continued for almost forty years. With the advances in computer and sensing technology, we are living in a world of virtually ubiquitous powerful computation and sensing. Computers of different dimension and nature are equipped with such devices as cameras, microphones, mechanical and magnetic sensors. In the near future, we probably will spend more time interacting with computers than with people.

Unfortunately, human-computer interaction(HCI) has been lagging behind the progress of computation. Currently, the dominant computer interface is the Graphical User Interface (GUI) [65] with WIMP(Windows, Icons, Menus, and Pointers) [34], which has little changes since its first appearance decades ago. This interaction model is mainly based on Shneiderman’s *Direct Manipulation* [117] paradigm. The main principles include continuous representation of objects of interest, physical actions on objects instead of complex syntax, incremental and reversible operations with an immediately-apparent effect on the objects of interest, and layered or spiral approach to learning.

Different interface models and techniques have emerged with advances in related technologies. Van Dam [34] proposed a general “post-WIMP” interface which contains interaction techniques not dependent on classical 2D widgets such as menus and icons. The ultimate goal is to involve multiple users and all senses in parallel, including vision,

speech, and haptics. Beaudouin-Lafon [6] proposed *instrumental interaction* in which GUI is described in terms of domain objects, which are objects of interaction, and interaction instruments, which are the computer artifacts with which domain objects are manipulated. The interaction instrument is essentially a mediator between the user and domain objects. It can be evaluated through three properties: degree of indirection, degree of integration and degree of compatibility.

Turk and Robertson [128] discussed the limitation of WIMP in today's pervasive computation environment and diverse interaction systems of disparate scales. They presented the *Perceptual Users Interface*(PUI), which is characterized by combining understanding of natural human capabilities (particularly communication, motor, cognitive and perceptual skills) with computer input/output devices, machine perception and reasoning. The main aim of PUI is to make user interfaces more natural and compelling by taking advantage of the ways in which people naturally interact with each other and with the world. Similar to post-WIMP paradigms, PUI also addresses the integration of multiple levels of technologies, including speech and sound recognition, computer vision, graphical animation and visualization, touch-based sensing and feedback (haptics), learning and user modeling. PUI integrates perceptive, multimodal and multimedia interfaces to carry human capabilities to create more natural and intuitive interfaces. *Multimodal interface* is a system that exploits multiple human sensory modalities for human-computer interaction; thus these modalities comprise integral and essential components of the interface language.

Almost all these new interaction models emphasize the importance of modeling and learning: after all, interaction is a way of communication between the human and the computer. The human needs to learn the interface to master it for efficient usage; the computer needs to learn the distinctive characteristics of interaction, both for human being as a whole and for a particular user.

Another important factor for natural and intuitive interaction is computer vision which essentially enables the computer to see and understand the environment. Computer vision is very promising for intelligent user interfaces. It is a passive input that can be integrated into many conventional environments ranging from offices to large-scale outdoor facilities. Even a common desktop computer can be equipped with inexpensive cameras to

build an adaptable interaction environment. Furthermore, rich visual information provides strong cues to infer the motion and configuration of the human body, thus enabling the computer to understand the underlying interaction messages. All these factors serve as the motivation to use vision as a tool to facilitate intuitive and intelligent human-computer interaction.

Although vision holds great promise for general human-computer interaction, it also brings huge challenges. First, imaging is essentially a noisy physical process. Second, human appearance and motion are very complex, articulated and difficult to model [23, 96, 145]. There are also tremendous variations across different persons. Third, it is a complicated task to learn the interaction language of an interface. Even for current interfaces (e.g., mice and joysticks), which put the burden of learning almost exclusively on the human, the learning curve for average users is still steep. For the computer to understand a natural and intuitive interaction and to adapt to different users, it mandates a precise yet flexible model, as well as efficient learning techniques. In this dissertation, we study these problems in the context of vision-based human computer interaction.

## 1.2 Motivation

We are interested in the general problem of human-computer interaction and particularly, how computer vision can be applied to a natural and intuitive perceptual interface. In this dissertation, we are motivated by two specific aspects of this broad problem. First, we want to apply vision techniques to a haptics-based interface to enhance the immersiveness and convenience of conventional haptic system. Second, we aim to incorporate vision to replace traditional cumbersome mediation via such devices as mice and keyboards with a natural and intuitive communication. Concretely, we propose a perceptual interface where users carry out hand gestures to manipulate interaction components and communicate with the computer.

In the remainder of this section, we will discuss our motivations in detail. First we will discuss incorporating vision into a multimodal interface. Then we concentrate on systems that mainly rely on vision for intuitive and natural interaction.

### **1.2.1 Vision In Multimodal Interfaces**

The PUI paradigm is closely related to perceptive user interfaces and multimodal user interfaces. In perceptive user interfaces, we integrate human-like perceptual capabilities into the computer, for example, making the computer aware of the user's speech, facial expression, body and hand motion. Vision provides rich visual streams to the computer to augment its perceptiveness. In multimodal user interfaces, various modalities, such as speech, vision and haptics, are combined to lead to more effective and more natural interaction. For example, in an augmented reality system for playing table tennis, it would be ideal that the user can issue controlling commands to the system using either speech or gesture, feel force feedback when he or she hits the ball using the bat, as well as moves his or her body and hand at will. In such an environment, computer vision plays roles in both communication (e.g., understanding gesture commands) and manipulation, for example, capture the user's striking motion and the trajectory of arm/hand movement for training purposes.

Although there has been some research on multimodal interfaces [91, 114], many open questions remain unanswered and until today, there have been very few systems that integrate multiple modalities seamlessly and effectively. In this dissertation, we try to address one specific aspect of integrating computer vision into haptics to achieve more natural interaction.

### **1.2.2 Vision-Based Interaction**

Recent development in Virtual Environments (VE) and other related technologies has taken us to three-dimensional (3D) virtual worlds and prompted us to develop new HCI techniques. Many of current VE applications employ such traditional HCI media as keyboards, mice, joysticks and wands. However, these tools and techniques are limited in dealing with 3D VE. For example, keyboards and mice are primarily focused on dealing with two-dimensional (2D) interfaces. Magnetic devices can also be used to measure the motion of the human body and act as important data collection tools in some VE systems. But they are prone to magnetic interference and are also cumbersome.

Vision-based interaction (VBI) is an excellent choice to replace cumbersome tools. The basic idea of VBI is to use a single or multiple cameras to observe the scene, and to parse the video streams to understand the user's gestures, expression or any other interaction messages conveyed in the visual channel.

Incorporating VBI into an interface allows new channels of interaction techniques. For example, computer vision permits intuitive interaction with freehand motion. A good example is to play a virtual jigsaw puzzle game on the computer. Using WIMP paradigm, the user has to learn a cumbersome rules of interaction, such as clicking a piece once a time, dragging it into a new position, releasing it, clicking another icon to switch to rotate mode, rotating the piece by dragging one of its corners and then releasing it. Using VBI, the user can move and rotate the pieces just as he or she is playing a real game on a desk. Furthermore, the user can use both hands in parallel to manipulate multiple pieces simultaneously.

Apparently, VBI is closely related to the models of post-WIMP and perceptual interaction. It is a very challenging problem that involves such disciplines as computer vision, graphics, imaging processing, linguistics and psychology. Although there have been enormous advances in this field, currently VBI is still in its infant stage. Ultimately, we aim to build a system that meets the following criteria:

1. **Robustness**

Many factors can affect the visual appearance of the interaction scene, such as imaging noise, illumination changes, occlusion, and dynamic backgrounds. Furthermore, different users have distinct appearances and their own dynamic character in carrying out the interaction actions. Thus, the interaction system must handle these variances robustly.

2. **Extensibility and Scalability**

The users should be able to modify the system's action vocabulary with a moderate effort to meet the requirements of different applications. The core HCI modules can also be easily incorporated into systems of different scales, such as a desktop system or large-scale VEs.

### 3. Efficiency of the Computation

Real-world HCI systems mandate real-time processing; otherwise the users can feel the distinct discrepancy between real-world and virtual reality, therefore harming the fidelity and immersiveness of the interaction experience. On the other hand, the data volume of the video stream is huge. Therefore, the VBI system must be efficient in computation to meet the requirement of real-time applications.

#### 1.2.3 Gestures for VBI

Although other kinds of human interaction media have been investigated [5, 11, 18, 47] such as facial expression and gaze, hand gestures are most popular for VBI. One reason for gesture's popularity is that gestures have been one of the most important human-human communicative means besides spoken language. For centuries, sign languages have been used extensively among speech disabled people. People that can talk and listen also carry out many kinds of gestures in daily communication. Because of its importance, gestures and sign languages have been studied for centuries. According to Kendon [70], gestures are organized separately in the brain and are complementary to speech serving in cooperation with it in the elaboration of a total utterance.

Hand gestures are also intuitive and sophisticated, possessing rich communication power. Human hands are deformable objects and their motion is highly articulate, with roughly 27 degrees of freedom (DOF). At the same time, human beings have been using hands extensively and can manipulate the hands with ease to carry out many complex tasks, such as pointing, grasping, rotating and other kinds of manipulation and communication.

Furthermore, for advanced interfaces in which the user manipulates 3D objects, gestures are more appropriate and powerful than traditional interaction media, such as a mouse or a joystick. For example, the user may try to grasp a virtual object in 3D space and turn it around to have a good look at it. In such cases, it would be ideal that the user just follow his or her intuition and daily customs to use hand gestures to accomplish such a task.

To implement gesture-based HCI, we need capture the necessary information to infer what gesture is performed by the user. Special data gloves have been used to collect

such information as spatial position and joint angles as input to HCI systems. However, users may find these kind of sensors awkward and inconvenient to wear.

Fortunately, video streams can also present many cues to carry out the inference of the user's gestures. Furthermore, vision-based systems are non-invasive and can also be made computationally efficient. Because of these reasons, vision-based gesture interfaces have been investigated and applied in many HCI systems [28, 35, 36, 54, 78, 92, 99, 102, 128, 141, 152, 162]. To successfully and efficiently use cues to implement a gesture-based interfaces, there are several key problems to solve:

### 1. **Gesture Modeling**

Although the gestures for HCI is only a subset of the whole spectrum of gestures, there is still no satisfactory gesture model to date, mainly because of the complexity of gestures. Gestures can involve articulated arm and hand motion, as well as complex temporal dynamics. As a language, its syntax and semantics has not been well modeled and applied to VBI. In general, modeling of gestures involves representation of hand shape and kinematics, hand motion, temporal and semantic constraints.

### 2. **Gesture Analysis**

The visual analysis of gesture involves computation of the parameters of the hand model based on selected visual features. Although visual streams provide rich information, the extraction and selection of visual features is not a trivial problem. To achieve efficient and robust representation of the gestures, effective algorithms for features selection and parameter estimation are mandatory.

### 3. **Gesture Recognition**

Given a gesture model and parameters computed from visual streams, recognition usually involves finding the gesture that best matches the parameters in the parameter space. Based on the differences in dynamic properties, gestures can be classified as static gestures (also called hand postures), and temporal (or dynamic) gestures. Dynamic gestures involves complicated spatio-temporal characteristics and have a more complex parameter space than postures.

## 1.3 Thesis Statement

Vision can be integrated into multimodal interfaces to enhance the fidelity and efficacy; vision-based techniques also make intelligent human-computer interaction possible without globally modeling the user.

## 1.4 Overview

In this dissertation, we present novel techniques for the integration of computer vision into a haptics-based multimodal system, application of vision into a modular HCI framework, efficient capture of hand motion, and integration of multimodal gestures into a coherent probabilistic framework. We also present the development of a real-time gesture-based interface system and discuss a human factor experiments carried out in this system. The dissertation is organized as follows:

- Chapter 2 presents a comprehensive overview of the state-of-the-art of research in the field of multimodal interfaces and gesture-based human-computer interaction. The research and systems on multimodal interface is discussed in Section 2.1. Section 2.2 presents related works in interface design for vision-based interaction. We discuss topics related to the gesture-based interfaces, such as gesture definition and modeling, visual cue extraction and selection, and gesture recognition in Section 2.3.
- Chapter 3 focuses on our design and implementation of a multimodal perceptual interface where vision is integrated into a haptics-based system to increase the immersiveness and fidelity of the interface. This framework eliminates the limitation of constant contact of the user's hand and the haptic device. In Section 3.3 we present some experimental results to demonstrate the framework's effectiveness.
- Chapter 4 presents our novel HCI framework which is based on the claim that global user tracking and modeling is generally unnecessary. This new framework is called the Visual Interaction Cues Paradigm or VICs. VICs fundamentally relies on a shared perceptual space between the user and computer using monocular and stereoscopic

video. By representing each interface component as a localized region in the image(s), it is not necessary to visually track the user. Instead, we model gestures based on stream of visual cues extracted from localized image(s). In Section 4.1 we discuss the basic principles of VICs, models of interaction, and system implementation. Section 4.2 presents a stochastic VICs processor that recognizes 2D gestures using Hidden Markov Models.

- Chapter 5 describes a novel methodology to capture hand motion for gesture recognition on a VICs platform. In Section 5.1, we present the design and implementation of a real-time VICs system: the 4DT platform. Through such techniques as component mapping, chromatic calibration and color modeling, it allows efficient capture of hand gestures. In Section 5.2, we propose a novel approach to extract 3D shape and motion features based on efficient stereo matching from localized image regions. These features are used to analyze both postures and dynamic gestures. Section 5.3 presents two methods (Hidden Markov Models and neural networks) to model dynamic gestures using these 3D features.
- Chapter 6 presents a probabilistic framework to incorporate multimodal gestures (e.g., static postures, dynamic gestures, and parameterized gestures). In this framework, low-level gestures are represented as gesture words and a complex gesture sequence is modeled as a sequence of words. The context constraints among gesture words are modeled using a HMM model. Supervised and unsupervised learning techniques are discussed to train the model. A specific greedy algorithm is also introduced for online inference of continuous gestures for real-time HCI. We present the basic model in Section 6.2 and a gesture recognition experiment based on this model in Section 6.3. A large-scale human factors experiment is discussed in Section 6.4. The results are presented in Section 6.5 to demonstrate the robustness and efficacy of our system.
- Chapter 7 summarizes the dissertation and presents discussions and future research work.

## 1.5 Dissertation Contributions

In this section, we explain the major contributions of this dissertation.

### 1.5.1 Contribution 1: Effective Integration of Vision and Haptics

As mentioned earlier, to build a multimodal interface requires effective and seamless integration of different sensing techniques. We design and implement a perceptual interface that combines vision and haptics [154]. The motivation is that traditional haptic environments require the user to be attached to the haptic device at all times, even though force feedback is not always being rendered. Our system eliminates this limitation through real-time visual tracking, thus seamlessly integrating force feedback with tactile feedback to generate a “complete” haptic experience. The VisHap framework allows the user to interact with combinations of virtual and real objects naturally, thereby combining active and passive haptics. The flexibility and extensibility of our framework is promising in that it supports many interaction modes and allows further integration with other augmented reality systems.

### 1.5.2 Contribution 2: Novel Methodology for Applying Computer Vision to Human-Computer Interaction

We take a general approach to incorporating vision into the human-computer interaction problem that is applicable for both current 2D interfaces and future 3D interfaces. A survey of the literature (Section 2.2) shows that most reported work on VBI relies heavily on visual tracking and visual template recognition algorithms as its core technology. While tracking and recognition are, in some sense, the most popular direction for developing advanced vision-based interfaces, one might ask if they are either necessary or sufficient. For example, when a user presses the buttons on a telephone to make a phone call, neither the world or the phone maintains the notion of the user. The only thing that matters to the phone is that something presses the key and releases it. In contrast, conventional approaches for VBI attempt to build a model of the user’s finger, track it through space even though it is far away from the phone, and perform some activity recognition when the user

presses the keys.

Instead, we present the VICs paradigm which does not employ globally modeling and tracking of the user [22, 29, 155, 150, 152]. This paradigm allows flexible and efficient gesture modeling and recognition by limiting the computation to a localized space. Based on this general framework, we have designed and implemented a real-time interactive platform. On this platform the user can use natural gestures to perform common tasks on the interface like pressing buttons and scrolling windows. Different gesture modeling techniques have been studied on this platform, including various posture modeling approaches and statistical models for dynamic gestures. Several real-time VBI applications are also developed to demonstrate the efficacy of the framework.

### **1.5.3 Contribution 3: Efficient Motion Capture for Gesture Recognition**

To apply VBI to small-scale and inexpensive computation environment (e.g., desktop computer), efficient computation is a necessity. We propose a novel approach to capture hand motion in the neighborhood of the target interaction component without tracking the user [153, 151]. Instead, we take an object-centered approach that efficiently computes the 3D appearance using a region-based coarse stereo matching algorithm in a volume around the hand. In this way, high-dimensional images are converted into a discrete vector whose dimensionality can also be scalable according to the size of the local space and resolution of the stereo matching. The motion cue is captured via differentiating the appearance feature.

In this dissertation, we implement several pattern recognition schemes to recognize postures using this localized feature. These schemes include unsupervised learning techniques (e.g., vector quantization), histogram-based Bayesian classification, and neural networks. We also investigate methods to classify dynamic gestures in this feature space. Such techniques as neural networks and Hidden Markov Models are used to demonstrate the representative power and efficiency of this motion capture scheme.

### **1.5.4 Contribution 4: Unified Gesture Modeling for Multimodal Gestures**

Communicative and controlling gestures can be classified into three types:

1. **Static postures** [2, 78, 90, 99, 125, 144, 163] that model the gesture as a single key frame, thus discarding any dynamic characteristics. For example, in recent research on American Sign Language (ASL) [163], static hand configuration is the only cue used to recognize a subset of the ASL consisting of alphabetical letters and numerical digits. This approach is efficient in capturing gestures that display explicit static spatial configuration.
2. **Dynamic gestures** contain both spatial and temporal characteristics, thus providing more challenges for modeling. Many models have been proposed to characterize the temporal structure of dynamic gestures: including temporal template matching [13, 82, 95, 116], rule-based and state-based approaches [15, 99], Hidden Markov Models (HMM) [92, 100, 122, 149, 152] and its variations [19, 92, 139], and Bayesian networks [115]. These models combine spatial and temporal cues to infer gestures that span a stochastic trajectory in a high-dimensional spatio-temporal space.
3. **Parameterized gestures** carry quantitative information like angle of a pointing finger or speed of waving arm. Most current systems model dynamic gestures qualitatively. That is, they represent the identity of the gesture, but they do not incorporate any quantitative, parametric information about the geometry or dynamics of the motion involved. However, to cover all possible manipulative actions in the interaction language, we include parametric gestures. One example of this type of gesture modeling is the parametric HMM (PHMM) [139]. The PHMM includes a global parameter that carries an extra quantitative representation of each gesture.

These individual gestures are analogous to a language with only words and no grammar. To enable natural and intuitive communication between the human and the computer, we have proposed a high-level language model that integrates the individual gestures into a single, coherent probabilistic framework [30]. In this framework, each low-level gesture is called a gesture word, or GWord. A composite gesture is a sequence of GWords that are constrained contextually. We propose efficient learning techniques and inference algorithm to allow online parsing of continuous gestures. To the best of our knowledge, this is the first model to include the three different low-level gesture types into

a unified model.

Furthermore, we carry out a human factors experiment on our 4DT platform, which essentially is an implementation of the VICs paradigm. Our review (See Section 2.3) on vision-based gesture interface reveals that most current research deal with a limited gesture vocabulary and a very small group of subjects. In our experiment, sixteen subjects take part in the experiment. Our gesture vocabulary includes fourteen low-level gestures and nine possible high-level gesture sentences. To our knowledge, our experiment of visually modeling multimodal gestures is the first that consists of a large-scale dataset with more than fifteen users. The experimental results justify the robustness and efficacy of our VICs paradigm, hand motion capture and high-level gesture modeling.

## 1.6 Notation

We denote the Euclidean space of dimension  $n$  by  $\mathbb{R}^n$  and the projective space of dimension  $n$  by  $\mathbb{P}^n$ . We define an image as any scalar or vector field:  $\mathbf{I} \doteq \{\mathcal{I}, I, t\}$ , which is a finite set of points  $\mathcal{I}$  in  $\mathbb{R}^2$ ) together with a map  $I : \mathcal{I} \rightarrow \mathcal{X}$ , where  $\mathcal{X}$  is some arbitrary value space, and  $t$  is a time. We extend this notation for an image sequence as  $\mathcal{S} = \{\mathbf{I}_1 \dots \mathbf{I}_m\}$  with length  $m \geq 1$ . While the distinction should be clear from the context, we make it explicit whenever there is ambiguity.

## 1.7 Relevant Publications

This dissertation is based on the following publications:

1. Guangqi Ye, and Gregory Hager, Appearance-based Visual Interaction. *CIR Lab technique report*, Department of Computer Science, Johns Hopkins University, 2002.
2. Jason Corso, Guangqi Ye, Darius Burschka, and Gregory Hager, Software Systems for Vision-Based Spatial Interaction. *Proceedings of 2002 Workshop on Intelligent Human Augmentation and Virtual Environments*. Chapel Hill, North Carolina, 2002. Pages D-26 and D-56.

3. Guangqi Ye, Jason J. Corso, Darius Burschka, Gregory D. Hager, VICs: A Modular Vision-Based HCI Framework. *Proceedings of 3rd International Conference on Computer Vision Systems (ICVS 2003)*, pages 257-267.
4. Guangqi Ye, Jason J. Corso, Gregory D. Hager, Allison M. Okamura, VisHap: Augmented Reality Combining Haptics and Vision. *2003 IEEE International Conference on Systems, Man & Cybernetics (SMCC'03)*.
5. Guangqi Ye, Jason J. Corso, Darius Burschka, and Gregory D. Hager. Vics: A Modular HCI Framework Using Spatio-temporal Dynamics. *Machine Vision and Applications*, 16(1), pages 13-20, 2004.
6. Guangqi Ye, Jason J. Corso, and Gregory D. Hager. Gesture Recognition Using 3D Appearance and Motion Features. *Proceedings of CVPR 2004 Workshop on Real-Time Vision for Human-Computer Interaction*, 2004.
7. Jason J. Corso, Guangqi Ye, and Gregory D. Hager. Analysis of Multi-Modal Gestures with a Coherent Probabilistic Graphical Model. *Virtual Reality*, 2005.
8. Guangqi Ye, Jason J. Corso and Gregory D. Hager, Visual Modeling of Dynamic Gestures Using 3D Appearance and Motion Features. in *Real-Time Vision for Human-Computer Interaction*, edited by B. Kisacanin, V. Pavlovic and T.S. Huang, Springer-Verlag, 2005.
9. Darius Burschka, Guangqi Ye, Jason J. Corso, and Gregory D. Hager. A Practical Approach for Integrating Vision-Based Methods Into Interactive 2d/3d Applications. *CIRL Lab Technical Report CIRL-TR-05-01*. The Johns Hopkins University, 2005.

## Chapter 2

# Vision for Natural Interaction: A Review

In this chapter, we review the state-of-the-art research in applying vision to natural and intuitive perceptual interface. We shortly summarizes work that integrates vision into multimodal interfaces in Section 2.1. The progress of vision-based HCI in the broad field of virtual environments is discussed in Section 2.2. Section 2.3 reviews the modeling and representation of gestures in the HCI context. Different methods to detect and track the hand are discussed in Section 2.3.4. Section 2.3.5 presents techniques to extract visual cues from video streams. In Section 2.3.6 we review techniques to model and recognize postures, temporal gestures and high-level gestures.

### 2.1 Vision and Multimodal Interface

In this section, we first review work in general multimodal interfaces. Then we discuss recent research on integrating vision into multimodal systems.

#### 2.1.1 Mutltimodal Interface

The conventional interaction model, i.e., GIMP [65], mainly relies on such cumbersome mechanic devices as mice and keyboards as a mediation between the user and

the computer. It roughly follows Shneiderman's principles of direct manipulation [117]. Advances in computation, sensing, and application demand more effective, natural, and intuitive interaction language. Van Dam's post-WIMP [34] model contains interaction techniques not dependent on classical 2D widgets such as menus and icons. The ultimate goal is to involve multiple users and all senses in parallel, including vision, speech, and haptics. Turk and Robertson [128] discussed the limitation of WIMP and proposed the perceptual user interface or PUI, which is characterized by combining understanding of natural human capabilities with computer input/output devices, machine perception and reasoning. The main aim of PUI is to make user interfaces more natural and compelling by taking advantage of the ways in which people naturally interact with each other and with the world. PUI integrates perceptive, multimodal and multimedia interfaces to carry human capabilities upon creating more natural and intuitive interfaces.

All these new paradigms address integrating multiple levels of technologies, including speech and sound recognition, computer vision, graphical animation and visualization, touch-based sensing and feedback (haptics). Multimodal interaction aims to achieve such a goal. It exploits multiple human sensory modalities for human-computer interaction; thus these modalities comprise integral and essential components of the interface language.

The research on designing integrated multimodal system is still in its infant phase although it has already been addressed by Bolt [16] years ago. Blattner and Glinert [12] summarized some human sensory modalities closely related to HCI in the literature, as shown in Table 2.1. They also discussed several integration mechanisms, such as frames which is essentially a net of nodes, neural nets, and agents that build a model of the user to help customize the interaction with the user.

To improve reliability and usability, there have been many efforts to design and build new multimodal systems [73, 130]. Johnston and Bangalore [66, 67] proposed finite-state methods to understand multimodal languages consisting of speech and gestures. Speech and gesture inputs are assigned a combined meaning by a finite-state device which essentially specifies a multimodal context-free grammar. The proposed scheme is used in a city information application in which users issue spoken commands while gesturing at icons on a dynamically generated display.

| <b>Modality</b>     | <b>Examples</b>  |
|---------------------|--|
| Visual              | Gaze<br>Image capture  |
| Auditory            | Voice (as part of language faculty )<br>Nonspeech audio (as part of the music faculty) |
| Haptic/Kinesthetic  |  |
| Touch               | Pressure<br>Texture  |
| Hand Movement       | Sign language<br>3D motions<br>Writing motions<br>Drawing or other nontextual gestures |
| Head Movement       | Lip reading, oral cavity<br>Head movement for pointing<br>Facial expressions           |
| Other Body Movement | Movement captured in dance or sports   |

Table 2.1: Human sensory modalities in human-computer interaction

Obrenovic and Starcevic [91] designed a generic multimodal HCI framework using well-known and widely-supported Unified Modeling Language(UML). A meta-model is defined to represent an abstract, higher-level view of various aspects of multimodal interaction. Standard UML is extended to model basic modalities and describe complex multimodal systems. Simple modalities include input modality which can be event-based or streaming-based, and output modality which can be static or dynamic. A complex modality is the integration of multiple simple modalities. This scheme provides a practical software framework for multimodal interface.

### 2.1.2 Integrating Vision into Multimodal Interface

Because of its importance and advantages, vision has been one of the most used modalities in multimodal interfaces. Most reported works focus on integrating vision and speech. Bolt [16] presented an interface where voice commands are augmented by simultaneous pointing by hand, thus making free usage of pronouns possible when commanding components on a large-screen graphical screen. Sharma, et al. [114] described a bimodal speech/gesture interface in a 3D visual-computing environment used by structural biolo-

gists. The interface uses both automatic speech recognition and gesture recognition in a command language. The basic syntax is <action> <object> <modifier>, where action is spoken (e.g., “rotate”), while a combination of speech and gesture specifies object and modifier. One example is to speak “this” while pointing and followed by a modifier to specify what is really pointed to. The interface is tested and evaluated by several researchers. Involved users reported that the speech/gesture interface is more convenient and more efficient than a traditional graphical interface.

Pai [94] introduced a multi-sensory interaction system called HAVEN (Haptics, Auditory and Visual Environment). This complex system allows users to interact with other users, physical objects, and computer simulations. As users interact in the HAVEN, their motion, applied forces, and sounds are simultaneously measured using a multitude of sensors located on the walls, on the ceiling, on the table top, and also attached to manipulated objects and probes, and even attached to the user’s hand. Examples of sensors include a motion capture system, a microphone array, a pressure sensor pad, a hand-held haptic texture sensor, haptic devices and cameras. A cluster of computers connected with a gigabit network process this huge amount of information in real-time. Although HAVEN contains sensors for vision, haptics, and sound, it is unclear how these multiple modalities would be efficiently and seamlessly integrated to enhance the naturalness and fidelity of the interface.

### 2.1.3 Vision and Post-WIMP Interfaces

Many modern non-WIMP 2D interfaces have been proposed over the years. Most are augmented desk style interfaces, such as Wellner’s DigitalDesk [138], which attempts to fuse a real desktop environment with a computer workstation by projecting the digital display on top of the desk. The EnhancedDesk [72, 92] provides an infrastructure for applications developed in an augmented desktop environment. Both color cameras and infrared cameras are installed above the desktop to provide visual cues for gesture recognition.

Another advance in interface design is the 3D interface. Its spectrum ranges from those on a 2D monitor to fully immersive 3D interfaces. The simplest 3D interfaces are those projected onto a conventional 2D screen [42], and have been used extensively in the gaming

industry and for scientific visualization. More complex 3D interfaces include single-source stereoscopic style displays [110] and holographic style displays [10] Recently, more immersive 3D display systems have been proposed and applied in virtual environments, including spatially immersive displays [32] and Head-Mounted Displays (HMD) [123]. Azuma [3, 4] summarized different types of HMDs and their applications in augmented reality. Basic types include closed and half-silvered see-through HMDs. A closed HMD enables both virtual reality and video see-through augmented reality. In the video see-through augmented reality case there are cameras mounted on the closed HMD. The images rendered to the screens in the HMD are the composition of the real imagery being captured by the cameras and synthetic imagery generated by the computer. By contrast, in optical see-through, the user can see the real world through the half-silvered HMD while the computer is also rendering synthetic objects into view.

## 2.2 Vision-Based HCI

Recent advances in virtual environments and augmented reality techniques [3] have presented us with the challenge of designing and implementing new HCI interfaces that are more natural and intuitive. Following this trend, spoken language [64], haptics [62, 109, 154, 157, 159] and vision [23, 24, 96, 128, 145] have been popular choices to replace traditional interaction media, such as keyboards and mice.

In the field of computer vision, there has been a great deal of interest in recognizing human body motion, faces and facial expression, with the general goal of supporting human-computer interaction [5, 11, 18, 21, 46, 47, 92, 96, 146, 148]. Many “demonstration systems” [81, 84, 103, 111, 112, 113, 120, 121] have been implemented and show the bright promise of vision-based interaction.

In the rest of this section, we will discuss work based on full body motion and head/face motion. We postpone discussion about hand gestures to Section2.3.

### 2.2.1 Full Body Motion

The Pfinder system [141] and its related applications is a commonly cited example of a vision-based interface. Pfinder uses a statistically-based segmentation technique to detect and track a human user as a set of connected blobs. Various filtering and estimation algorithms extract information from these blobs to produce a running state estimate of body configuration and motion [97]. Most systems make use of body motion estimates to animate a character or allow a user to interact with virtual objects.

Gavrila and Davis [47] use a skeleton-based model of the 3D human body pose with seventeen degrees-of-freedom and a variation of dynamic time warping [88] for the recognition of movement. Bregler, et al. [20, 21] present a visual motion estimation technique to recover high degree-of-freedom articulated human body configurations in complex video sequences. The product of exponential maps and twist motions are integrated into a differential motion estimation method, resulting in solving simple linear systems to recover the kinematic degrees-of-freedom in noisy and complex self-occluded configurations.

### 2.2.2 Head and Face Motion

Basu and Pentland [5] proposed an algorithm for robust, full 3D tracking of the head using model-regularized optical flow estimation. This method uses a 3D ellipsoidal model of the head and interprets the optical flow in terms of the possible rigid motion of the model.

Black and Yacoob [11] presented a technique for recognizing facial expressions based on the combination of global rigid motion information with local non-rigid features. Local features are tracked with parametric motion models. The model gives a recognition accuracy of over 90% on a dataset of 40 subjects.

Bradski [18] developed an extension of the mean-shift algorithm for face tracking. The algorithm continuously adapts to the dynamically changing color probability distributions. The tracking algorithm is applied to explorative tasks for computer interfaces.

## 2.3 Visual Modeling of Gestures for HCI

In this section, we will discuss different techniques for gesture modeling, analysis and recognition.

### 2.3.1 Definition and Taxonomy of Gestures

Because of their importance and popularity in human-human communication, gestures [70, 89, 141] have been a popular choice for vision-based interfaces. However, there exist some differences in the definition of gestures in general and that used in the HCI paradigm. For example, the Merriam-Webster Dictionary defines “gesture” as “a movement usually of the body or limbs that expresses or emphasizes an idea, sentiment, or attitude”, or “the use of motions of the limbs or body as a means of expression”. This definition, which emphasizes the intention of “expression”, hardly fits the concept in the HCI context. In many HCI systems, the user uses hand gestures to both manipulate objects and to communicate with some interaction elements, such as icons in a graphical user interface (GUI). This use of hand gestures is categorized as practical gestures [70] that are beyond the traditional notion.

In the literature, several taxonomies are proposed to deal with the psychological definition of gestures. Kendon [70] distinguishes “autonomous gestures” from “gesticulation” which occur in association with speech for the aim of a total articulation. Quek [99] developed a taxonomy that better fits the HCI context. In this taxonomy, a distinction is made between manipulative gestures and communicative gestures, the later of which are particularly intended to convey messages of communication. Figure 2.1 shows the gesture taxonomy proposed in [96].

Since both spoken language and gestures are means of communication, people tend to model the gesture in a similar fashion as spoken language [96]. According to this model, gestures originate as a gesturer’s mental concept, then are expressed via the hand/arm movement, and finally are perceived by the receiver as visual images. Equation 2.3.1 formulates such a process.

$$V = T_{vh}H = T_{vh}T_{hg}G = T_{vg}G \quad (2.3.1)$$

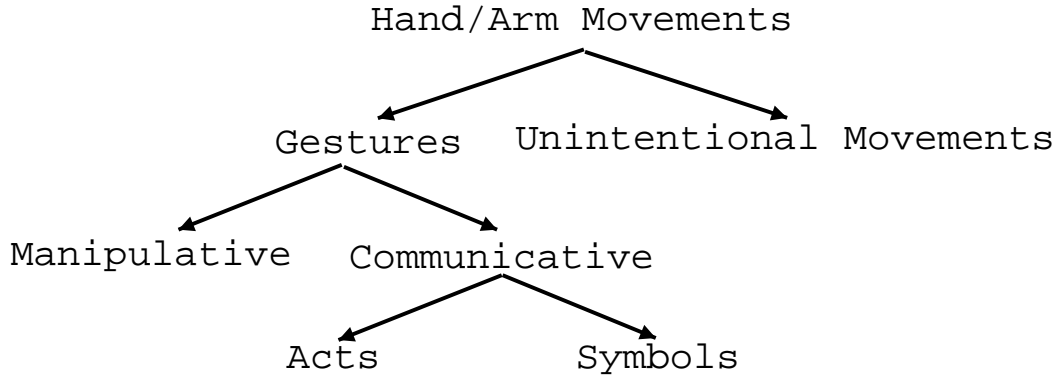


Figure 2.1: Taxonomy of hand gestures.

In this equation,  $V$  refers to the visual perception of the gesture,  $H$  is the hand/arm representation of the gesture  $G$ .  $T_{hg}$  is the transformation that models the hand/arm movement given the gesture  $G$ , and  $T_{vh}$  models the visual process of the hand movement  $H$ . Put together,  $T_{vg}$  is the transformation between the gesture  $G$  and visual perception  $V$ .  $T_{vg}$  is a parametric model with the parameter space  $M_T$ . Thus the visual interpretation of gestures is to infer  $G$  from the visual images  $V$ , which is modeled by  $T_{vg}$ .

$$\hat{G} = T_{vg}^{-1}V \quad (2.3.2)$$

Following this formulation [96], gestures are defined as:

*A hand gesture is a stochastic process in the gesture model parameter space over a suitably defined time interval.*

According to this definition, each realization of a gesture can be viewed as a trajectory in the model parameter space. Furthermore, the stochastic nature means that no two instances of the same gesture will generate the same hand/arm motion or the same set of visual images.

### Static and Dynamic Gestures

According to the nature of the gestures in the vocabulary, the gestures in existing interfaces can be classified into two categories: static hand postures and dynamic gestures.

Static postures [2, 78, 90, 99, 125, 144] model the gesture as a single key frame, thus discarding any dynamic characteristics. For example, in recent research on American Sign Language (ASL) [163], static hand configuration is the only cue used to recognize a subset of the ASL consisting of alphabetical letters and numerical digits. The advantage of this approach is the efficiency of recognizing those gestures that display explicit static spatial configuration. However, it has an inherent shortcoming in handling dynamic gestures whose temporal patterns play a more important role than their static spatial arrangement.

Dynamic gestures contain both spatial and temporal characteristics, thus providing more challenges for modeling. Many models have been proposed to characterize the temporal structure of dynamic gestures: including temporal template matching [13, 82, 95, 116], rule-based and state-based approaches [15, 99], Hidden Markov Models (HMM) [92, 100, 122, 149, 152] and its variations [19, 92, 139], and Bayesian networks [115]. These models combine spatial and temporal cues to infer gestures that span a stochastic trajectory in a high-dimensional spatio-temporal space.

Most current systems model dynamic gestures qualitatively. That is, they represent the identity of the gesture, but they do not incorporate any quantitative, parametric information about the geometry or dynamics of the motion involved. To overcome this limitation, a parametric HMM (PHMM) [139] has been proposed. The PHMM includes a global parameter that carries an extra quantitative representation of each gesture. This parameter is included as an additional variable in the output probabilities of each state of the traditional HMM.

### 2.3.2 Modeling Temporal Structure

Most researchers agree on the temporal nature of dynamic gestures, i.e., gestures are performed over a characteristic duration of time. As mentioned in [70, 96, 99], dynamic gestures are considered as composed of three temporal phases, i.e., (1) preparation, (2) nucleus, also called peak or stroke and (3) retraction.

Generally speaking, the nucleus of the gesture has definite form and enhanced dynamic qualities that we can make use of as the key clues to identify the gesture. For example, both the preparation and retraction stage of the gestures are often characterized

with rapid movements of the hands, while the stroke involves slower motion. These differences can help us to recognize temporal gestures. One extreme example is the static gesture recognition [2, 78, 90, 99, 125, 144], which models the gesture as a single key frame, thus discarding dynamic characteristics.

### 2.3.3 Modeling the Spatial Structure

The gesture is performed in 3D space and demonstrates its own spatial characteristics. One way to model this spatial pattern is to follow Equation 2.3.1 and infer the gesture directly from visual cues. This method is often called the appearance-based approach. The other method tries to infer the parameters of the predefined hand/arm model first, then determine the gesture based on the hand/arm configuration. In the literature, this is called the model-based approach.

3D hand/arm modeling can be classified into two categories: volumetric model and skeletal model. Volumetric modeling is to describe the 3D visual appearance of the hand/arm. This approach belongs to the general frame of *analysis-by-synthesis* [125, 136, 141]. Basically, a 3D model is used to describe the visual appearance, and a set of parameters of the model are varied until the model matches the visual image. Example models include 3D surfaces, geometric structure such as generalized cylinder and super-quadrics, etc [162]. On the other hand, skeletal model uses a set of joint angles and segment length of the parts of the hand/arm as the model parameters [95]. A commonly-used kinematics model of the hand can be found in [96, 145]. A simpler kinematic model is used by Rehg and Kanade [106] to predict possible occlusion and to track the hand.

Generally speaking, a 3D model-based approach offers a rich description and discrimination capabilities that potentially allow a wide class of gestures. However, these models are often of quite high dimension and the computation of the 3D model parameters from real-time video is computationally extensive.

Instead of using an explicit 3D or 2D model, appearance-based modeling of the hand tries to extract image features and statistics to model the visual appearance of the hand directly [2, 13, 15, 35, 39, 78, 82, 85, 95, 102, 116, 133, 142, 143, 144, 151, 152]. One popular choice is 2D templates, such as 2D blobs and template images [39, 149, 162]. An

extension to the template-based approach is to use the whole sequence as a template. For example, a history image can be constructed to indicate the change of the appearance over time [13, 82]. Visual features and statistics extracted from the image also play an important role in hand modeling. These cues include contours, edges and corners, the positions of the fingertips, moments and other mathematical statistics [15, 39, 54, 92, 152, 151].

Appearance-based approach provides a simpler model to implement real-time gesture recognition. However, it suffers from inherent limitations such as ambiguity due to projection, instability of the appearance features because of change of lighting and distance.

### **2.3.4 Hand Detection and Motion Capturing**

Gesture detection is one of the fundamental problems for gesture-based interfaces. Although special markers [141] and sensors [78, 92] can ease the difficulties, for general applications, much effort is required to solve this problem.

In many cases, the problem can be considered as an image segmentation problem that segments the human hand/arm from the scene [1, 48, 49, 51, 60, 61, 68, 124, 141]. Color plays an important role in detecting human skin regions from the image. Commonly used approach include color histograms [61, 68, 124, 152], look-up tables, and color density modeling [141]. One of the difficulties of color-based segmentation and detection is the variance of skin color due to imaging condition and discrepancy among different persons.

Motion cues can also be used to detect the hand if we assume that the background and other parts of the user's body are static. The multi-model approach [50] combines multiple visual cues to perform segmentation. Pentland, et al. [141] presented a multi-class statistical model of color and shape to obtain a 2D representation of head and hands. Another commonly used approach is to predict the position based on the past history and known dynamics of the motion.

One of the important issues in gesture modeling is to track the hand and learn the dynamics of the gesture. Kalman filter and extended Kalman filter are widely adopted to model the dynamics of the hand [92, 141]. However, the assumption of small motion is not always satisfied. Other general tracking methods are also implemented for gesture motion

capture, such as particle filtering [37], and dynamic Markov networks [142].

### 2.3.5 Visual Feature Extraction

To a large extent, the selection and extraction of visual features is directly determined by the model used to characterize the gesture. As mentioned in Section 2.3, appearance-based and model-based approaches are used to model the spatial structure of the hand. For the appearance-based approach, the whole hand image can be used as the feature [39, 149]. Another way is to directly use the whole sequence by building so-called *motion history image* [13, 82], which combines the motion information of the gesture sequence by accumulating the motion of the image points over the sequence.

Silhouettes are also popular image features. Geometric moments [82] are often used to characterize the shape of the silhouette, as well as orientation histogram and shape context [8, 85]. These features have the advantage of relative invariance against lighting changes. However, they tend to be over-simplified and miss many visual cues. Another feature to model the appearance is the image contour, which can be extracted using an edge detector [40, 41, 79]. It can also act as the basis of matching between a 3D model and the image in a model-based approach.

Because of their distinct geometric properties, fingertips are widely used in both the appearance-based and model-based approaches [54, 92, 99, 106, 162]. Cylindrical models of the finger and the hemispherical structure of the fingertip are often used in detecting fingertips [54, 92]. A conic model is used in [162] to detect the fingertip. Other heuristics can also help the fingertip detection. For example, in some applications, the fingertips are the foremost points of the hands [152].

Motion [92, 151] is also an important feature for modeling dynamic gestures. In [92], the average motion of the fingertips is extracted as the input to recognize dynamic gestures. Motion is formulated by differentiating 3D appearance features in [151]. The appearance feature itself is computed in a site-centered fashion to characterize the local scene around the manipulated object.

### 2.3.6 Gesture Recognition

Both static postures and dynamic hand movements can represent gestures. Since the definition of valid gestures is largely dependent on the application environment, there is no universal gesture vocabulary.

#### Static Gesture Recognition

In essence, the recognition of gestures is to find a good gesture model and perform optimal partitioning of the parameter space of the model. Thus, the accuracy of the recognition largely depends on the efficiency of the model to characterize the gestures and to separate different gestures. For appearance-based approaches [33, 126, 144, 163], which map the model space directly into the visual features, the key task is to extract and select characteristic image features from the training data. Many learning techniques, such as vector quantization (VQ) [64], principal component analysis [107] and linear discriminant analysis [38], have been investigated to handle this problem.

Another approach is 3D model-based [106], which makes use of a 3D hand model to represent gestures. The mapping between the model and the image are often based on matching of such image features as edges, contours, etc. Analysis-by-synthesis is also popular to ease the matching process [2, 39, 78, 125]. In [2], the 3D hand model is represented as a set of synthetic images of the hand with different configurations of the fingers under different viewpoints. Image-to-model matching is carried out using Chamfer distance computation. One of the difficulties of this approach is to construct a good 3D model of the hand that can deal with the variance between different users. Furthermore, efficient algorithms are necessary to handle the matching between models and input images.

#### Temporal Gesture Recognition

Many gestures demonstrate strong temporal characteristics. Thus it is natural to capture the hand dynamics to model these temporal gestures. An intuitive approach is to use a template matching approach that directly carries out the comparison between the given sequence and pre-stored templates. One example is to match a motion history image [13, 82].

Since a temporal gesture can be viewed as a trajectory in 3D space, matching the trajectory directly or in parameter space can also be a way to classify gestures [82, 69, 95, 116].

The rule-based method tries to impose semantic constraints to represent gestures [83, 99]. Similarly, the state-based approach models the rules in a finite state machine [14, 15, 36, 58]. Characteristic gesture events are represented using linguistic symbols or states, while the rules or transitions between states define the dynamic relationship between events.

The HMM and its various extensions have gained huge popularity among gesture-based interfaces [19, 39, 45, 86, 100, 102, 139, 141, 149, 150, 152, 156]. The attractiveness of HMMs include their sound mathematical basis and efficient algorithms, the capability to learn and adapt to training data, relative robustness against variance of gestures, etc. The parametric HMM [139] is proposed to include a global parameter in the output probabilities to handle parameterized gestures. The coupled HMM [19] is used to model interacting processes that may have different state structures. General graphical models, such as the dynamic Bayesian networks and finite-state machines, can also be viewed as a generalization of HMMs.

## 2.4 Applications of Gesture Interfaces

There exist many potential applications for gesture-based interfaces. One application area is to use gestures to replace traditional HCI media to control computers and other machinery [54, 55, 81, 99, 105, 140, 152, 162]. A bare-hand HCI framework is proposed in [54]. The hands are segmented from background via image differencing. The fingertips of the user's hands are detected and grouped. Gestures are defined as various configuration of the fingers. Zhang, et al. [162] present a "visual panel" system that also uses a single fingertip as the sole interaction medium. The lack of a temporal model for dynamic gestures is resolved via a simple timer, i.e., a certain gesture command is issued when the user stays in the same gesture configuration over a predefined period of time. The SocialDesk system [81] uses shadow infrared scheme to segment the user's hand from background, which is a projected desktop. Gesture commands are simply modeled as a transition of two postures.

One limitation is the 2D nature of the system, which can not handle 3D gestures.

Another popular application area for gesture-based interface is for virtual environments [24, 28, 30, 31, 71, 92, 102, 113, 118, 121, 151]. In many virtual environments, manipulative gestures are employed to interact with virtual or real objects. Communicative gestures [7, 141] can also be used to communicate with the virtual environments. The Perceptive Workbench [121] system employs multiple infrared light sources and a common black/white camera with infrared filter to detect the 3D shape of the hand and the objects inserted into the system. Based on tracked hand position and pointing direction, deictic gestures are recognized. Several applications, such as terrain navigation and virtual game playing, have been built over the system. The gesture vocabulary is limited compared to the complexity of the system and the capability of 3D construction.

Automatic recognition of gestures [45, 122, 134] as general means of communication are also promising. ASL and other sign languages are well structured and their applications potentially play an important role for communication with people who have difficulty using spoken language. Starner, et al. [122] recognize real-time continuous ASL using HMMs. The gestures are captured using a single color camera either mounted on a desk or on the cap of the signer. Moment features of the blobs of the hands are extracted after color-based image segmentation. Forty words are modeled and a simple context grammar is incorporated to improve the sentence-level recognition. The sentences conform to such a form of “personal pronoun, verb, noun, adjective, personal pronoun”. The scheme achieves recognition accuracy of 92% on a desk mounted system and 98% on a cap mounted system. Vogler, et al. [134] model and recognize continuous sign sentence by coupling HMMs and 3D visual tracking system. The vocabulary consists of 53 signs. Their experiments show that incorporating 3D information improves the recognition accuracy. One major problem in continuous sign language recognition is the lack of labeled data to train a good high-level language model.

Since human being has been using speech and gesture, as well as other communication modes in daily communication, it will be ideal to build a multi-modal interface to make the full use of these communication media [67, 70, 91].

Table 2.2 lists some existing gesture systems. With the increase of computing

power as well as better understanding of gesture modeling and recognition, many more promising applications will emerge in the future.

One explicit limitation of current research in visual modeling of gestures is the limited gesture vocabulary and limited number of human subjects, as shown in Table 2.2. Most systems that model dynamic gestures have a vocabulary of less than ten gestures. Furthermore, in most experiments, there are only very few human subjects. These limitations are partly caused by the difficulty of collecting labeled visual data and the lack of a good gesture model. In Chapter 5 and Chapter 6, we will address these problems.

## 2.5 Conclusions

In this section, we have reviewed related work in multimodal interaction and vision-based interaction. Particularly, we have presented research in multimodal interfaces where vision is combined with speech and/or haptics for natural human-computer interaction. We also discussed vision-based systems which are based on visually modeling of human activities for intuitive interaction. Because of its importance, gesture has been a popular choice for vision-based interaction. We have reviewed research in gesture-based systems and have discussed several important aspects of visual modeling of gestures: gesture definition, modeling, analysis, recognition, and applications.

Our review reveals several limitations of current research in related areas. First of all, applying vision to intelligent human-computer interaction is a hard problem and current research is still in infant stage. Great efforts are necessary to build an applicable multimodal interface. Second, visual modeling of gestures for natural interaction are limited in such crucial areas such as visual data collection, efficient motion capture, gesture modeling, recognition of a relatively large vocabulary, and handling variances across a large group of users. In the rest of this dissertation, we will present our research that tries to address these problems.

| <b>Application</b>         | <b>Modeling</b>                     | <b>Recognition</b>              | <b>Vocabulary</b>               | <b>Complexity</b>          |
|----------------------------|-------------------------------------|---------------------------------|---------------------------------|----------------------------|
| ASL [122]                  | Silhouette moments                  | HMM and context grammar         | ASL sentence                    | 40 words                   |
| Waving and pointing [15]   | Position and normal of palm         | State-based                     | Waving, pointing                | 2 gestures                 |
| Gesture VR [112]           | Position and orientation of fingers | Finite state and tracking       | Pointing, reaching and clicking | 3 gestures                 |
| Bare-Hand interaction [54] | Fingertips tracking                 | Postures and duration           | Postures controlling GUI        | 5 gestures, multiple users |
| HOWARD [118]               | Ellipses of hand and object         | HMM                             | Manipulative gestures           | 12 gestures, 9 users       |
| Visual Panel [162]         | Tracked fingertip                   | State-based & duration          | Clicking and dragging           | 2 gestures                 |
| 3D Gestures [78]           | Features sampled on depth image     | K-nearest-neighbor              | Greek sign language             | 20 gestures                |
| Hanoi [83]                 | Blobs                               | Stochastic grammar              | Gestures playing game           | 3 gestures                 |
| 3D Tracking [125]          | Mellin transform of contour         | VQ-PCA                          | Hand signs                      | 24 signs                   |
| Enhanced Desk [92]         | Fingertip tracking                  | HMM                             | Figure drawing                  | 12 gestures                |
| Direct Manipulation [129]  | Tracking and 3D reconstruction      | Shape matching and tracking     | Scene creation                  | 8 gestures                 |
| Wu [144]                   | Appearance description              | MDA and EM                      | Gesture commands                | 14 gestures                |
| Continuous Sign [45]       | 3D Sensor                           | HMM and DTW                     | Sign language                   | 5000 signs                 |
| Shin [116]                 | Tracking                            | Geometric feature of trajectory | Manipulative gestures           | 6 gestures                 |

Table 2.2: Example gesture systems.

## Chapter 3

# Augmented Reality Combining Haptics and Vision

Recently, haptic devices have been successfully incorporated into the human-computer interaction model. One common drawback of most current haptic systems is that the user's hand must be attached to the haptic device at all times. However, in many situation, the force feedback is not always being rendered. This constant binding of the user's hand and the haptic device hinders perception of the virtual environment, mainly because it prevents the user from feeling new tactile sensations upon contacting with various virtual objects.

To eliminate the constraint of constant contract, we present the design and implementation of an augmented reality system called VisHap. It uses visual tracking to seamlessly integrate force feedback with tactile feedback to generate a “complete” haptic experience. The VisHap framework allows the user to interact with combinations of virtual and real objects naturally, thereby combining active and passive haptics. An example application of this framework is also presented. The flexibility and extensibility of our framework is promising in that it supports many interaction modes and allows further integration with other augmented reality systems.

---

<sup>1</sup>Parts of this chapter are jointed work with Prof. Dr. A. Okamura and J. Corso.

### 3.1 Introduction

In recent years, many human-computer interaction and virtual environment systems have incorporated haptic devices. A general survey reveals that in most haptic systems, the user must be constantly attached to the haptic device in order to feel the generated forces. The user typically grasps a stylus or places a fingertip in a thimble. Continuous contact with a physical tool hampers the perception of the virtual environment and human-computer interaction in many ways. Since the user’s hand is attached to a fixed object all the time, it is impossible for the user to feel any new tactile sensations upon contact with virtual objects. In general, haptics includes both kinesthetic (force) and cutaneous (tactile) information. Most commercially available devices only apply force feedback. Devices specifically designed for tactile feedback, e.g. [56, 135], are typically complex and not yet integrated with force feedback devices.

Furthermore, most haptic devices have a very limited workspace. For example, the workspace of the PHANToM Premium 1.0A model [25, 80], which is our experimental platform, is approximately a  $13\text{cm} \times 18\text{cm} \times 25\text{cm}$  rectangular solid. If the user has to attach his or her hand to the device all the time and move his or her hand in such a limited space, it would be hard to extend the virtual environment to incorporate rich interaction elements. In addition, constant contact impairs the experience of the virtual environment because it acts as a constant reminder to the user that he or she is interacting with a virtual world through a tool.

To overcome these drawbacks, we designed and implemented an augmented reality system that employs visual tracking to seamlessly integrate force feedback with tactile feedback in order to generate a “complete” haptic experience. The basic idea is to incorporate a computer vision system to track the user’s movement so direct contact via a physical tool becomes unnecessary. The framework also allows the user to interact with combinations of virtual and real objects naturally, thereby combining active and passive [62] (or kinesthetic and cutaneous) haptics. Our work builds upon previous research in encountered-type haptic displays [158, 159], with the goal of creating a higher fidelity haptic interaction with both passive and active elements.

The VisHap system is composed of three subsystems: computer vision, the haptic device, and an augmented environment model, which is also called the world subsystem. We explain them in detail as follows:

1. **Vision Subsystem:** which tracks the movement of the user's finger and transfers the 3D position and velocity of the finger to the augmented environment model. Our current implementation includes stationary static stereo cameras and XVision [53].
2. **Haptic Device,** which is controlled as a robot to meet the finger at the point of contact with a virtual object. Once contact is made, a hybrid control allows the user to feel virtual dynamics in the direction normal to the object surface, while maintaining the position of the haptic device directly behind the finger. The character of this feedback depends on the environment model.
3. **Augmented Environment Model:** which defines the physical configuration and interaction properties of all the virtual and real objects in the environment, such as the position and surface properties of the virtual objects and the passive objects attached to the haptic device end-effector.

In the VisHap system, these three modules are cooperating in a coherent fashion. Once the user is close enough to a virtual object, the augmented environment model sends a command to the haptic device to prepare to meet the user in space. When the user is interacting with the object, the augmented environment model continuously sends the position of the user to the haptic model, and the haptic device applies force feedback to the user's finger according to the positions of the finger and the object, as well as the dynamic properties of the object. For example, the virtual object may be designed to allow the user to push a virtual button, slide along a virtual wall, or hit a virtual ball. In each case, a proper passive element is attached to the haptic device end-effector and the corresponding interaction mode is defined. Our framework is flexible and extensible because it supports many interaction modes and allows further integration with other augmented reality systems, such as head-mounted displays.

In Section 3.2 we describe the framework and implementation of our system, as

well as some example applications. We present experimental results for our system in Section 3.3. Section 3.4 provides the conclusions drawn from this work.

### 3.1.1 Related Work

Incorporating haptics into virtual environments is a promising field. Insko et al. [62], show that augmenting a high-fidelity visual virtual environment with low-fidelity haptics objects, which they call “passive haptics”, can increase participant’s sense of presence as measured by subjective questionnaires, based on observed participant behaviors and physiological responses. Experiments show that in navigating an identical real environment while blindfolded, those participants trained in a virtual reality (VR) augmented with passive haptics performed significantly faster and with fewer collisions than those trained in a non-augmented virtual environment. The augmented VR is implemented using styrofoam model for passive haptics. The participants who experienced the virtual environment not augmented with the styrofoam model were given audio and visual contact cues; while in augmented VR, the participants had the physical haptic cues and the visual observation.

Salada et al. [109, 108] investigated the use of fingertip haptics to directly explore virtual environments instead of via an intermediate grasped object (a tool). They render the relative motion between the fingertip and the object surface using a rotating drum or sphere. Their experiments show that relative motion between the fingertip and the object surface is a key psychophysical aspect of fingertip haptics.

The touch/force display system [159] is probably the first system based on the encountered-type haptic device concept. An original optical device was designed to track the finger position. When the finger is in contact with a virtual object, the device contacts the skin. However, it not possible to attach additional passive objects to the device.

Yokokohji et al. [157] studied the problem of multiple objects in an encountered-type virtual environment. They present path planning techniques to avoid collisions between the hand and the robot, for a single robot attempting to apply multiple encountered-type virtual haptic objects.

Yokokohji et al. [158] implemented a haptics/vision interface, WYSIWYF (What You See Is What You Feel), for VR training system for visuo-motor skills. Using visual

tracking and video keying, the system is registered so that the visual and haptic displays are consistent spatially and temporally. A Puma robot is used to simulate the physical interaction, so high fidelity haptic sensations cannot be conveyed.

### 3.2 System Design and Implementation

The VisHap system is composed of three parts, called the vision, haptics and world subsystems. Figure 3.1 illustrates the configuration of the system. The user is interacting in an augmented reality that is composed of several virtual planes around the workspace and a passive key (removed from a conventional computer keyboard) attached to the end of a haptic device. A stereo camera system tracks the finger in 3D space. The user can observe the scene on a standard PC monitor or a head mounted display (HMD).

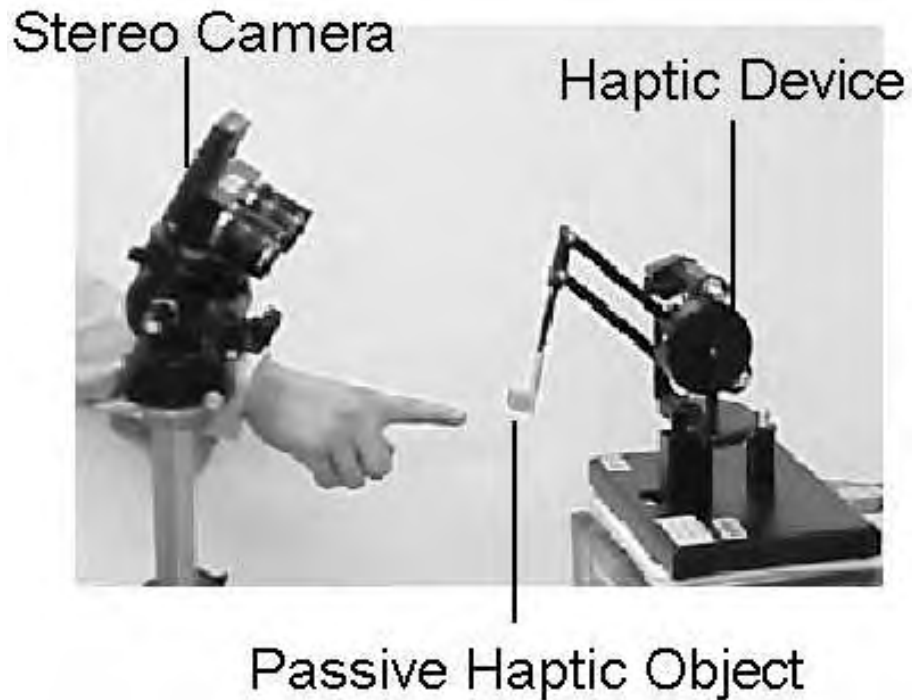


Figure 3.1: Overview of the VisHap system.

The vision subsystem is composed of a stereo camera and tracking software. It is responsible for capturing the scene and tracking the user's finger in real time. Since 3D

registration between the vision and haptics modules is required, the cameras are calibrated in order to calculate the 3D position of the finger in the coordinate system of one of the cameras. To enhance the flexibility of the system, we used automatic finger detection and tracking without any attached marker or special imaging media.

The haptic subsystem, which consists of a 3D haptic device, is the interface between the virtual/augmented environment and the user. When the user is far from objects in the environment (set by a threshold), the haptic device is motionless and the user moves his finger freely in space. When the user is in contact with a virtual object, the haptics module simulates the sensation of the interaction through force feedback to the user's finger. To simulate different objects under different interaction scenarios, the haptic module produces corresponding force feedback to the user. In order to display appropriate tactile sensations, a real "passive" haptic object is attached to the end-effector of the haptic device.

The world subsystem acts as the overseer of both vision and haptic subsystems. It is responsible for defining the configuration of the whole augmented reality, specifying the properties (e.g., position, orientation, dimensionality, surface property, etc.) of all virtual and real objects, rendering the virtual environment to the user if necessary, and carrying out the 3D registration between vision subsystem and haptic subsystem. At any time, it queries the vision system to check whether the user's finger is in the scene, and if so, transforms the 3D position and velocity of the finger to the world coordinate system. When the finger is close to certain virtual or real objects in the environment, which indicates that an interaction is possible, it sends the positions of the finger and object to the haptic subsystem and notifies the haptic subsystem to prepare for the coming interaction. During interaction, it continues sending the necessary information to the haptics module. Graphical rendering of the environment can be implemented using standard computer graphics and video processing techniques. For augmented reality, the video of the scene is displayed to the user, with the haptic device "deleted" and a virtual object overlaid. A head mounted display can be used to achieve better immersiveness.

Our framework has several advantages compared to the encountered-type systems discussed in Section 3.1 and those that require constant contact with a haptic device. First, it allows a much richer set of haptic interactions, including kinesthetic and cutaneous sen-

sations. Interaction with real “passive” haptic objects is easy to configure. Second, the system is very flexible. The configuration of the scene, the properties of virtual objects and the mode of interaction can all be customized by editing corresponding configuration files, without many changes to the hardware. Furthermore, almost all hardware components in our design are standard devices, therefore easing the difficulties of implementation and integration. A standard PC, with a stereo camera system and a haptic device, is the minimum requirement. Some of the software implementations are recent research topics [25, 54, 75, 92, 162] and even free to download [53].

In our current implementation, the entire system runs on a Pentium III PC with the Linux operating system. A SRI Small Vision System (SVS) with a STH-MDCS stereo head by Videre Design is the imaging unit. A PHANToM Premium 1.0A model [25, 80] from SensAble Technologies is the device used to simulate haptic interaction.

### **3.2.1 Vision Subsystem**

As mentioned earlier, the main task of the vision subsystem is to track the user’s finger and provide 3D information and video to the world subsystem. Using the SVS system, we capture real-time image pairs of the scene and calculate disparity information to acquire 3D data. In our current implementation, we assume that the user is interacting with the scene using a single finger. We perform fingertip tracking on the left color image and compute the 3D position of the finger in the coordinate system of the left camera.

#### **Appearance-based Hand Segmentation**

An advantage of our system is that it does not require any special marker on the user’s finger. Instead we perform efficient and robust appearance-based skin segmentation on the color image [124, 150, 152]. The basic idea of the appearance model is to split the image into small image tiles and build a hue histogram for each of the image patches. At the start of each session, we carry out a fast on-line learning procedure of the background scene by averaging the first several images and building the appearance model for the averaged image. For future images, we also build the histogram for image patches in the region of interest and carry out pairwise histogram comparison with the background model.

Histogram intersection [124] is an efficient way to match histograms.

$$H(I, M) = \frac{\sum_{j=1}^n \min(I_j, M_j)}{\sum_{j=1}^n M_j} \quad (3.2.1)$$

Here  $I$  and  $M$  refer to the model and measure histograms, respectively. The match score is the criterion of foreground segmentation.

Another offline procedure is carried out to learn the color appearance model of human skins. We collect the training data by recording image sequences with segmented hands. We convert all skin pixels from RGB color space to HSV color space and learn a single Gaussian model of the hue distribution. We perform a skin/non-skin check on every foreground pixel by thresholding the probability that the given pixel belongs to the skin model, and then filter out non-skin points. Then a median filter operation is used to remove noise. The result of the whole hand segmentation procedure is a binary image with foreground pixels indicating the skin region. Figure 3.2 shows an example segmentation result.



Figure 3.2: An example of background image (left), foreground image (middle) and segmentation result (right).

### Fingertip Detection and Tracking

An efficient way to detect the fingertip is to exploit the geometrical property of the finger [54, 92, 162]. We use a cylinder with a hemispherical cap to approximate the shape of the finger. The radius of the sphere corresponding to the fingertip is approximately

proportional to the reciprocal of the depth of the fingertip with respect to the camera. We model this relationship with the following equation.

$$r = \frac{K}{z} \quad (3.2.2)$$

Here  $r$  refers to the radius of the fingertip and  $z$  corresponds to the  $z$ -coordinate of the fingertip in the camera system.  $K$  is determined by the experiment configuration.

A series of criteria are checked on the candidate fingertips to filter out false fingertips. The following pseudo-code segment illustrates the algorithm.

$\forall p(x, y) \in \text{search region}$

1. IF (  $p(x, y)$  is not a skin pixel ) CONTINUE LOOP
2. Calculate the  $z$ -coordinate of  $p$  in the camera system
3. Compute desired fingertip radius  $r = \frac{K}{z}$
4. Calculate number of skin pixels  $S_{filled}$  in the circle  $C$  centered at  $p$  with radius  $r$
5. IF (  $S_{filled} < \text{area of the circle } C$  ) CONTINUE LOOP
6. Compute number of skin pixels  $S_{square}$  along a square  $S$  centered at  $p$  with side-length  $r_2 = r + \delta r$
7. IF (  $S_{square} < 2 \times r_2$  or  $S_{square} > 4 \times r_2$  ) CONTINUE LOOP
8. Check pairwise diagonal pixels along  $S$  and calculate number of pairs  $N$  that both pixels are skin points
9. IF (  $N > 1$  ) CONTINUE LOOP
10. Record  $p(x, y)$  as a candidate fingertip with  $score = S_{filled} / \text{area of the circle } C$

The circle assumption of the fingertip is enforced by checking the number of points in the neighboring circle. The square is examined to make sure that there is a cylinder of reasonable size connected to the circle. The diagonal check aims to remove false fingertips that may appear in the middle of the finger. In most cases this algorithm outputs multiple

candidates around the true location of the fingertip. We select the one with the highest score to be the fingertip. Figure 3.3 displays an example detection result.

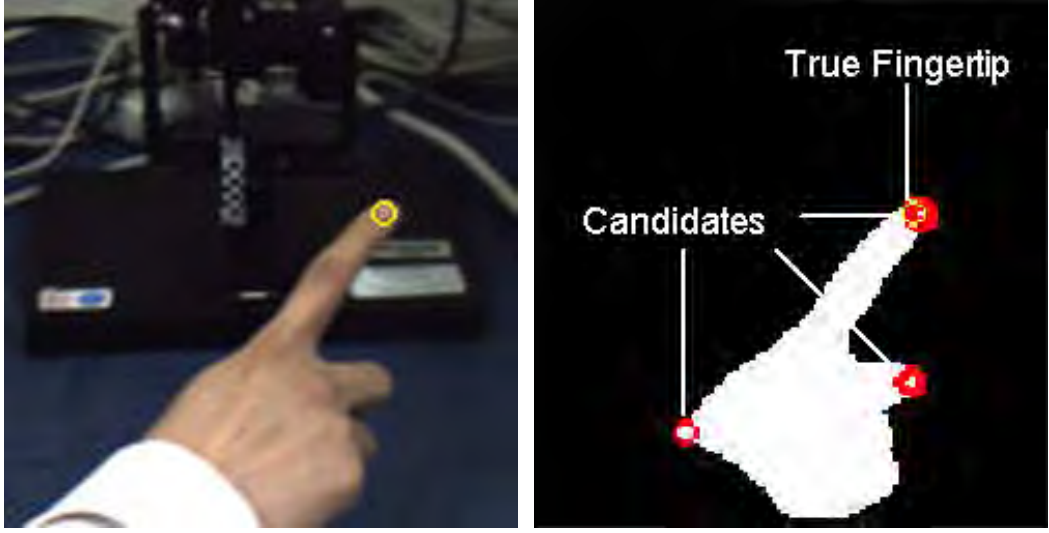


Figure 3.3: Fingertip detection result: the foreground image and detected fingertip specified with a yellow circle (left) and the candidate fingertips on the segmented foreground image (right).

We implement a simple Kalman Filter [92, 119, 137] to predict the position of the fingertip in each frame to improve real time fingertip tracking. We assume that the fingertip moves approximately at a constant velocity and in a straight line. Following these assumptions, we use  $x_t = (x(t), y(t), v_x(t), v_y(t))^T$  to represent the system state vector which contains the 2D position and velocity of the tracked fingertip. The system dynamics are characterized as:

$$x_{t+1} = Fx_t + Gw_t \quad (3.2.3)$$

$$y_t = Hx_t + v_t \quad (3.2.4)$$

Here,  $F$  is the state transition matrix,  $G$  is the driving matrix and  $H$  is the observation matrix.  $w_t$  and  $v_t$  is the system noise and observation noise. Based on our assumption, we specify them as follows:

$$F = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2.5)$$

$$G = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2.6)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (3.2.7)$$

At each frame, we use this Kalman filter to predict the position of the fingertip. Then we search for the fingertip in a small window around the predicted position. The search method is similar to fingertip detection. Our experiments show that this algorithm tracks the fingertip quite accurately and robustly. The position of the tracked fingertip is computed in the coordinate system of the left camera frame by combining calculated disparity image and the camera parameters.

### 3.2.2 World Subsystem

As the overseer of the entire VisHap system, the world subsystem is responsible for performing 3D vision/haptics registration, scene rendering, notifying the haptic device about imminent interaction, etc.

#### Vision/Haptics 3D Registration

Since the coordinate systems of the camera and haptic device are canonical Euclidean frames, they are related by a rigid transformation:

$${}^H\vec{P} = {}^H_C R \ {}^C\vec{P} + \vec{t} \quad (3.2.8)$$

Here  ${}^H\vec{P}$  and  ${}^C\vec{P}$  refer to the coordinates of point  $P$  in haptic and camera coordinate system, respectively.  $R$  and  $\vec{t}$  denote the rotation and translation between the two coordinate systems. During the system calibration process, we move the haptic device

around in the field of view of the camera system and record more than three pairs of coordinates of the end-effector in both the camera and haptic frames. Then  $R$  and  $\vec{t}$  are calculated as the optimal absolute orientation solution [59, 75]. We carry out the calibration using the SVS system and PHANToM 1.0A model and achieve highly accurate results. The average error in each dimension is less than  $0.5mm$ .

### Scene Configuration

We implemented example applications of the framework in which the user interacts with a virtual wall or button. We define the wall as a plane with parameter  $P = (\vec{n}, d)$  where  $\vec{n}$  and  $d$  are the normal and the distance of the plane to the origin, respectively. The button is defined as  $B = (\vec{p}, \vec{n}, w, h)$  with  $\vec{p}$  and  $\vec{n}$  specifying the position of the button and the normal of the surface, while  $w$  and  $h$  indicating the size of the button. To enhance the fidelity of haptic experience, we attach appropriate passive objects to the haptic device, such as a flat metal piece, a computer keyboard key, etc.

We define different interaction properties corresponding to different objects. A simple way to implement this is to define a database of various interaction modes and object surface properties. For example, a button allows a click, and a hard wall only allows a sliding along the surface. For each object we only need specify the index into a database that describes its interaction property.

### 3.2.3 Haptic Subsystem

The main task of the haptic subsystem is to simulate the touching experience by presenting suitable force feedback to the user’s fingertip. For combined tracking and force feedback, a control scheme is necessary.

#### Control Law for the Haptic Device

Figure 3.4 illustrates the closed-loop PD control law. This control law is based on error space and intended to guide the haptic device to a target position that is determined by the world model and current interaction status. For example, if the user is interacting with a virtual plane in space, the target position of the haptic device,  $X_d$ , is the projection

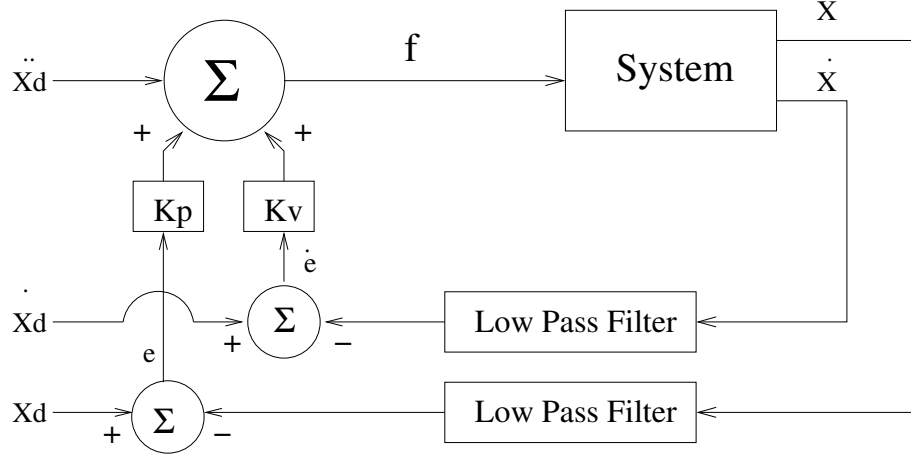


Figure 3.4: Control scheme in error space.

of the fingertip onto the plane, i.e., the intersection of the plane and the line that is parallel to the normal of plane and passes through the point corresponding to the current position of the fingertip. Parameters  $K_p$  and  $K_v$  are adjusted experimentally to make the system critically or over damped. The control law is:

$$\ddot{e} + K_v \dot{e} + K_p e = f \text{ where } e = X_d - X \quad (3.2.9)$$

Here  $X_d$  and  $X$  refer to the desired and current position of the device, respectively.  $f$  is the impedance force applied to control the haptic device. The desired position is determined from the position of virtual objects and the user's fingertip. The force  $f$  is scaled appropriately for interaction with virtual objects, as described in Section 3.2.3.

To solve the problem of the large difference in frequency between the vision subsystem, which normally runs at no more than 20Hz, and the haptic subsystem, which typically runs around 1KHz, we add a low pass filter on  $e$  and  $\dot{e}$  to achieve smooth control and to remove high-frequency noise.

$$y = \frac{ax}{s+a} \quad \text{or in time space} \quad y_t = \frac{ax + y_{t-1}}{1+a} \quad (3.2.10)$$

Here  $y$  and  $y_t$  refer to the filtered result of  $x$ , while  $a$  is a constant that characterizes the filter. This control law is used in each degree of freedom of the haptic device.

## Gravity Compensation for PHANToM

The manufacturer of the PHANToM provides a counter-weight attached to one of the motors that maintains the equilibrium of the device without additional forces. In our experiment, we attach other objects to the endpoint for the purpose of interaction instead of the gimble that is typically counterbalanced. Without additional gravity compensation, the device has a tendency to fall into a degenerate configurations from which it is almost impossible to recover. Thus, we implement a simple gravity compensation scheme. As shown in [25], the following equation gives the motor torques required to counteract the wrench  $F$  applied to the manipulator:

$$\tau = J^{bT} R^T F \text{ or } F = (J^{bT} R^T)^{-1} \tau \quad (3.2.11)$$

where  $J^b$  is the body Jacobian of the manipulator and  $R$  is the rotation matrix of the forward kinematics.

We calculate the total torque caused by the gravity of all the parts of the device as:

$$\tau_g = \begin{pmatrix} 0 \\ g(m_a l_1 + 0.5 l_1 m_c + m_{be} l_5) \cos \theta_2 \\ g(0.5 m_a l_2 + m_c l_3 - m_{df} l_6) \sin \theta_3 \end{pmatrix} \quad (3.2.12)$$

The definitions of the variables used in this equation are the same as in [25].

By adjusting  $m_a$  in Equation 3.2.12 we can calculate the gravity torque  $\tau_g$  for the device with attached objects. Using Equation 3.2.11, the force  $F_{GC}$  is computed to compensate for gravity. Combining  $F_{GC}$  and weighted  $f$  calculated from control law, we are able to achieve smooth and stable trajectory tracking.

$$F = F_{GC} + \Lambda_{gain} f \quad (3.2.13)$$

where  $\Lambda_{gain}$  is the matrix that controls the force gain. When the user is not interacting with any object,  $\Lambda_{gain}$  is the identity matrix.

## Interaction with Objects

When the user’s finger is touching a virtual or real object, we simulate the interaction forces by adjusting the force gain  $\Lambda_{gain}$  in Equation 3.2.13 according to the object properties and interaction mode. For convenience, we define  ${}^O\Lambda_{gain}$  for each object in its own coordinate system as this object’s gain matrix. A similarity transform converts the object’s gain matrix to that of the haptic device  ${}^H\Lambda_{gain}$ .

$${}^H\Lambda_{gain} = {}^H_O R^T {}^O\Lambda_{gain} {}^H_O R \quad (3.2.14)$$

where  ${}^H_O R$  is the rotation matrix between the frame of the object and the haptic device.

In our current implementation, we define  ${}^O\Lambda_{gain}$  as a diagonal matrix with  $\lambda_x$ ,  $\lambda_y$  and  $\lambda_z$  referring to its diagonal elements. The  $z$ -axis of the object’s frame is along the normal of the surface of the object. In our experiments where the user interacts with buttons or planes, we adjust  $\lambda_z$  to simulate interaction force while  $\lambda_x$  and  $\lambda_y$  stay constant. For example, in the case that the object is a solid wall, which allows no piercing along its normal direction, we use a very large  $\lambda_z$  when the fingertip is under the plane. Effectively, this creates a linear spring whose force is proportional to the depth of the finger into the virtual object. When the user’s finger enters the object, the haptic device presents a strong force in the direction normal to the surface of the object, in order to push the finger back to the point of contact back on the surface. Figure 3.5 illustrates the relationship of  $\lambda_z$  to the distance of the fingertip under the plane.

Another example is to push a button or a key on the keypad or keyboard. Similar to the wall, we define the destination point of the haptic device as the center of the surface of the button at the time of initial contact. Once the user pushes the button down and enters the object, we increase  $\lambda_z$  to a proper value to simulate the resistance of the button, until after some point the user triggers the button and feels a lower stiffness. Then a much stronger stiffness is felt when the finger pushes the button all the way down to the bottom board. The relationship between  $\lambda_z$  and the depth of the finger into the surface of the button is shown in Figure 3.5. Note we use  $d_1$  and  $d_2$  to represent the value of the finger’s depth before and after the triggering of the button, respectively.

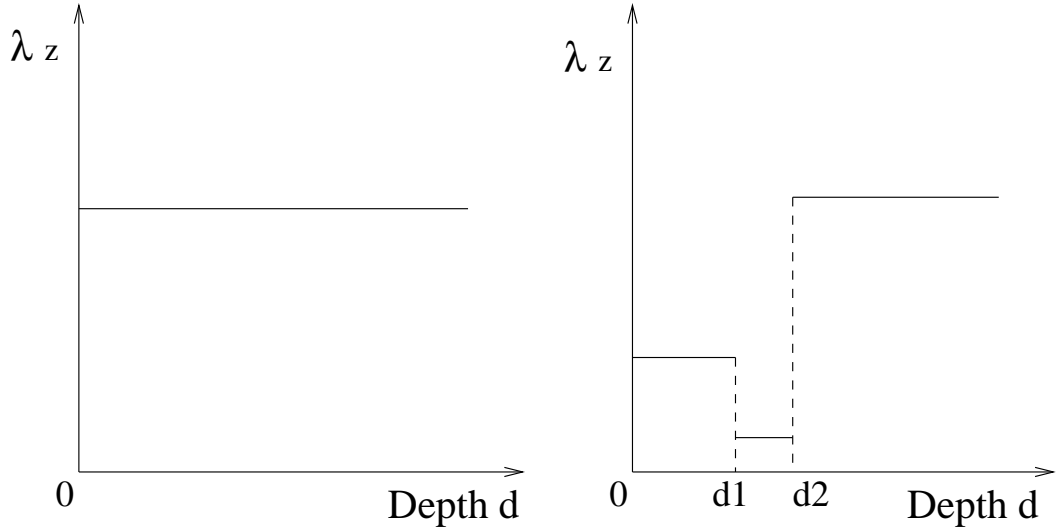


Figure 3.5: Relationship of force gain  $\lambda_z$  and the depth  $d$  of the fingertip under a plane (left) or the surface of a button (right).

A more complex situation is interaction with multiple objects in the same scene. The distance of the fingertip to each object is updated at each frame. The object nearest to the fingertip is chosen to be the current interaction subject. Some methods for approaching this problem are presented in [157].

### 3.3 Experimental Results

For foreground segmentation, we use the first 10 frames to learn the appearance model of the background. We build hue histograms of 8 bins for each of the  $5 \times 5$  image patches. To test the algorithm, we record image pairs of the background and foreground. By comparing the segmentation result and the ground-truth classification image, which is generated by manually marking the foreground part of the scene, we are able to evaluate the scheme. We captured more than 20 pairs of background/foreground images with different background scenes and carried out the experiment on these images. The test set also included 6 pairs of images that undergo illumination changes. As a result, the average correct ratio was 98.16%, with average false positive ratio of 1.55% and false negative ratio of 0.29%.

We set up several interaction scenes and tested the overall performance of the VisHap system. A simple virtual environment consists of a virtual plane in space. The user moves his finger to interact with the plane. Figure 3.6 shows the relationship of the distance of the fingertip to the plane and the average PHANToM force feedback along the normal to the plane. Note that the force shown here does not include the component to compensate for gravity. It corresponds to component of  $\Lambda_{gain}f$  in Equation 3.2.13 in the direction of the normal to the plane. Thus, it is the net force that the user feels along the plane's normal. It can be seen that the force feedback matches our model of the plane as shown in Figure 3.5 very well, i.e., the plane feels like a linear spring.

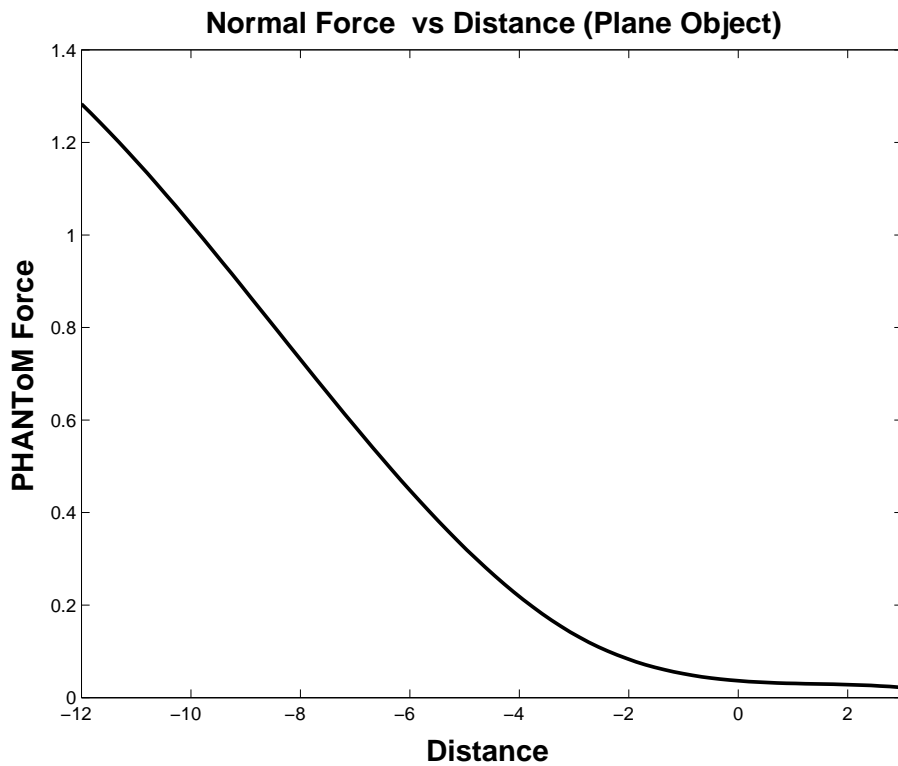


Figure 3.6: Relationship of haptic force feedback along the normal of the plane surface and the distance of the fingertip to the plane. A negative distance indicates that the finger is under the surface.

Figure 3.7 shows a more complex case in which the user interacts with a fixed button. When the user is not in contact with the button, the haptic device is stationary

at the position of the button. When the fingertip touches the button, force feedback is presented in a manner very similar to the model described in Figure 3.5, i.e., two distinct linear springs along the normal of the surface of the button models the force feedback before and after the button is triggered. During the stage when the button is triggered, the user feels much smaller resistance. Note that, in actual experiments, the change of force is a gradual process when the button is triggered. This is necessary to ensure stability of the PHANTOM device.

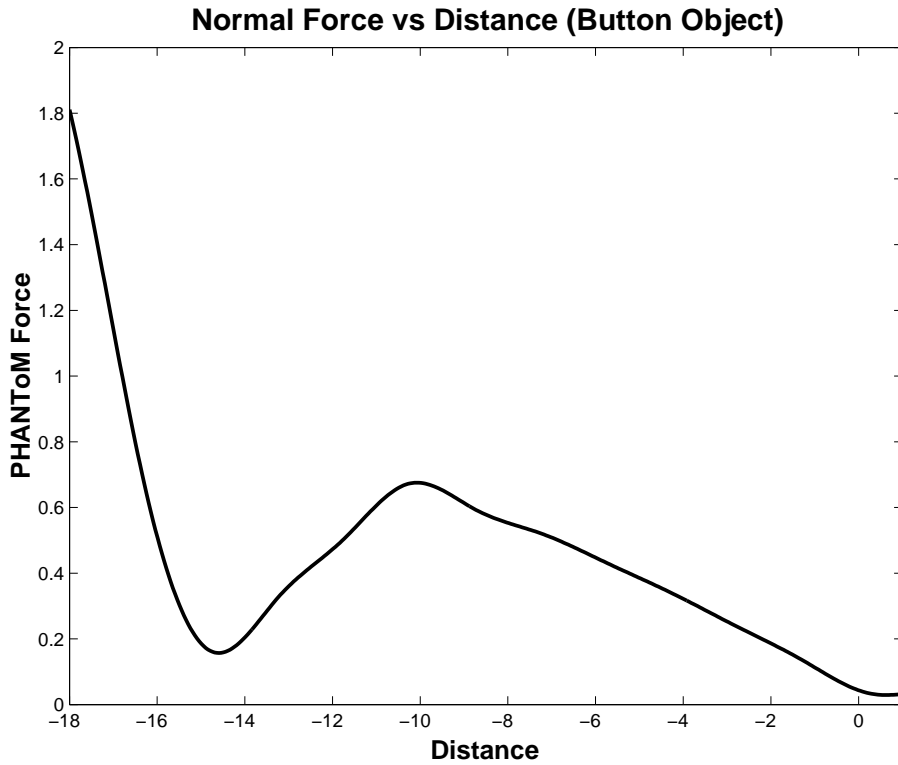


Figure 3.7: Relationship of haptic force feedback along the normal of the button surface and the distance of the fingertip to the button. A negative distance indicates that the finger is under the surface.

We also experimented with more complex scenes with multiple objects. The VisHap system is capable of automatically switching interaction subjects according to the scene configuration and current fingertip position.

### 3.4 Conclusions

A drawback common to almost all haptic systems is that the user must be attached to the haptic device continuously even though the force feedback is not always being rendered. In this paper we present the design and implementation of a haptics-based interaction system that uses finger tracking to overcome this problem and to generate a “complete” haptic experience. We present a multimodal interaction framework that consists of computer vision, the haptic device and an augmented environment model. The key elements of the implementation of the system are also discussed. These include 3-D vision/haptics registration, automatic fingertip detection and tracking, haptic device control and gravity compensation, virtual environment configuration and force feedback rendering. We implemented several example applications on a standard PC and achieved real time performance. The experimental results justify our design and show the flexibility and extensibility of our framework.

Our work builds upon previous research in encountered-type haptic displays [158, 159]. Our goal is to create a higher fidelity haptic interaction with both passive and active elements. This flexibility and extensibility is the most important advantages of our system.

Future improvements of the system include incorporating a head mounted display and rendering the scene and the user’s hand directly on HMD. The advantage of an HMD is that it can achieve higher immersiveness and fidelity. This brings new challenges such as enhanced rendering of real and virtual objects on the HMD and online calibration between the HMD and the vision module if we allow the user to freely move his or her head.

Another goal is to incorporate richer sets of objects and interaction modes to extend the virtual environment. For example, the user could play a puzzle game by moving tiles around on a board, and a rotating drum could simulate sliding. For such applications, better visual rendering is paramount for the user to visually notice the arrangement of such complex configuration of various objects.

## Chapter 4

# The Visual Interaction Cues Paradigm

In the previous chapter, we have presented a novel framework to integrate computer vision and haptics into a multimodal human-computer interaction system. We have shown that, via 3D registration between haptic device and vision system, we can remove the limitation that the user's hand must be attached to the haptic device all the time. Thus, it enhances the fidelity of the haptic experiences. In this chapter and the next two chapters, we will concentrate on techniques that allow natural and intuitive vision-based interaction.

Vision-based human-computer interaction is a promising approach to building more natural and intuitive interfaces. As discussed in Chapter 1, vision techniques allow large-scale, unencumbered motion from multiple concurrent users. They could make use of hand gestures, movements or other similar input means; and video itself is passive, (now) cheap, and (soon) nearly universally available. In the simplest case, tracked hand motion and gesture recognition could replace the mouse in traditional applications. Furthermore, computer vision offers the additional possibility of defining new forms of interaction that make use of whole body motion, for example, interaction with a virtual character [77].

However, using vision in HCI has been proven to be a complicated task, which is evidenced by the absence of vision-based interfaces in production. As discussed in Chapter 1

---

<sup>1</sup>Parts of this chapter are jointed work with J. Corso.

and Chapter 2, most reported work on vision-based HCI relies heavily on visual tracking and visual template recognition algorithms as its core technology. While tracking and recognition are, in some sense, the most popular directions for developing advanced vision-based interfaces, one might ask if they are either necessary or sufficient. For example, complete, constant tracking of human body motion might be a convenient abstraction for detecting that a user’s hand has touched a virtual “button”, but general human motion tracking is well-known to be a complex and difficult problem [46, 104]. What if button contact can instead be detected using simple motion or color segmentation combined with template matching? What if, as in most cases, the user is not interacting at all? Is there any reason to perform potentially expensive image processing to generate negative results? Finally, one might turn the question around and ask whether it is possible to add structure to the interaction problem so as to render the vision problem solvable with high reliability.

In contrast, we present a general approach that does not attempt to globally track and model the user. Our methodology, the Visual Interaction Cues paradigm (VICs), is based on a shared perceptual space between the user and the computer. In the shared space, the computer is monitoring the scene for sequences of expected user activity at the locations corresponding to interface components.

Figure 4.1 shows the difference between VICs and conventional VBI. In conventional VBI, the camera is centered on the user while the user is interacting with the computer. In VICs, the camera is monitoring the interface. In this way, globally tracking the user is not necessary. Instead, the interaction problem is solved by modeling the stream of localized visual cues that correspond to the user interacting with various interface elements. We claim that this additional structure renders a more efficient and reliable solution to the VBI problem.

In the rest of this chapter, we present our VICs paradigm in Section 4.1. We then discuss a particular 2-D implement of VICs framework in Section 4.2 where efficient local features are extracted to recognize a button-push gesture based on Hidden Markov Models.

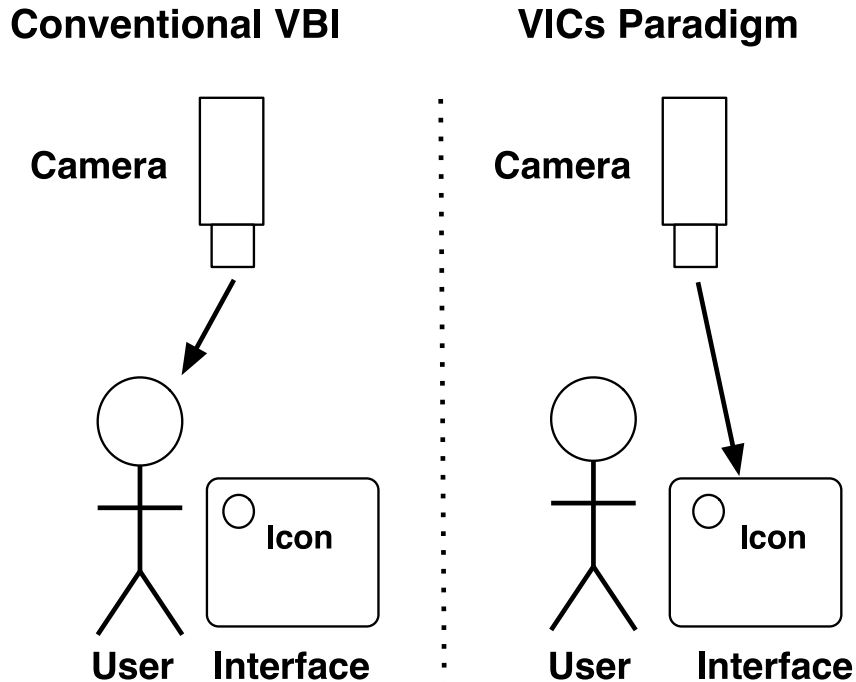


Figure 4.1: Comparison between conventional VBI and VICs paradigm. Left figure shows the concept of traditional vision-based interface; right figure shows the VICs approach.

## 4.1 The VICs Framework

### 4.1.1 Modeling Interaction

Beaudouin-Lafon [6] defines *interaction model* as a set of principles, rules and properties that guides the design of an interface. It describes how to combine interaction techniques in a meaningful and consistent way and defines the “look and feel” of the interaction from the user’s perspective. Current interface technology, WIMP [34], is modeled with a simple state-machine (Figure 4.2). The dominant interface component in these *third-generation interfaces* is the icon. Typically, these icons have one pre-defined action associated with them that is triggered upon a mouse click.

We extend the functionality of a traditional icon by increasing the number of associated actions that can be triggered by the user in a straightforward manner that does not require the user to learn complicated interaction semantics. For standard WIMP

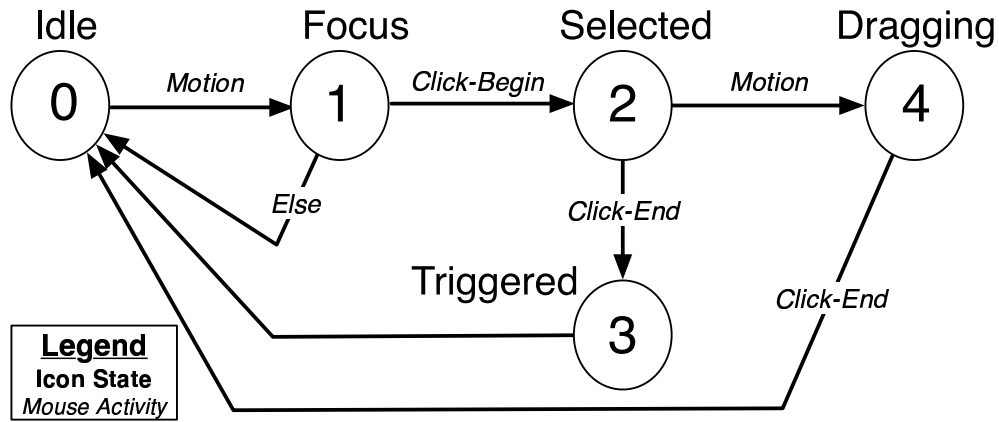


Figure 4.2: The icon state model for a WIMP interface.

interfaces the size of this set is 1: point-and-click. We call a *super-WIMP* interface one that includes multi-button input or mouse-gesture input. One such example is the SKETCH framework [160]. For the *super-WIMP* interfaces the size of this set is larger, but still relatively small; it is limited by the coarse nature of mouse input. Our vision-based extension greatly increases the set of possible user inputs by employing the increased spatial input dimensionality. Figure 4.3 shows a possible state model of our the interface component in our VICs paradigm.

### Principles of VICs

We develop an interaction model that eliminates the limitation of a pointing devices. Instead, we using information from vision streams to parse natural and intuitive gestures. With video input, the user is unencumbered and free to interact with the computer much in the same way they interact with objects in the real-world. The user would bring their prior real-world experiences, and they could immediately apply it in the HCI setting. This model extends the direct interaction model to better utilize the multi-modal nature of future interfaces. The main principles of our model include:

1. **Sited-Interaction:** we claim that the user’s interactive action can be captured in a limited space [99]. All physical interaction with the system should be localized

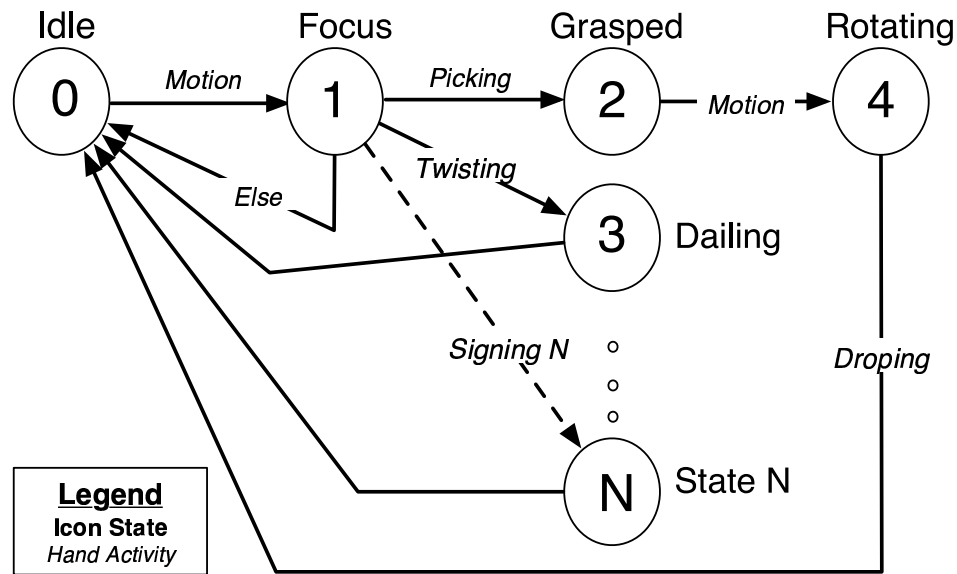


Figure 4.3: The state model for a VICs-based interface component.

to specific areas (or volumes) in the interface to reduce the ambiguity of the user intention.

2. **Responsive Interaction:** continuous feedback to the user during the course of interactions and thereafter is essential to an intelligent and intuitive interface, otherwise the user will probably be confused and discouraged.
3. **Learning-Based Interaction:** there exists a two-way learning process between the user and the interface. The user must learn the set of initial techniques and procedures with which to interact with the system. After that, duplex learning will ensue that the system will adapt to the user and more complex interaction techniques can be learned by the user.

#### 4.1.2 The VICs Architectural Model

The architectural model describes the set of computer vision techniques we use to realize the VICs interaction model in post-WIMP interfaces and the core parts of the

resulting interface. The sited-interaction principle is the basis of the architectural model. Since all physical interaction is with respect to an interface component, we use the video cameras to monitor these components. If a registration is known between the components of the interface and where they project in the video images, then the recognition problem is reduced to one of spatio-temporal pattern recognition. A gesture will appear as a sequence of visual cues in the local image regions near the components. In this section we discuss these three parts in more detail: the component mapping, the spatio-temporal pattern recognition, and the VICs interface component (VICon).

### Interface Component Mapping

Let  $\mathcal{W}$  be the space in which the components of the interface reside. In general,  $\mathcal{W}$  is the 3D Euclidean space  $\mathbb{R}^3$  but it can be the Projective plane  $\mathbb{P}^2$  or the Euclidean plane  $\mathbb{R}^2$ . Define an interface component mapping  $M : \mathcal{C} \rightarrow A(\mathcal{I})$ , where  $\mathcal{C} \subset \mathcal{W}$  and  $\mathcal{I}$  is the image pixel locations as defined in Section 1.6 and  $A(\mathcal{I})$  being an arbitrary function,  $A : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ . Intuitively, the mapping defines a region in the image to which an interface component projects as shown in Figure 4.4.

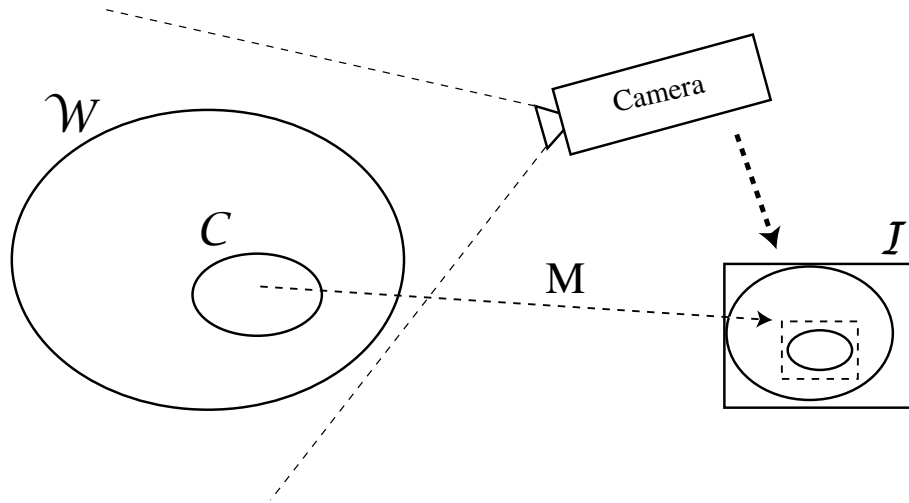


Figure 4.4: Schematic explaining the principle of localized image analysis for the VICs paradigm. Here  $M$  is the component mapping that yields a region of interest in the image  $\mathcal{I}$  for analyzing actions on component  $\mathcal{C}$ .

Given such a mapping, detecting a user action around the component is simplified to analyze a local region in the image. We define the mapping in this way to enforce the principle of sited-interaction.

### **Spatio-Temporal Gesture Recognition**

The component mapping adds a structure to the interaction problem. It disambiguates the process of understanding user activity by reducing the hard problem of global user tracking and modeling to spatio-temporal pattern recognition. Each interface component will be responsible of recognizing a function-specific set of gestures. For example, for the circular dial component, the set of gestures would include grabbing, rotating and dropping.

We model a gesture based on signatures in the local region surrounding an interface component. We define a visual cue loosely as any salient event in the spatio-temporal video stream: for example, motion, color-change, or shape. Thus, the spatio-temporal signature of a gesture will present itself as a sequence of visual cues.

For example, consider a standard push-button with a known interface component mapping for single color cameras. The button-pushing process can be divided into three stages. First, the user enters the local region, which can be detected using background modeling and skin segmentation. Then the user's finger will touch the button and stay on it for a short period of time to trigger it. Last, the user retracts his or her finger from the button, which is accompanied by change of appearance around the button. More powerful and sophisticated parsing models are plausible under this paradigm: an example showing the use of a Hidden Markov Model is presented in Section 4.2.

### **The VICs Interface Component: VICon**

We use the standard Model-View-Controller(MVC) design pattern [44] to define the VICs-based interface component. In MVC paradigm, the model contains all logic of the application, the controller is the part of the interface with which the user can interact, and the view of the interface is how it presents the current application state to the user.

In VICs, the model contains the visual processing engine and all associated internal

logic. The view of a VICON mainly deals with how the VICON displays itself, as well as how to provide all feedback reinforcement during an interaction session. The controller consists of the set of gestures to which the VICON will respond.

Besides these three components, the VICON also contains a set of application hooks which facilitate communication between the VICON, the application, and the system. These utility modules will ensure that VICONs model can be implemented for both conventional 2D interfaces and future 2D/3D interfaces. In the next section we present our system implementation of the interaction and architectural models.

### 4.1.3 VICONs Implementation

We implement VICONs in a stratified fashion to divide the whole system into relatively independent functional modules, thus easing the integration of VICONs into existing and future HCI systems.

#### System Framework

The system structure has a stratified design to partition component responsibilities as shown in Figure 4.5. There are four layers to the system from top down:

1. **Application Layer:** which contains all application logic.
2. **Message Processing Layer:** which is responsible for transferring messages between the input/output systems and the application.
3. **Input/Output Layer:** which contains the vision processing, conventional input data passage (e.g. mice, keyboards) and the graphical rendering.
4. **System Layer:** which is an abstraction of the underlying physical computer system. It generates signals corresponding to mouse-clicks and key-presses, as well as acquires the video signals that are used in the VICONs processing.

In such an architecture, the system layer sends both conventional signals as well as videos to the Input/Output layer. VICONs will process these videos to extract application-

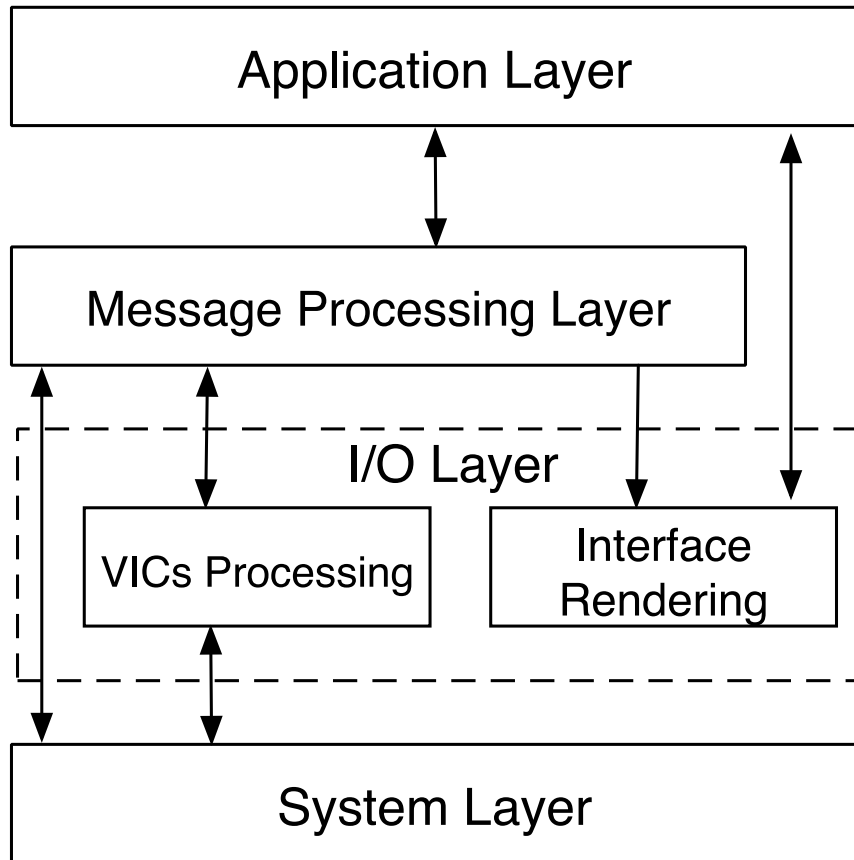


Figure 4.5: VICs system architecture and data flow.

specific gestures for interface components. These output and standard messages are fed into the message processing layer, which dispatch them into appropriate applications.

### **Integrating Current 2D Interfaces Into VICs**

The VICs paradigm can be seamlessly integrated into traditional 2D applications without any modification to the existing application code. The basic idea is to complement the current set of user-events, which are mouse and keyboard generated, with a standard set of hand gestures.

In general, there are two kinds of 2D components that will be captured: clickable items and grabbable items. For clickable components, we define a natural pressing gesture

that is performed by the user extending an outstretched finger near the interface component, touching the element and then removing the finger. A “button-press” event is generated when the visual processor observes the finger touching the element, and then a “button-release” event when the finger retracts.

For draggable items, a natural gesture is again used: the user approaches the item with two opposing fingers open, and upon reaching the item, he or she closes the fingers as if grasping the item. Then, the user is permitted to drag the item around the workspace, and whenever he or she wishes to drop the item, the two grasping fingers are quickly released. The corresponding events we generate are “button-press”, “motion-notify”, and “button-release”. These events are passed to the application through the message processing layer as shown in Figure 4.5.

To wrap an application, we analyze its component structure. In typical 2D windowing libraries, the interface components are stored in a hierarchical manner. The hierarchical representation allows an efficient analysis of arbitrary windowing applications. For each component, we check the events for which it accepts (e.g. mouse clicks). If any of these events match those discussed above for clickable and draggable components, then we construct a VICon at the same location and with the same size as the original component. The VICon is created with a transparent visual representation to avoid modifying the application’s visual representation.

#### 4.1.4 Modes of Interaction

In this section, we describe the set of interaction modes in which a VICon may be used.

1. **2D-2D Projection** - One camera is pointed at a workspace and one or many projectors are used to project interface components onto this surface while the video-stream is processed. This mode has been proposed in [162]. We feel incorporating VICs-based interface components will increase its effectiveness and broaden the domain of applications.

2. **2D-2D Mirror** - One camera is aimed directly at the user and the image stream is displayed in the background of the user-interface for the user. Interface components are then composited into the video stream and presented to the user. This interface mode could also be used in a projection style display to allow for a group to collaborate in the shared space. Figure 4.6 shows some example applications of this model.
3. **3D-2D Projection** - This mode is similar to 2D-2D Projection except that two or more cameras will be aimed at the workspace and the set of possible interaction modes is increased to include more robust 3D geometry. The 4D-Touchpad is an experimental platform based on the VICs framework. We will discuss it in detail in Chapter 5.
4. **2.5D Augmented Reality** - Both video-see-through and optical-see-through augmented reality are possible if the user(s) wear a stereo HMD. With stereo cameras mounted atop the HMD, knowledge of a governing surface can be extracted from the view, e.g. planar surface [27]. All VICons can then be defined to rest on this governing surface and interaction is defined with respect to this surface. One possible application is a piano where each key is a separate VICon.
5. **3D Augmented Reality** - In this case, we remove the constraint that the interface is tied to one governing surface and allow the VICs to be fully 3D. Two example applications areas are (1) motor-function training for young children and (2) medical applications.

#### 4.1.5 Robustness and Applicability

In this section, we presented a general framework for vision-based interaction that provides a set of efficient techniques to employ video as input to computer systems without requiring the user to learn complex interaction semantics. We argue that given a finite set of interaction primitives, the site-centric parsing will perform well. Under the site-centric model, each interface object is *trained* on a prior set of activation patterns through its processor. These patterns are built with well-understood image processing techniques and operate on a small subset of the image. In Section 4.2 and future chapters (see Chap-



Figure 4.6: (left) A VICs-based calculator using a motion-color processor. (middle) Gesture-based demonstration of multiple interaction triggers for a single VICon. (right) VICs-based 2D-2D mirror mode interface for a *Breakout*<sup>TM</sup> style game.

ter 5, Chapter 6), we provide experimental results that support our claim of the scheme’s robustness.

Also, our framework is based on the principle of duplex learning between the user and the interface. In the remainder of this dissertation, we will investigate many learning techniques such as HMMs, neural networks and Bayesian classifiers to learn the gestures based on local image cues. In Chapter 6, we also discuss a human factors experiment on our experimental platform for VICs. This experiment allows such a duplex learning between the user and our system. The experimental results show that our paradigm enables natural and effective interaction for a large group of users.

Another advantage of VICs is that it allows cooperation among multiple users on the same interface. Our approach does not make any assumption of the number of users, nor does it globally model or track the users. Therefore, multiple users can work together and control the same interface to carry out complex tasks. In contrary, traditional tracker-based interface has to track multiple users and handle the situation of user entering and leaving.

The VICs model can immediately be employed to provide fully-functional standard user interfaces by mimicking their components (buttons, scrollbars, etc). We will present our real-time implementation of such systems in the next chapter (Chapter 5). However, it is not universally applicable for there may be some interfaces perfectly suited to user-centric interaction models. For example, eye gaze tracking may be the only way to enable

interaction during automobile driving.

## 4.2 A Stochastic Extension of the Parser for 2D VICs

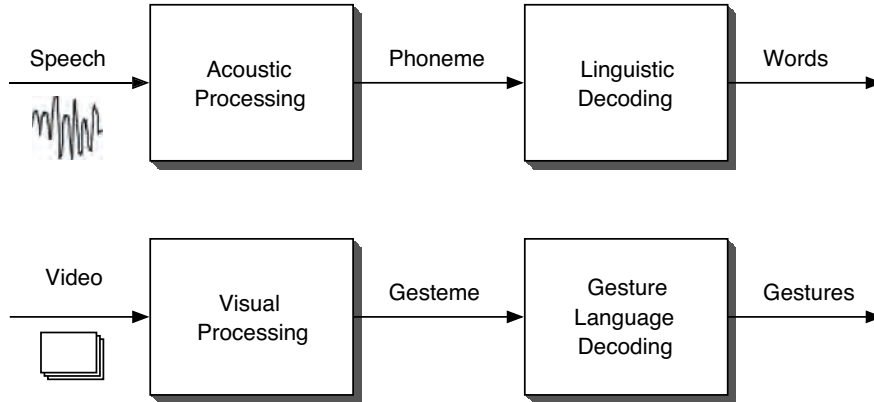


Figure 4.7: The parallels between a speech processing system and visual “language” processing.

As noted at the outset, one of the central problems in vision-based interaction is to effectively manage the complexity of the input space. As a first step, VICs makes use of geometrically localized interaction and state-based activation to minimize both input complexity and computational complexity. As a next step, we are abstracting the visual input space using ideas originally developed in the area of speech processing (Figure 4.7).

In a speech system, interpretation of the incoming waveform is performed in two steps. First, an acoustic pre-processor turns the continuous input waveform into a relatively small set of discrete “symbols” (phonemes). Subsequently, temporal sequences of phonemes are processed into words using a learned language model. In our proposed vision processor, we also abstract the incoming high-dimensional image sequence into a lower-dimensional discrete input string, now representing some aspect of spatial appearance, and likewise train temporal models on these “gestemes” to recognize complete gestural cues. In Chapter 6, we use Gesture Word or GWord to refer to basic gesture elements and build a high-level language model to model the constraints among GWords.

In the remainder of this section, we describe the results of an initial prototype

VICs-based interaction system to identify a button-pushing action. We use a static camera to supervise a virtual button, which is represented by a graphical icon. The user is expected to move his finger toward the button and stay on the button for a short period of time to trigger it. The system will decide if the user has triggered the button. Thus, fast and robust foreground segmentation and action recognition are two key elements of our system. We first discuss Hidden Markov Models in Section 4.2.1. Then we present our implementation of hand segmentation in Section 4.2.2 and HMM-based activity recognition in Section 4.2.3. Finally we present the experimental results in Section 4.2.4.

### 4.2.1 Hidden Markov Models

The Hidden Markov Model (HMM) is a probabilistic graphical model [64, 87, 100, 119] that represents the state of the system as a discrete hidden variable and an observable variable. We limit our discussion to “first-order” HMMs with discrete observation. In such an HMM, the current state of the system only depends on the previous state.

#### Elements of an HMM

An HMM is a five-tuple  $\langle N, M, A, B, \pi \rangle$  where:

1.  $N$  is the number of states in the model. Although the states are not directly observed, for many practical scenarios there is often some physical significance attached to the states or to sets of states. We represent individual states as  $S = \{S_1, S_2, \dots, S_N\}$  and the state at time  $t$  as  $q_t$ .
2.  $M$  indicates the number of distinct observation symbols, i.e., the size of the discrete alphabet. The observation symbols represent the physical output of the system. We denote the vocabulary as  $V = \{v_1, v_2, \dots, v_M\}$ .
3.  $A = \{a_{ij}\}$  represents the state transition probability as  $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$ .
4.  $B = \{b_j(k)\}$  model the each state’s observation probability.  $b_j(k) = P(v_k | q_t = S_j)$ .
5.  $\pi = \{\pi_i\}$  is the initial state distribution,  $\pi_i = P(q_1 = S_i)$ .

For convenience, we represent the parameter set of an HMM model using  $\lambda = \{A, B, \pi\}$ .

### Three Basic Problems for HMMs

Given an representation of an HMM, there are three basic problems to be solved for the model to be useful:

1. **Evaluation Problem:** Given observation sequence  $O = O_1O_2 \dots O_T$  and model parameter  $\lambda$ , compute  $P(O|\lambda)$ . The forward-backward procedure can compute the overall probability in  $O(N^2T)$ .
2. **Decoding Problem:** Given  $O$  and  $\lambda$ , calculate the state sequence  $Q = q_1q_2 \dots q_T$  that is optimal in some meaningful sense. The Viterbi algorithm efficiently finds the sequence with maximal probability  $P(Q|O, \lambda)$ .
3. **Learning Problem:** Given a set of observation sequences, adjust parameters  $\lambda$  to maximize  $P(O|\lambda)$ . The well-known Baum-Welch method is a specialization of Expectation-Maximization [9] algorithm that iteratively refines the model parameter based on observed sequences.

The attractiveness of the HMM include its sound mathematical basis and efficient algorithms, the capability to learn and adapt to training data, and relative robustness against variances of the input. It has gained huge popularity and achieved excellent performance in such areas as speech recognition [101] and visual modeling of dynamic activities [149]. In this dissertation, we use HMMs to model both low-level dynamic gestures as well as high-level gesture sentences (Chapter 6).

#### 4.2.2 Background Modeling and Image Segmentation Based on Hue Histograms

Background subtraction, gray-scale background modeling [61], color appearance modeling [124], color histograms [68] and combination of multiple cues [141] are among the most widely used methods to model the background and perform foreground segmentation.



Figure 4.8: An example of background image, unsegmented image and segmentation result. The leftmost image is the background. The second image shows when the hand has entered the scene. The final segmentation result is shown in the third image.

We propose to use a hue histogram for its computational efficiency and relative color invariance. Hue is a good color-invariant model that is relatively invariant to translation and rotation about the viewing axis, and changes slowly under change of angle of view, scale and occlusion [48, 49].

We assume that the background is known for a given session. We split the background image into an array of non-overlapping equal-sized sub-images. For each sub-image, we build a hue histogram to model it. We process the foreground image in a similar way and perform pairwise histogram matching between background and foreground image histograms. Here, we employ histogram intersection [124] as the comparison criterion.

$$H(F, B) = \frac{\sum_{j=1}^n \min(F_j, B_j)}{\sum_{j=1}^n B_j} \quad (4.2.1)$$

Here  $B$  and  $F$  refer to the background and foreground histogram respectively. If the matching value is below the threshold, which is determined empirically, the corresponding sub-image is classified as foreground; otherwise, it is background. Our experiments show that combining a hue color model with histogram intersection can achieve relative invariance to illumination changes and obtain good segmentation results. Figure 4.8 shows an example.

One of the limitation of this approach is that we have to empirically select a threshold. In our current experiment, this threshold is set to be very low, usually around 0.005, because the difference of the color appearances between the background and the

user’s hand is large. We also carry out a median filter on the segmentation map to reduce segmentation errors caused by imaging noises and window-based matching.

### 4.2.3 HMM-based Human Activity Recognition

In our experiment, we use HMMs [64, 100, 149] to train and recognize the button-pushing action. The basic idea is to define a discrete feature space onto which the image is mapped. Given any captured image sequence, we convert it to an array of discrete features. A discrete forward HMM is constructed for each class of actions and trained based on training sequences using the Baum-Welch algorithm. The probability that each HMM generates the given feature sequence is the criterion of recognition.

#### Feature Extraction

We propose a computationally efficient and robust feature extraction scheme. The extracted feature indicates the direction and distance of the finger from the center of the button. In principle, we split the local activation region of the button into a  $5 \times 5$  grid. Based on the foreground segmentation result, we can determine whether a certain cell is covered by the hand by thresholding the percentage of the foreground pixels in the cell. The direction of the hand with respect to the button is then decided by comparing the number of cells touched by the hand in each of the four directions. The distance of the finger with respect to the button is calculated as the distance between the center of the button and the nearest cell covered by the hand. Combination of all discretized directions and distances forms our feature space. Figure 4.9 shows the definition of our feature space.

#### HMM Structures

We experiment with two different HMM structures to model the dynamics of the gestures. There are four different gestures, each of them modeling triggering the button from a distinct approaching direction: i.e. left, right, up and down. The first HMM paradigm is the singleton-based composite model. For each feature state, we define a basic singleton HMM (Figure 4.10) to represent the dynamics of the finger movement for this particular spatial configuration. To capture the temporal dynamics of the whole gesture, we learn a

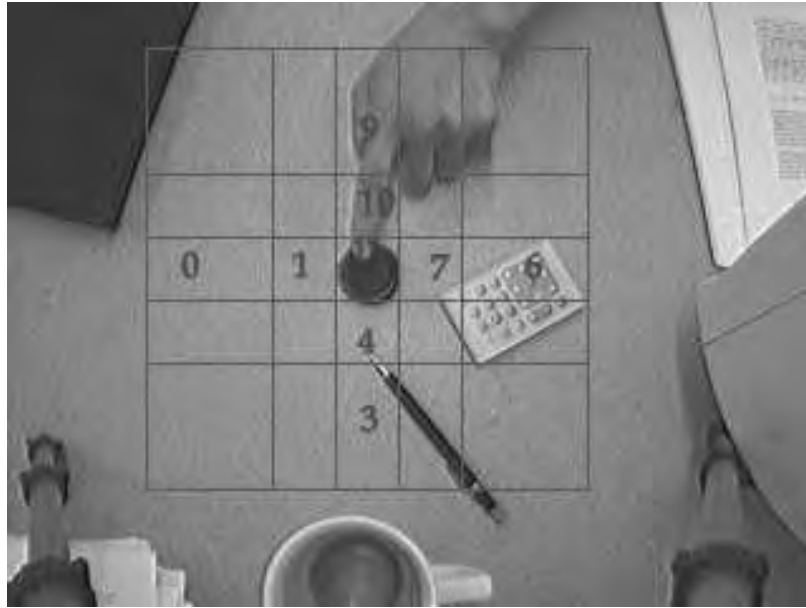


Figure 4.9: The definition of feature space for gesture recognition.

representative sequence for each of the four classes of actions. Based on this characteristic sequence, we build the HMM for this class by concatenating, with null transitions, all the basic singleton HMMs corresponding to each symbol in the sequence. Figure 4.10 shows the structure of this singleton-based HMM structure. In essence, this topology models the gesture as a tour through a series of spatial configurations.

The second model is a three state forward HMM as shown in Figure 4.11. Each class shares the same topology, with the parameter set trained separately based on the

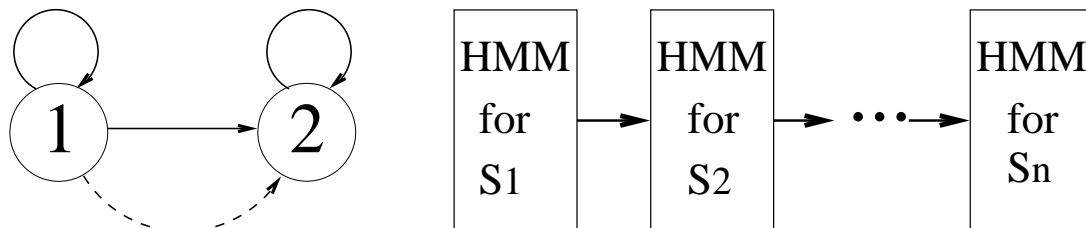


Figure 4.10: HMM structures used to model dynamic gestures. The left figure shows the structure of a basic singleton HMM. The right figure illustrates the composite HMM topology that concatenates singleton HMMs.

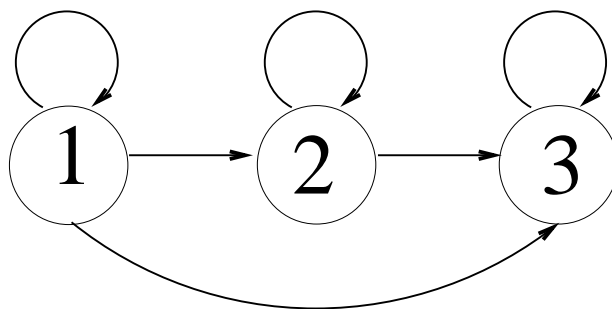


Figure 4.11: The topology of a simple three-state HMM for gesture modeling.

sequences belonging to the corresponding class. The attractiveness of this topology is that it intuitively models the temporal dynamics of the gesture, as opposed to the singleton-based HMMs that try to model each of the symbols in the observation dictionary. In our experiment, for each of the four classes, we expect the user to move the finger through the feature space in the following order: outer layer, inner layer and center of the button. Intuitively, the dynamics is composed of three distinct stages. Ideally, each of the three states of the forward HMM would model the corresponding stage of the dynamics. The parameter sets of the trained class HMMs verify our expectation. For example, if the recognizer observes a sequence in which the hand stays on the button all the time but does not witness the process showing the hand approaching the button, it will reject the sequence as a valid trigger action.

Since it is difficult to capture all possible patterns of non-pushing actions, we use a threshold on the highest possibility of the classes to perform rejection. However, the duration of the action and the possibility that each class generates such a sequence may vary significantly even though the action pattern is still the same. To overcome this time variation, we perform sequence aligning in training and recognition. That is, we choose a fixed length to be the standard length (for example, 20 frames). We perform sampling or interpolation for longer or shorter sequences to fit our standard.

#### 4.2.4 Experimental Results

In our experiments, we use a color camera with an image size of  $640 \times 480$  as the imaging sensor on a standard Pentium III 1Ghz PC running Linux.

##### **Background Modeling and Segmentation Result**

To test our segmentation scheme, we captured image pairs of the background and foreground. By comparing the segmentation result and the ground-truth classification image, which is generated by manually marking the foreground part of the scene, we are able to evaluate this algorithm. We captured more than 20 pairs of background/foreground images with different background scenes and carried out the experiment on these images. The test set also includes 6 pairs of images that undergo illumination changes. As a result, the average correct ratio based on per pixel basis is 98.16%, with average false positive ratio of 1.55% and false negative ratio of 0.29%. Figure 4.8 shows a typical segmentation example. These results show that our histogram-based approach can efficiently segment the foreground.

As shown in Figure 4.12, we also compare the segmentation result with different sub-window sizes and with different numbers of hue histogram bins. The result shows that histograms with at least 8 bins perform better than those with less, while increasing the bins to 16 or more does not bring much performance enhancement for our experimental set. It can be foreseen that more bins will be needed to carry out good segmentation for those images in which the appearance of the foreground is close to that of the background. The result also shows that the false positive ratio will decrease with the increment of tile size because imaging noise would be compressed when the histogram is constructed over a larger neighboring area, which in essence is a process of averaging the appearance of the tile.

Our approach is based on previous research on histogram-based image modeling [124, 68]. Since we only use the Hue chromacity channel, our approach is computationally efficient and relatively invariant to the change of the intensity of lighting. This simplicity also limits the application of our approach to those situations where the foreground and

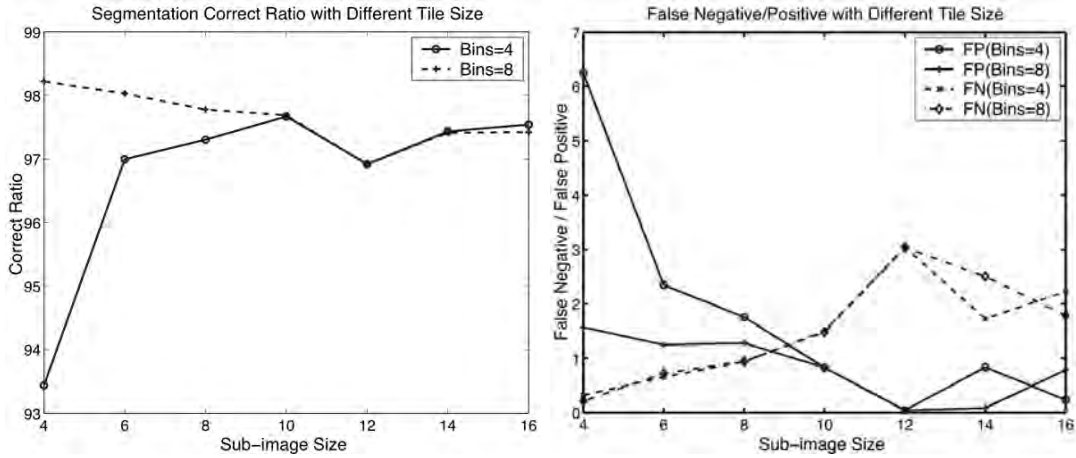


Figure 4.12: Segmentation correct ratio/false positive/false negative with different sub-image size and number of bins in the histogram.

background have similar hue distribution but different saturation.

#### 4.2.5 Action Recognition Result

For training and testing of our HMMs, we recorded over 300 action sequences performed by 6 different people, with 76 of the sequences being used for training.

For the singleton-based model, an offline procedure is carried out to find the best characteristic sequence for each class and construct the HMM. This is essentially a model selection problem. Basically, for each class, we carry out a brute-force search over the whole training sequences of that class. We compare the overall recognition accuracy of using each sequence as the characteristic sequence. The the sequence with greatest overall training accuracy is chosen as the final representative of the class.

For both the singleton-based model and the simple 3-state forward HMM, we carry out the training and testing. Both models achieve correct ratio of 100% on the training set. The testing set contains over 270 sequences. The length of the test sequences varies significantly, ranging from 30 to over 200. The test set includes some sequences with limited illumination changes. To test the rejection scheme, we also include invalid sequences in the test set, such as those that do not conform to the temporal pattern of the model gesture and those that do not follow the correct order of expected visual features of the models.

| Length | Training Accuracy | Testing Accuracy |
|--------|-------------------|------------------|
| 10     | 100.0%            | 86.8%            |
| 20     | 100.0%            | 94.2%            |
| 30     | 100.0%            | 96.8%            |

Table 4.1: Experiment results with different length of characteristic sequence.

The resulting correct ratio is 96.8% for the singleton-based HMM model and 96.0% for 3-state simple HMM. This suggests that a simple 3-state HMM can model the button-pushing gesture almost as good as a 60-state (standard length(=30)  $\times$  2) singleton-based composite HMM. Another advantage for 3-state HMM is its computational efficiency.

For the singleton-based model, the standard length of the class characteristic sequence will influence the system performance and speed. Along with the increase of the size of primary sequence, the time needed to carry out the recognition will also grow linearly. However, since a longer sequence contains more information and thus, has a larger HMM, the total system performance will improve. Table 4.1 shows the experimental results with differing standard lengths.

For the simple  $n$ -state forward HMM structure, we also carry out experiments with different number of states in the topology of the HMM. We find that on our current dataset, an increase in the number of states from three to four or five does not bring much accuracy improvement for our simple 3-stage gesture.

### 4.3 Conclusions

In this chapter, we have introduced the VICs approach to vision-based interaction. VICs stems from our experience using locally activated iconic cues to develop simple vision-driven interfaces. Since component mapping reduces the computational space to local image regions, VICs essentially adds an extra structure to the difficult problem of visually modeling human activities. VICs also extends traditional GUI and interaction techniques by allowing a much richer set of intuitive interaction modalities.

Following the VICs framework, we have developed a 2D-2D application where

two problem are discussed to enable natural gesture-based interaction. The first problem is to develop an efficient way to carry out foreground-background disambiguation. The second problem is how to incorporate dynamics into gestures. We have shown that, given good solutions to the former problem, the latter can be addressed using standard HMM techniques.

In the next chapter, we will concentrate on a recent implementation of the VICs paradigm, the 4D-Touchpad platform. We also propose efficient motion capturing and gesture modeling techniques.

## Chapter 5

# Efficient Capture of 3D Gesture Motion

In the previous chapter, we presented our general VICs paradigm and its various modes of interaction. We also discussed a specific stochastic parser to model 2D hand gestures. In this chapter, we first present 4D Touchpad(4DT) in Section 5.1, a VICs system based on 3D-2D projection. We explain the design and implementation of 4DT in detail, including geometric calibration, chromatic calibration and hand segmentation. Experimental results also show that the 4DT system provides an excellent platform for both visual data collection and gesture-based interaction.

In Section 5.2, we present a novel and efficient scheme to capture the shape and motion of 3D gestures using 4DT platform and VICs concept, i.e., site-centered computation. We efficiently compute the 3D appearance using a region-based coarse stereo matching algorithm in a volume around the hand. The motion cue is captured via differentiating the appearance feature. An unsupervised learning scheme is carried out to capture the cluster structure of these feature-volumes. Then, the image sequence of a gesture is converted to a series of symbols that indicate the cluster identities of each image pair. In Section 5.3, we propose two schemes (forward HMMs and neural networks) to model the spatio-temporal features of dynamics gestures. We test this scheme on a database of over 600 gesture se-

---

<sup>1</sup>Parts of this chapter are joint work with J. Corso.

quences and achieves recognition accuracy of over 96% using both the proposed appearance and the motion cues.

## 5.1 4D Touchad: A VICs Platform

In this section, we present a VICs platform that has been constructed based on the 3D-2D projection interaction mode [22, 28]. A pair of wide-baseline cameras are directed at the interaction surface, which in our current implementation is a standard flat-panel display that is laid atop the table and used as the interaction surface. These setups are shown in Figure 5.1.

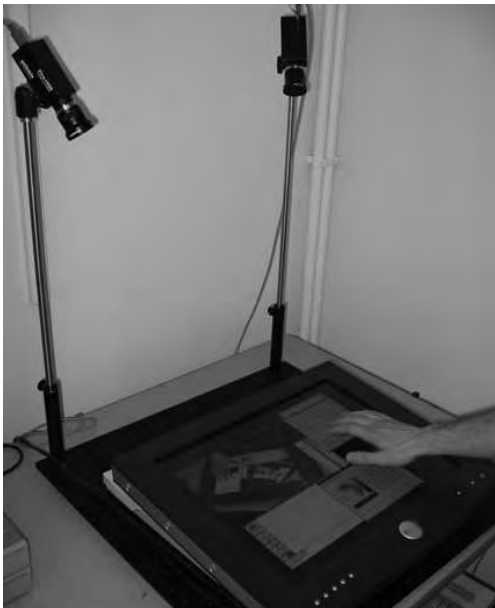


Figure 5.1: 4DT system with a standard flat-panel display.

### 5.1.1 Camera Calibration and Rectification

For applications that deal with 3D gestures and potential 3D interface components, it is necessary that the system is capable of reconstructing the 3D structure of the scene when needed. We carry out a full camera calibration using Bouguet’s calibration

calibration toolbox for Matlab [17]. The core algorithm of this toolbox is based on methods by Zhang [161] and Heikkila et al. [57]. To ease the problem of correspondence and reconstruction, the calibration toolbox also carries out image rectification [40, 41, 76, 127].

We can analyze the system’s spatial resolution based on the geometric parameters of the setup. In our current system as shown in Figure 5.1, we use two cameras with focal length of  $f = 4.2mm$ . The length of the baseline between the two cameras is about  $b = 0.6m$ . Let  $z$  denote the  $Z$ -coordinate of a point  $P$  in the system of the left camera. As discussed in [127], the disparity  $d$  can be computed as  $d = \frac{bf}{z}$ . Thus we have:

$$\frac{\partial z}{\partial d} = -\frac{z^2}{bf} \tag{5.1.1}$$

In our system, the pixel size is  $s = 0.0056mm$ . We can compute the spatial resolution measured by image pixel as  $s\frac{\partial z}{\partial d}$ . Since the distance ( $z$ ) of the display surface is between  $0.80m$  and  $0.88m$ , and users carry out gestures above the display, the lowest spatial resolution happens on the display, which can be computed as between  $1.72mm/pixel$  and  $1.42mm/pixel$ .

### 5.1.2 Geometric Calibration Between the Display and Images

As discussed in Section 4.1, one of the basic concepts of VICs is component mapping to facilitate site-centered computation. For 4DT, we need to compute a mapping between image planes and the monitor plane on which components are rendered. Furthermore, in our current experiments, we assume that the background is known<sup>1</sup>. Thus, if we can geometrically and chromatically model the background of interaction scene (here is the 2D display), we can efficiently segment the user’s hands through image differencing.

We use the well-known homography that maps points on a plane (here the display surface) to their image [40]. We use perspective projection to model the camera projection. The projection  $[wx \ wy \ w]^T \in \mathcal{P}^2$  of a point  $[X \ Y \ Z]^T$  in space into the camera-frame can be modeled with standard perspective projection:

---

<sup>1</sup>In our current 4DT setup this is trivial since the background is precisely what is rendered on the screen at any given moment.

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (5.1.2)$$

where  $m_1 \dots m_{12}$  form the entries of the projection matrix, which has 11 degrees-of-freedom.

Without loss of generality, we can say that the plane lies at  $Z = 0$  yielding the following homography,  $\mathcal{P}^2 \rightarrow \mathcal{P}^2$ :

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_4 \\ m_5 & m_6 & m_8 \\ m_9 & m_{10} & m_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (5.1.3)$$

We use a point-based techniques similar to [161] to recover this homography between the surface plane and images. Here we apply the homography directly to the imaged points. Since there is a five-parameter linear transformation  $\mathcal{P}^2 \rightarrow \mathcal{P}^2$  that maps the camera-frame to the image-plane, we capture them directly in the computed set of homographies.

The basic calibration step is like this: we display a model image on the surface with marked keystone points  $b_{j \in [1 \dots n]}$ . One such pattern is shown in Figure 5.2. Based on their corresponding position on the  $i$ -th image  $q_j^i$ , we define the homography  $H_i$  for the  $i$ -th camera as:

$$b_j = H_i q_j^i, i \in \{1, 2\}, j \in \{1 \dots n\} \quad (5.1.4)$$

### 5.1.3 Color Calibration and Foreground Segmentation

Since the interaction take places in the volume directly above the interface and the VICs system is aware of the rendered interface, we can perform background subtraction to segment objects of interest (e.g. gesturing hands) in the interaction volume. Human hands demonstrate distinct appearance characteristics, such as color, hand contour, and geometric properties of the fingers. We carry out segmentation based on the appearance of the hand.



Figure 5.2: Calibrate pattern used in homography computation.

### Background Modeling

As mentioned previously, we assume that the background is known. The hand can be detected by segmenting the foreground from the image. The key is to find an efficient and robust method to model the appearance of the background and the hand. Background subtraction, gray-scale background modeling [61], color appearance modeling [124], color histogram [68] and combining of multiple cues [141] are among the most widely used methods to model the background and perform foreground segmentation.

We propose to directly model the appearance of the background by computing the transform between the physical scene and the images of the scene captured by the cameras. We model the color appearance of the background using an affine model. The color images from the cameras and the rendered scene are represented in YUV format. An affine model represents the transform from the color of the rendered scene, i.e.,  $s = [Y_s, U_s, V_s]^T$ , to that of the camera image  $c = [Y_c, U_c, V_c]^T$ . Using a  $3 \times 3$  matrix  $A$  and a vector  $t$ , we represent

this model using the following equation.

$$c = As + t \tag{5.1.5}$$

The model parameters,  $A$  and  $t$ , are learned via a color calibration procedure. Basically, we generate a set of  $N$  scene patterns of uniform color,  $\mathcal{P} \doteq \{P_1 \dots P_N\}$ . To ensure the accuracy of the modeling over the whole color space, the colors of the set of the calibration patterns occupy as much of the color space as possible. We display each pattern  $P_i$  and capture an image sequence  $S_i$  of the scene. The corresponding image  $C_i$  is computed as the average of all the images in the sequence  $S_i$ . The smoothing process is intended to reduce the imaging noise. For each pair of  $P_i$  and  $C_i$ , we randomly select  $M$  pairs of points from the scene and the images. We construct a linear equation based on these  $N \times M$  correspondences and obtain a least squares solution for the 12 model parameters. Figure 5.3 shows an example of a pattern and its images in the cameras.

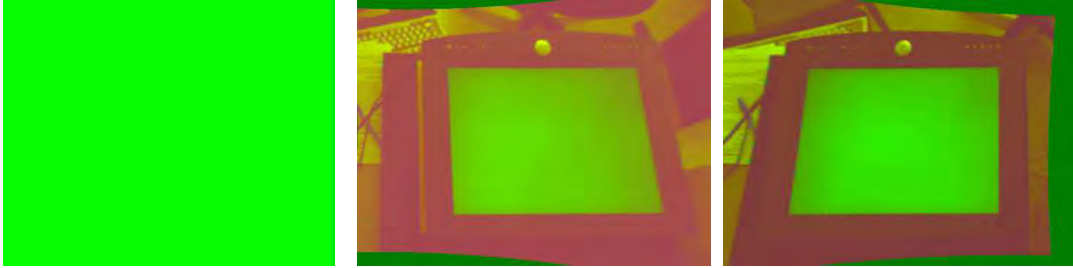


Figure 5.3: (Left) An image pattern for color calibration. (Middle) and (Right) Images of the pattern on the display.

We use image differencing to segment the foreground. Given background image  $\mathbf{I}_B$  and an input image  $\mathbf{I}_F$ , a simple way to segment the foreground is to subtract  $\mathbf{I}_B$  from  $\mathbf{I}_F$ . We compute the sum of absolute differences for the color channels of each pixel. If the difference is above a certain threshold, this pixel is set to foreground. Figure 5.4 shows an example of the segmentation.

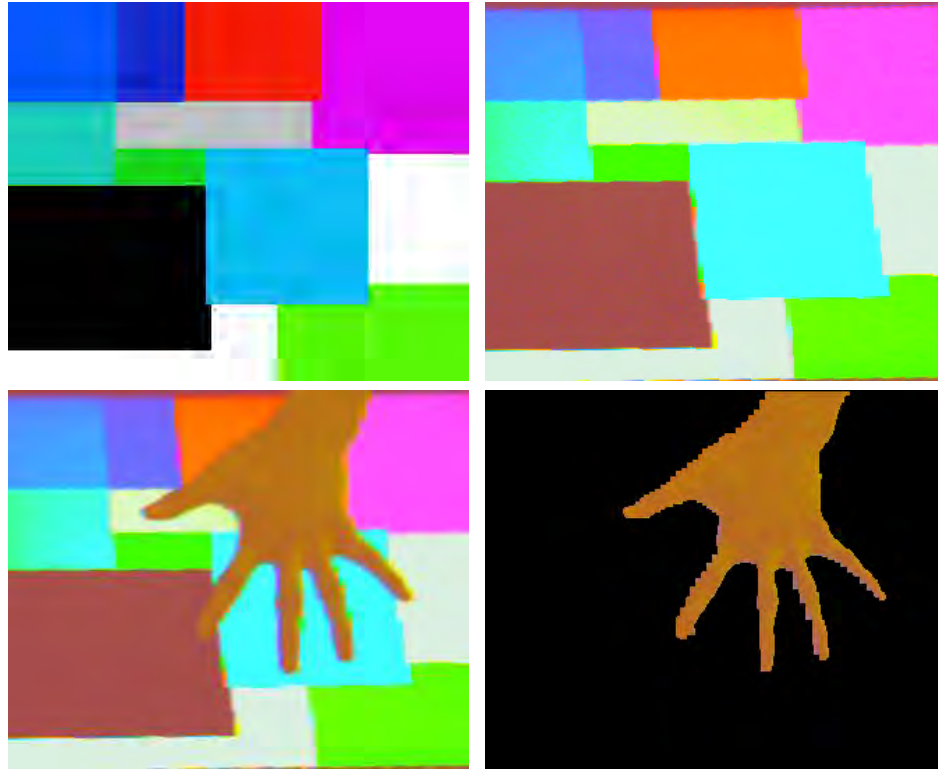


Figure 5.4: An example of image segmentation based on color calibration. The upper left image is the the original pattern rendered on the screen. The upper right image is the geometrically and chromatically transformed image of the rendered pattern. The lower left image shows the image actually captured by the camera. The lower right image is the segmented foreground image.

### Skin Modeling

To improve the robustness of the foreground segmentation, we include an additional skin color model. Many skin models have been proposed in the literature [98]. Here we choose a simple linear model in UV-space. Basically, we collect skin pixels from segmented hand images and train the model as a rectangle in the UV plane. Four parameters, i.e.,  $U_{min}, U_{max}, V_{min}, V_{max}$ , are computed and used to classify image pixels.

After background subtraction and skin filtering, we apply a median filter to the segmentation map to obtain a smooth result.

| Camera | Y    | U    | V    |
|--------|------|------|------|
| Left   | 1.96 | 4.01 | 4.80 |
| Right  | 2.36 | 4.71 | 5.70 |

Table 5.1: Training error of color calibration of each camera.

## Experiments

We have carried out a series of experiments to quantify the accuracy and stability of the system. First, we examine the robustness and stability of the color calibration algorithm. We train the affine color model using 343 unicolor image patterns which are evenly distributed in the RGB color space. We compute the error as the average absolute difference in each color channel between the captured camera images and corresponding transformed images. Table 5.1 shows the average absolute error of each color channel (in YUV space) over all training pixels.

To test the accuracy of the learned affine model, we display over 100 randomly generated color images and examine the resulting segmentation. In this case, the ground truth is an image marked completely as background pixels. For both cameras, the system achieves segmentation accuracy of over 98%.

We also investigate the efficacy of the linear skin model. We learn the model by analyzing image sequences containing the hands of 15 people. These people are from different ethnic groups, including several Asian and African Americans. To test the model on our platform, the user is asked to place his or her hand on the flat-panel and keep it still. Next, we render a background which is known to perform well for skin segmentation and treat the resulting skin foreground segmentation as the ground truth. The user is asked to keep his or her hand steady while we render a sequence of 200 randomly generated patterns. For each image, we count the number of incorrectly segmented pixels against the true segmentation. The overall skin segmentation accuracy is over 93%.

#### 5.1.4 Posture Recognition on 4DT

The 4DT provides an excellent platform for visual capture of hand, as discussed above. The robust and efficient hand segmentation facilitates gesture analysis and recognition. In this section, we discuss a straightforward way to capture hand appearance and use learning-based methods to recognize postures.

##### Posture Recognition Using Neural Network

As summarized in Chapter 2, neural networks [38] are among the most popular methods for pattern recognition. We use a standard three-layer neural network to classify gestures. The three layers are input layer, hidden layer and output layer. The output is usually a nonlinear function of the input. The parameters of the neural network are trained using the standard back-propagation algorithm. In our experiments, the output indicates the posterior probability that the input image sequence is a certain gesture.

We carry out a set of experiments that use the 4D-Touchpad platform. The gestures in these experiments have been integrated into applications that run on the platform and have been tested in a demonstration setting. In the experiments, we have implemented a posture recognition scheme to model pressing and grasping (Figure 5.5) gestures. We use a network with only one output node for each posture. The output is real-value ranging from 0 to 1 interpreted as the network posterior probability.

##### Posture Recognition Results

In our experiments, we use the segmented image data that is available on the 4DT and subsample each local VICON region into a  $16 \times 16$  image resulting in 512 inputs to the network (there is binocular image data). Example images are given in Figure 5.5. The networks both have 20 nodes in the hidden layer. In the results shown in Table 5.2, we distinguish between positive cases and negative cases because such an on/off network can be more susceptible to false-positives. We note the recognition rate for these experiments is nearly 100% for the testing data 5.2.

Next, we carry out an experiment to test the accuracy and spatial sensitivity of

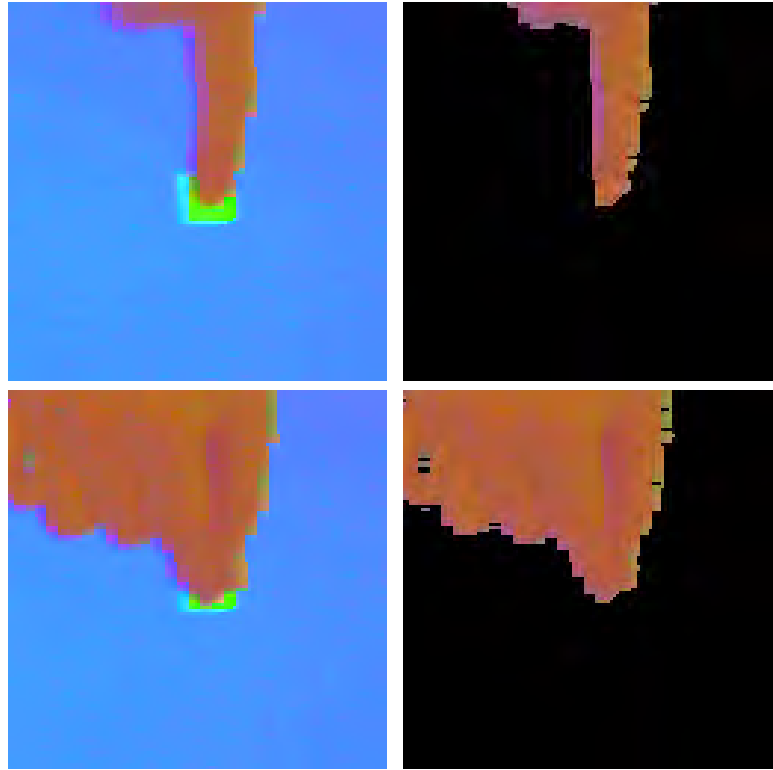


Figure 5.5: (Top)Pressing Gesture. (Bottom)Grasping Gesture. (Left) Original Image. (Right) Segmentation Image.

the button press gesture. We display an array of square buttons, which are adjacent to each other with no buffer-space. Then, the system randomly chooses one of them and instructs the user to press it. We vary the size of the button from  $20 \times 20$  pixels to  $75 \times 75$  pixels and repeat the experiment for each size. Figure 5.6 shows the scene when the size of the button is  $40 \times 40$  pixels; here, we see that the size of the user's finger tip is about 40 pixels wide at the display resolution. Figure 5.7 shows the testing results. In the graph, we see the accuracy of the press recognition rapidly increases with the size the button. We also note that, as expected, for button sizes smaller than the fingertip resolution (about 40 pixels), the accuracy is much worse.

We also incorporate our posture recognition method and VICs scheme into existing GUIs. Figure 5.6 shows a real-time application using gestures to control a calculator program in the X-Windows system. The gesture vocabulary includes a simple button press

| Posture         | Pressing  |           | Grasping |         |
|-----------------|-----------|-----------|----------|---------|
|                 | Training  | Testing   | Training | Testing |
| <b>Positive</b> | 2051/2051 | 2057/2058 | 366/367  | 364/367 |
| <b>Negative</b> | 1342/1342 | 1342/1342 | 480/480  | 483/483 |

Table 5.2: On/Off neural network posture classification for the pressing and grasping gestures.

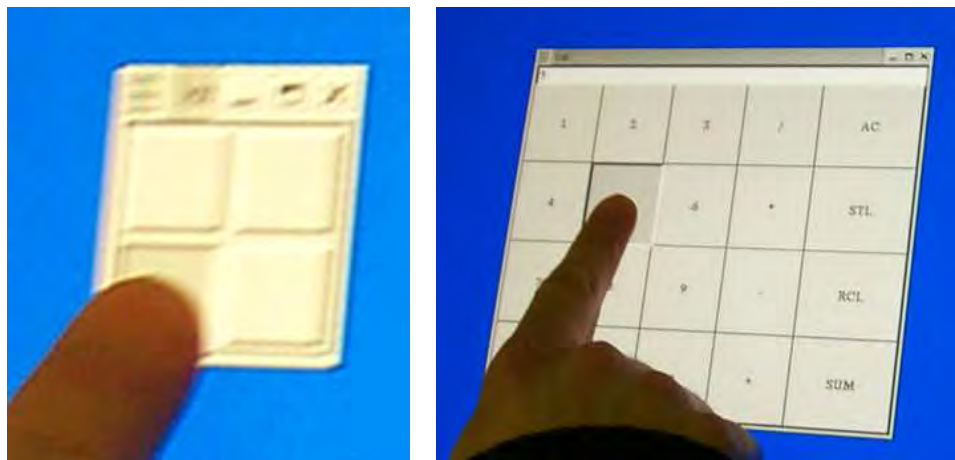


Figure 5.6: Left picture shows the scene of when the finger is trying to trigger a button of  $40 \times 40$  pixels. Right picture shows a user is controlling an X window program using finger press gestures.

gesture and a clear screen gesture, which is represented a flat palm over the center of the program window.

## 2D Pointing Stability

To investigate the stability of the location recognition and the underlying segmentation, we performed an experiment in which the user is asked to point and stay at an arbitrarily specified position on the screen. For this experiment, we have trained a neural network system to localize the spatial coordinates of the fingertip. Thus, for a given image region defined for an interface element, the images are processed by the network which yields a single  $(x, y)$  sub-pixel resolution tip-location. Again, the input to the network is a coarse, sub-sampled image (256 total samples), and the network has 16 hidden nodes.

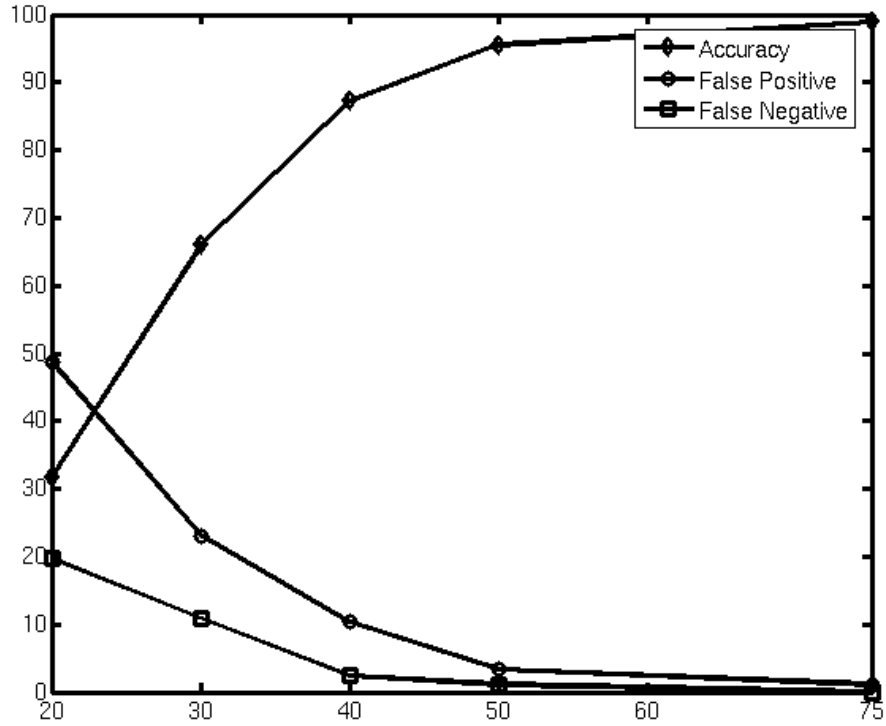


Figure 5.7: Button press experiment result on 4DT

We hand-annotated each training datum by clicking on the fingertip location. Over 600 training samples are used and the average error for the training data is 0.0099 image pixels.

To test the stability, we dynamically change the scene by rendering randomly generated background images and record the output of the pointing gesture recognizer. The stability of the system is measured by the standard deviation of the recorded 2D position. On a standard flatpanel monitor with a resolution of  $1280 \times 1024$  pixels, our system reports an average standard deviation of 0.128 and 0.168 pixel in the  $x$  and  $y$  direction, respectively.

## 5.2 Efficient Motion Capturing of 3D Gestures

As discussed in Section 2.3.4, human motion capture is crucial for gesture analysis. Human hands and arms are highly articulate and deformable objects and hand gestures normally consist of 3D global and local motion of the hands and the arms. Manipulative and interactive gestures [99] have a temporal nature that involve complex changes of hand configurations. The complex spatio-temporal properties of such gestures render the problem too difficult for pure 2D (e.g. template matching) methods. Ideally we would capture the full 3D information of the hands to model the gestures [78]. However, the difficulty and computational complexity of visual 3D localization and robust tracking prompts us to question the necessity of doing so for gesture recognition.

To that end, we present a novel scheme to model and recognize 3D temporal gestures using 3D appearance and motion cues without tracking and explicit localization of the hands. Instead we follow the site-centered computation fashion of VICs. We propose that interaction gestures can be captured in a local neighborhood around the manipulated object based on the fact that the user only initiates manipulative gestures when his or her hands are close enough to the objects.

The advantage of our scheme is that it is efficient and highly flexible. The dimension of the volume of the local neighborhood around the manipulated object can be adjusted conveniently according to the nature of the particular interaction environment and the applicable gestures. For example, in a desktop interaction environment where the interaction elements are represented as small icons on a flat panel and manipulative gestures are only initiated when the user’s hand is near the surface of the panel, we only need to observe a small volume above the panel with the icon sitting at the center of the bottom. The height and diameter of the volume is also limited to be able to capture enough visual cues to carry out successful gesture recognition.

The remainder of this section is structured as follows. In Section 5.2.2 we present a novel method to efficiently capture the 3D spatial information of the gesture without carrying out a full-scale disparity computation. We discuss how to learn the cluster structure of the appearance and motion features via an unsupervised learning process in Section 5.2.3.

### 5.2.1 Related Work

With a model-based approach [2, 95], it is possible to capture a gesture in higher dimensionality than 2D. In [2], a 3D hand model is represented as a set of synthetic images of the hand with different configurations of the fingers under different viewpoints. Image-to-model matching is carried out using Chamfer distance computation. One of the difficulties of this approach is to construct a good 3D model of the hand that can deal with variance between different users. Furthermore, efficient algorithms are necessary to handle the matching between models and input images. Another approach to capture 3D data is to use special cameras [78], such as 3D cameras or other range sensors. However, the hardware requirement limits its application to and make it difficult for general HCI systems.

### 5.2.2 Visual Capturing of 3D Gesture Features

As discussed in Section 2.3.5, visual features are important to the analysis and recognition of dynamic gestures. Good visual features should have strong discriminative capabilities and robustness against variance and noises. These requirements present challenges for the feature space to overcome the variance among image sequences of the same gesture. These differences can be caused by variance of hand configuration and motion, temporal pace, illumination and all the factors introduced by different gesturers.

We propose to achieve robust feature extraction from two perspectives. One way is to extract visual cues that contain rich information of the hand configuration and motion, but discard noise. Our 4DT system provides such a platform. The other way to handle the variance among the visual signals is the statistical approach. In essence, we learn a statistical model of the visual data and represent the visual signal in a statistical fashion.

### 3D Gesture Volume

Given a pair of rectified stereo images of the scene, a disparity map can be computed using a standard correspondence search algorithm. Since we only care about the local neighborhood around the object, we can constrain the stereo search to a limited 3D space around the object. This spatial constraint brings about two advantages: (1) we only care

about the small patch of the image centered at the object, and (2) we only need to search through a limited range of disparities (depths), which is a volume around the depth of the object. To speed up the disparity computation, we further simplify the process using block matching technique.

Formally, let  $\mathbf{I}_l$  and  $\mathbf{I}_r$  be a pair of rectified images of the scene. We split the images into tiles of equal size of  $w \times h$ . Here  $w$  and  $h$  refer to the width and height of the tile, respectively. Suppose we only consider a local area of size of  $m \times n$  patches, starting at patch  $(x_0, y_0)$ . Given a discrete parallax search range of  $[0, (p-1) \times w]$ , we can characterize the scene using a  $m \times n \times p$  volume  $F$  as:

$$F_{x,y,d} = Match(I_{l(x_0+x,y_0+y)}, I_{r(x_0+x+d,y_0+y)}) \quad (5.2.1)$$

$$x \in [0, m-1], y \in [0, n-1], d \in [0, p-1]$$

Note that the image index indicates a patch of the image, not a particular pixel. Different block matching algorithms can be applied, such as sum of squared difference and sum of absolute differences.

Following this scheme, we can extract the features of the image as a vector with the size of  $m \times n \times p$ . The typical size of the extracted appearance vector is from 125 to 1000. Figure 5.8 shows examples of the stereo image pair and the extracted 3D feature of the scene. In this figure, we shows the bottom level ( $d = 0$ ) of the appearance volume. It can be seen that the extracted feature volume characterizes the different configuration of the user's hand with respect to the target object.

### **Motion by Differencing**

Since we represent the 3D appearance of the gesture images using feature vectors, one way to capture the motion information of the gestures is to compute the displacement in this feature space. In Equation 5.2.2, we represent the motion vector as the difference of the appearance vectors between consecutive frames. Figure 5.9 shows an example of the computed motion feature.

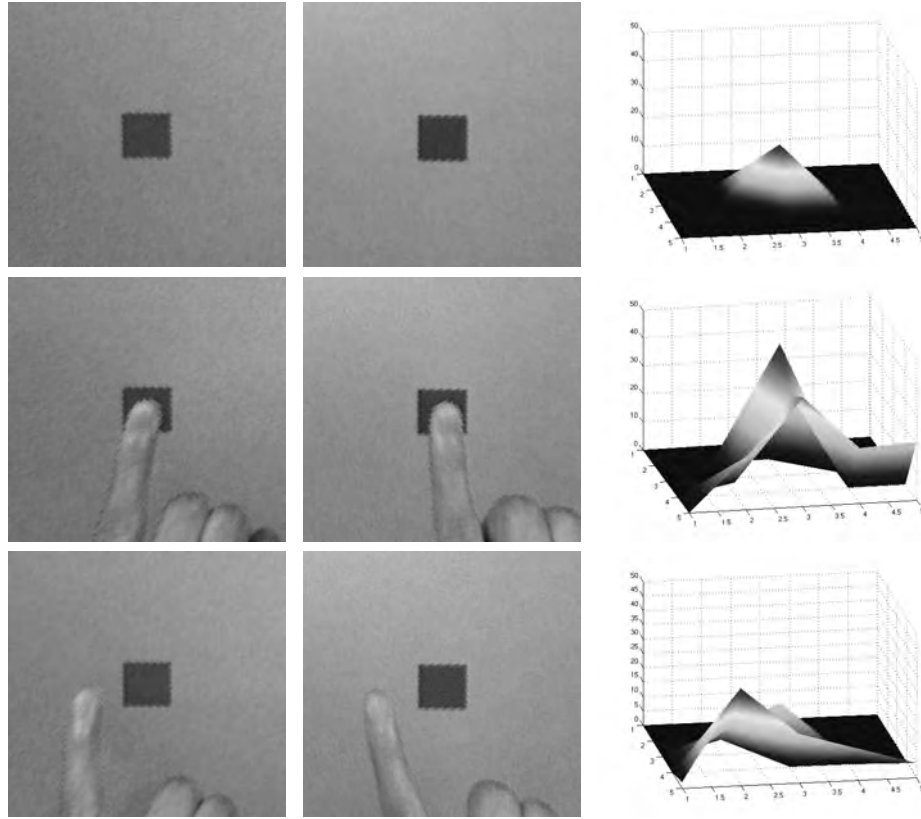


Figure 5.8: Examples of the image pair and extracted appearance feature. Left and middle column display left images and right images of the scene, respectively. Right column shows the bottom layer of the feature volume (i.e.,  $F_{x,y,d}$  with  $d = 0$ ). The z-axis of the feature figure indicates the value of each cell with corresponding x and y indices.

$$Motion_i = F_i - F_{i-1}, i = 2, \dots, M \quad (5.2.2)$$

One way to combine the appearance feature and the motion feature is to concatenate the two vectors to form a larger vector. This new vector contains both the static and temporal information of the gesture.

### 5.2.3 Unsupervised Learning of the Cluster Structures of 3D Features

Give a corpus of training gesture sequences, we obtain a set of feature points in the high-dimensional feature space. We expect a lot of redundancy of information because the training set contains repeated gestures and there are only a limited number of gestures

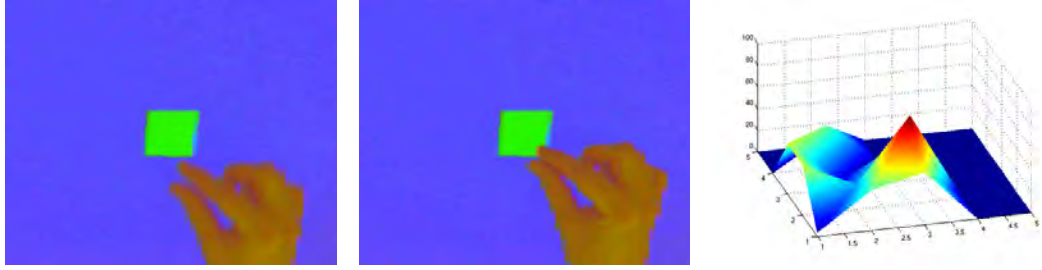


Figure 5.9: An example of the motion vector. Left and middle column display the images of the scene at two different time. Right column shows the bottom layer of the motion volume.

in the set. For example, in our experiment [151] containing six manipulative gestures, we perform Principal Component Analysis(PCA) analysis on the 125-dimension appearance feature that we extracted from over 600 gesture sequences. We define the distortion error as the ratio between the reconstruction distortion and the original feature vector. We can achieve an average reconstruction error of less than 5% using only eight eigenvectors. Therefore, we are able to represent the high-dimensional visual feature of the gestures using data of much lower dimensionality without losing the capability to discriminate between them.

One popular way to model temporal signals is to learn a statistical model [38]. Generally speaking, the size of training data needed for statistical learning normally increases exponentially with the dimensionality of parameter space. This curse of dimensionality is one of the reasons that visual modeling of gestures is difficult. Thus we propose to reduce the dimensionality of the 3D feature by learning its cluster configuration.

We propose an unsupervised learning procedure based on Vector Quantization (VQ) [64] to cluster the raw data. Particularly, we employ K-means clustering to represent each feature  $v_i$  with one of the  $k$  clusters  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$  based on nearest-neighbor criterion.

$$VQ(v_i) = \arg \min_{C_j \in \mathcal{C}} Dist(C_j, v_i) \quad (5.2.3)$$

An iterative algorithm is implemented to learn the  $k$  cluster centroids. We first

initialize the  $k$  clusters randomly. Then we alternately determine nearest neighbors for each cluster and recompute the cluster centroids from those feature points that belong to the corresponding cluster. We evaluate the clustering result based on the average representation error, which is defined as the distance between a feature point  $v_i$  and its corresponding cluster centroid  $VQ(v_i)$ . We run this randomly-initialized K-means algorithms several times and choose the one with the least representation error.

The choice of the number of clusters to initialize the VQ algorithm is a difficult model selection problem. Many approaches have been presented, such as Rival Penalized Competitive Learning algorithm [147] and Competitive and Cooperative Learning approach [26]. We handle this problem based on the analysis of the average representation error. In theory, as the number of clusters  $k$  increases, the average representation error decreases. On the other hand, our aim of feature dimensionality reduction prefers smaller  $k$ . A trade-off is achieved by increasing the number of clusters until the average representation error only decreases slightly as  $k$  grows larger.

Figure 5.10 shows an example of the relationship between  $k$  and representation error. We can see that when the cluster number is around 10, increasing the cluster number can only slightly reduce the average error. Thus, we can select the number of clusters to be about 10.

VQ algorithm also allows the combination of multiple visual cues. Suppose we extract  $C$  different visual features  $\{V^1, V^2, \dots, V^C\}$  from the image frame. One way is to normalize each visual feature to the same scale and then concatenate the feature vectors to form a new feature  $(V^1, V^2, \dots, V^C)$ . The dimensionality of this new feature space is the sum of that of all the individual feature space. Then we can carry out VQ on this new feature space. Another way to combine these different kinds of visual cues is to carry out VQ on each feature space separately. Let  $VQ_i$  denote the VQ projection in the  $i$ -th feature space. The overall representation of the visual features thus can be expressed as a discrete vector  $(VQ_1(V^1), VQ_2(V^2), \dots, VQ_C(V^C))$ . Since we know the number of clusters in each feature space, which is equivalent to the dimensionality of corresponding element of the discrete vector, we can further convert the multi-dimensional vector into a scalar.

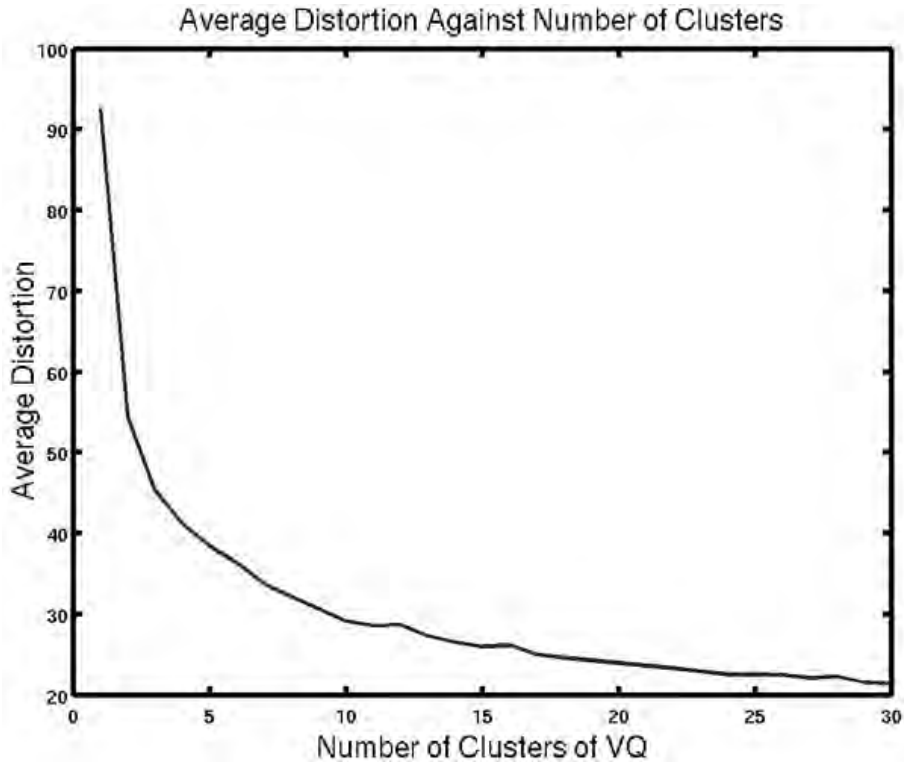


Figure 5.10: The average representation error against number of clusters.

### 5.3 Modeling Dynamic Gesture Based on 3D Feature

To test the efficacy and efficiency of our proposed motion capture and unsupervised learning scheme, we carry out gesture recognition using two approaches, HMMs and neural networks.

#### 5.3.1 Modeling Dynamic Gestures Using HMMs

We use typical left-to-right HMMs to model the dynamics of the temporal gestures. The main reason is that the underlying state sequences of our gesture don't have back transitions. The input to the HMMs is the gesture sequence represented as a series of symbols with each symbol indicating the cluster identity of current frame.

In our experiment, we use a 6-state left-to-right HMM to model each of the six manipulative gestures. Figure 5.11 shows the topology of the HMMs.

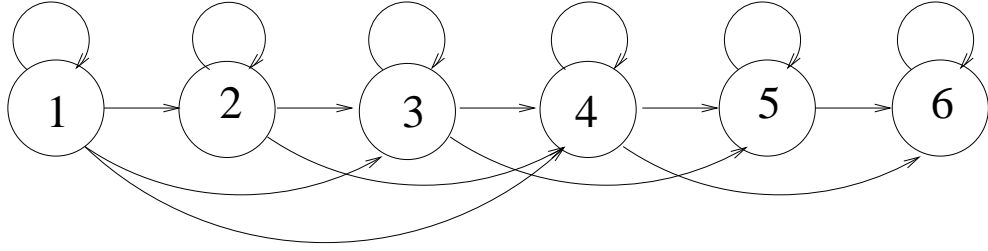


Figure 5.11: HMM structure for modeling dynamic gestures.

The choice of the number of the states in the forward HMM is based on the intuitive analysis of the temporal properties of the gestures to be modeled. In our current experiment, each of the gestures can be decomposed of less than 6 distinct stages. For example, if we use 3 spatial layers to represent the vicinity of an manipulated object, the gesture of swiping an icon to the left can be viewed as such a configuration sequence of the hand: (1) entering the outer layer of the vicinity of the icon, (2) entering the inner layer (3) touching the icon to select it and (4) swiping the icon by moving the finger to the left side of the icon. Ideally, each of the distinct stages can be modeled by a certain state of the forward HMM. The parameter sets of the trained HMMs verify our expectation, in which the observation probability of each symbols of a gesture is dominantly high in one of the states and very small in other states.

Since our training data is limited, we carry out a simple smoothing for the observation probability [64]. Given a smoothing constant  $\alpha$  and the size of the observation vocabulary  $M$ , the observation probability of  $B_{jk}$  is smoothed as:

$$B_{jk} = \frac{B_{jk} + \alpha}{\sum_{i=1}^M B_{ji} + M\alpha} \quad (5.3.1)$$

### 5.3.2 Modeling Dynamic Gestures Using Neural Networks

Another way to learn the gestures is to use multilayer neural networks. The input to the neural network is the whole gesture sequence, which is now a sequence of symbols. The output is the identity of the gesture. To meet the requirement of the neural network, we need to fix the length of each input sequence. We align each sequence to a fixed length

by carrying out sub-sampling on those sequences that are longer than the predefined length and interpolation on those that are shorter. The parameters of the network are also learned from training data using the standard back-propagation algorithm [38].

In our current system, the neural network consists of 3 layers and we have 50 nodes in the hidden layer.

### 5.3.3 Gesture Recognition Experiments

#### Experimental Setup

In our experiments, we collect gesture sequences consisting of 6 interactive gestures, i.e., pushing a button, twisting a dial clockwise, twisting a dial anti-clockwise, toggle a switch, swiping an icon to the left, and swiping an icon to the right.

We implement the system on a PC with dual Pentium III processors. The system achieves real-time speed; the processing is limited by the cameras (30Hz). The system processes the continuous video in the following fashion. For each captured image pair, the appearance and/or motion features are extracted and the corresponding cluster identity of current features is computed based on trained cluster centroids. We define “silence” configuration as a scene where the hand has not entered the vicinity of the target icon. We begin recording a new sequence when a non-silence configuration appears. We carry out the recognition of current sequence and notify the user when a valid gesture is observed. Then we terminate the recording of the current sequence and the system enters a new cycle. Another case for ending current sequence is when the system continuously observes silent configuration.

#### Gesture Recognition Results

To perform training of the HMMs and the neural networks, we record over 100 gesture sequences for each of the 6 gestures. A separate test set contains over 70 sequences of each gesture.

We carry out the training and testing on several feature sets. These different sets are characterized by the dimensionality of our 3D gesture volume described in Section 5.2.2

and different combinations of the appearance and motion cues.

1. **Appearance Only (125-Dimension)**

In this set, we only use the appearance feature with the dimensionality as  $m(= 5) \times n(= 5) \times p(= 5) = 125$  (see Section 5.2.2 for definition of  $m$ ,  $n$ , and  $p$ ). We carry out the K-means on the training set of these features using 8 cluster centers.

2. **Appearance Only (1000-Dimension)**

Similar to the first set. But the dimensionality of the appearance feature is  $10 \times 10 \times 10 = 1000$ .

3. **Motion Only (1000-Dimension)**

We compute the motion feature by taking the difference between two 1000-dimension appearance vectors. We use 15 cluster centers to represent the cluster structure.

4. **Concatenation of Appearance and Motion**

In this set, we concatenate the 125-Dimension appearance feature with the 1000-dimension motion vector to form a 1125-Dimension vector. We carry out the K-means on this appearance-motion feature set using 18 clusters.

5. **Combination of Appearance(125-Dimension) and Motion**

We carry out K-means on the 125-Dimension appearance feature and 1000-Dimension motion features separately. Then each frame is represented as a 2D discrete vector containing both the appearance cluster identity and motion cluster character.

6. **Combination of Appearance(1000-Dimension) and Motion**

Similar to the previous setting except that we use the 1000-Dimension appearance feature.

We perform the training and testing on these sets for the HMM models and the neural network. For the neural network, we align each gesture sequence to the fixed length of 20, as discussed in Section 4.2.3. For the HMM models, we also carry out comparison experiments between using the same aligned sequences as the neural network and applying the raw unaligned sequence. Table 5.3 shows the gesture recognition results for all the

feature sets and both gesture models. For each model we report both the recognition accuracy on the training set and that on the test set.

| Set                | HMM   |       | NN    |      | Unaligned |      | Collapsed |      |
|--------------------|-------|-------|-------|------|-----------|------|-----------|------|
|                    | Train | Test  | Train | Test | Train     | Test | Train     | Test |
| Appearance(125-D)  | 99.5  | 99.5  | 100.0 | 98.8 | 99.4      | 99.4 | 89.3      | 88.8 |
| Appearance(1000-D) | 99.5  | 100.0 | 98.4  | 94.4 | 98.4      | 98.0 | 88.3      | 86.1 |
| Motion(1000-D)     | 98.4  | 98.1  | 97.7  | 86.3 | 97.9      | 98.8 | 98.4      | 96.6 |
| Concatenation      | 98.9  | 99.0  | 98.9  | 87.7 | 96.7      | 96.1 | 90.8      | 89.0 |
| Combination 1      | 100.0 | 100.0 | 100.0 | 96.6 | 98.2      | 97.3 | 94.2      | 96.8 |
| Combination 2      | 99.8  | 99.8  | 99.8  | 97.1 | 99.2      | 99.5 | 99.8      | 98.8 |

Table 5.3: Dynamic gesture recognition results for different feature spaces.

The results show that aligning the sequences to the same length improves the recognition accuracy. Also the motion feature alone performs slightly worse than those with appearance cues. However, combining appearance features with the motion features achieves the best recognition accuracy for our current gesture set.

Another interesting comparison between the HMM model and neural network shows that our multilayer neural network tends to over-train on the feature sets. The neural network model achieves equivalent or higher accuracy on the training set as the HMM model, but perform worse on the test set. During the training of the HMMs, the Baum-Welch algorithm runs for less than 5 iterations before the overall system likelihood reaches a local maximum. While during the neural network training process, the back-propagation algorithm typically runs for over 1000 iterations. We stop the procedure when the decrease of the output error between consecutive runs is lower than a threshold, which is typically a very small number such as 0.00001.

Alternatively, one could stop the back-propagation algorithm interactively by measuring the performance on a validation set after each iteration and halting the training process if the classification on this validation set degenerates. However, we choose a fixed threshold to preserve the generality of the method and keep the training process automatic.

We also compare the gesture modeling using HMM based on the raw sequences and those using collapsed sequences. Each raw sequence containing a gesture is packed in

such a way that we only record a symbol if it is different to its previous one. In essence, we only record the order of the appearance of each feature, excluding the duration in the original temporal sequence. This is similar to the rule-based and state-based gesture modeling [15, 145]. Table 5.3 shows the gesture recognition results based on the datasets of collapsed sequences.

Compared to the results using raw sequences, the gesture recognition using collapsed sequences perform slightly worse. Still, for the combination of the appearance and the motion features, this scheme of gesture modeling based only on key frames achieves very good recognition performance.

## 5.4 Conclusions

In this chapter, we present the design and implement of the 4DT system: a HCI system based on 3D-2D projection of VICs paradigm. With system calibration and skin modeling, 4DT allows efficient motion capture for gesture analysis. Various experiments have been implemented on this system to demonstrate its efficacy and robustness.

We also present a novel approach to efficiently capture the appearance of the hand gesture in a localized space via region-based stereo matching. We reduce the dimensionality of the 3D feature by employing unsupervised learning. These features can be used to model postures and dynamic gestures.

We implement a real-time system based on the 4DT platform and test the system using two different approaches to model the temporal gestures, i.e., forward HMMs and multilayer neural networks. By combining the appearance and motion cues, both HMM models and the neural network achieves an recognition accuracy of over 96%. The proposed scheme is a flexible and efficient way to capture the 3D visual cues in a local neighborhood around the object. The experimental results show that these local appearance and motion features capture the necessary visual cues to recognize different manipulative gestures. Our approach avoids using special 3D sensors to capture the hand shape [78]. Compared to 2D appearance-based methods, as discussed in Section 2.3.3, our scheme has much stronger capability of handling the shape ambiguity caused by imaging projection.

One disadvantage of our appearance-based is that the shape volume is dependent on the geometric configuration of the cameras. This limits its application to those systems where cameras are fixed.

## Chapter 6

# Robust Modeling of Multimodal Gestures With Heterogeneous Observation

In the previous two chapters, we have presented VICs, which is a novel paradigm for human-computer interaction. We have also discussed techniques for efficient hand motion capture and various models to recognize postures and dynamic gestures. These techniques enable us to collect large amount of visual data and to investigate higher-level models to analyze continuous and contextually constrained gesture sequences. In this chapter, we concentrate on modeling multimodal gestures and evaluating our gesture-based interface system based on large-scale experiments.

Traditionally, gesture-based interaction in virtual environments is composed of either static, posture-based gesture primitives or temporally analyzed dynamic primitives. However, it would be ideal to incorporate both static and dynamic gestures to fully utilize the potential of gesture-based interaction. To that end, we propose a probabilistic framework that incorporates both static and dynamic gesture primitives. We call these primitives Gesture Words (GWords). Using a Hidden Markov Model, we integrate these GWords and a high-level language model in a coherent fashion. Composite gestures are represented as

---

<sup>1</sup>Parts of this chapter are jointed work with J. Corso.

stochastic paths through the HMM. A gesture is analyzed by finding the path that maximizes the likelihood on the HMM with respect to the video sequence. To facilitate online computation, we propose a greedy algorithm for performing inference on the HMM. The transition parameters of the HMM can be learned via three different methods: supervised, unsupervised, and hybrid.

To investigate the efficacy of the high-level model for online processing, we carry out two separate experiments. The first experiment consists of ten GWords with six composite gestures which are carried out by two users. The second is a human factor experiment that involves sixteen subjects. There are fourteen GWords and nine composite gestures in the vocabulary. We intend to investigate our model’s efficacy in recognizing gestures carried out by a relatively large group of users. The experimental results show that the HMM can accurately and robustly recognize composite gestures. Furthermore, 88% of the subjects consider our gesture interface to be comparable or more comfortable than traditional GUIs with a mouse.

## 6.1 Introduction

As discussed in Section 2.3.1, the gestures in existing interfaces can be classified into two categories: static hand postures and dynamic gestures. Static postures model the gesture as a single key frame, thus discarding any dynamic characteristics. Dynamic gestures contain both spatial and temporal characteristics. Furthermore, parameterized dynamic gestures incorporate quantitative information about the geometry or dynamics of the motion involved.

It seems clear that to fully harness the representative power of human gestures, static postures and non-parametric and parametric, dynamic gestures must be integrated into a single coherent gesture model. For example, visual modeling of ASL is still limited by the lack of capabilities to handle the composite nature of gestures. To that end, we present a novel framework that integrates static postures, unparameterized dynamic gestures and dynamic parameterized gestures into a coherent model.

In this framework, an HMM is used to model the semantics and temporal patterns

of different parts of a complex gesture; essentially, the HMM model is a high-level language (or behavioral) model. In the model, each stage of the gesture is represented as a basic language unit, which we call a Gesture Word (GWord). A GWord can be modeled as either a static posture, unparameterized dynamic gesture or a parameterized gesture. A composite gesture is composed of one or more GWords with semantic constraints. These constraints are represented in the HMM, with nodes denoting GWords and edges describing the temporal and contextual relationship between GWords. The parameters of the model can be learned based on heuristics or via a probabilistic framework based on recorded training data. Online gesture recognition is carried out via greedy inference because we need to render the feedback to the user immediately and the underlying gestures are modeled using different schemes. Here, online processing means that the algorithm does not have access to future video frames.

Another limitation of current gesture systems is the relatively small gesture vocabulary and the limited number of users, as discussed in Chapter 2. Difficulty of data collection, lack of a good gesture model, and highly articulated hand motion all contribute to the scarceness of successful large gesture systems. To investigate the efficacy and efficiency of robustly modeling gestures using localized processors, we carry out a human factors experiment that involves fourteen gestures and sixteen users. Our gesture modeling experiments show that, using efficient motion capture and gesture modeling, we can recognize gestures on a fairly large vocabulary and user group.

### **6.1.1 Related Work in High-Level Gesture Modeling**

Our proposed framework is related to work in the field of visual modeling of human activity. The first class of models is the probabilistic graphical model, including HMMs and finite-state machines [131, 132]. Bregler [20] abstracted human activity in a three-layered model. In the data-driven approach, regions of coherent motion are used as low-level features. Dynamic models capture simple movements at the mid-level, and HMMs model the transition relationship among dynamic models. Pentland and Liu [97] proposed Markov Dynamic Models which couple multiple linear dynamic models (e.g. Kalman filters) with a high-level Markov model. In both models, the low-level activities are homogeneous, thus

allowing a straightforward incorporation into a high-level model.

Another class of methods is the grammar-based approach. Ivanov and Bobick [63] proposed a probabilistic syntactic approach for activity modeling. In their two-layered model, a discrete symbol stream is generated from continuous low-level detectors and then parsed with a context-free grammar. Minnen et al. [83] described high-level expectation of temporally extended and well-ordered activity using parameterized stochastic grammars. They extend stochastic grammars by adding event parameters, state checks, and sensitivity to an internal scene model to analyze and predict object interaction events, such as Towers of Hanoi task.

Most research has concentrated on learning the parameters of the model, assuming that the structure and topology of the model is known. Galata et al. [43] proposed an approach to learn the size of structure of the stochastic model for high-level activity recognition.

The main contribution of this work is to investigate a high-level language model to integrate the three different low-level gesture forms in a coherent manner. We extend the state-of-the-art in gesture modeling by relaxing the assumption that the low-level gesture primitives have a homogeneous form: e.g., all can be modeled with a HMM. We also investigate the efficacy of modeling multiple subjects based on learning techniques.

## 6.2 Modeling Composite Gestures

As discussed in Section 4.2.1, HMMs are probabilistic graphical models for modeling the spatial and temporal characteristics of dynamic processes, such as speech and activity. HMMs provide a mathematically sound framework for learning and probabilistic inference.

Most previous work in gesture and activity recognition assume a consistent model for all low-level processes (GWords). We propose to use HMMs to integrate multiple, multimodal low-level gesture processes into a high-level composite gesture. Intuitively, we combine multiple GWords to form a *Gesture Sentence* that corresponds to a complete interaction task. For example, grasping a virtual object  $\rightarrow$  moving it  $\rightarrow$  dropping the object

(Figure 6.1).

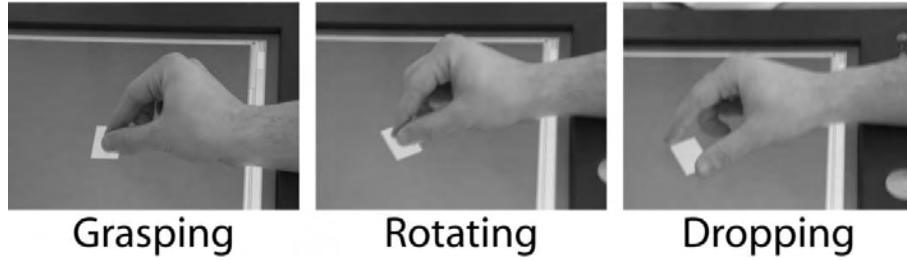


Figure 6.1: Composite gesture example with a corresponding graphical model.

In the remainder of this section, we define notation in Section 6.2.1 and present our construction of the composite gestures using HMMs in Section 6.2.2. We formulate the learning of the gesture model in Section 6.2.3. The gesture inference is discussed in Section 6.2.4. Section 6.2.5 we discuss different types of GWords.

### 6.2.1 Definitions

As discussed in Section 4.2.1, we denote the parameter of a HMM using  $\lambda = \{\mathcal{V}, A, B, \pi\}$ . Here, we explicitly using  $\mathcal{V}$  to represent the set of states with each  $v \in \mathcal{V}$  representing a GWord. Associated with each state  $v$  is a probability function  $B(\mathcal{S}|v)$ , which measures the observation likelihood of  $\mathcal{S}$  for a given GWord  $v$ . Note that  $\mathcal{S}$  represents a segment of video sequences, not a discrete symbol.  $A$  describes the transition probability among states, which intuitively models the temporal relationship between successive gesture units in the composite gesture.  $\pi$  models the prior of each gesture.

### 6.2.2 The Gesture Language

Our first-order HMM assumes that current GWord is dependent only on the previous one. The HMM model represents the relationship between pairs of GWords. Formally, given a vocabulary  $\mathcal{V}$ , define a GWord sequence  $\mathcal{W} = \{v_1, \dots, v_k\}$  where  $v_i \in \mathcal{V}$ . Thus, a composite gesture is a path through the HMM. In Figure 6.2, we give an example HMM that can model six gestures. For example, the path  $1 \rightarrow 3 \rightarrow 6 \rightarrow 10$  is a candidate Gesture Sentence.

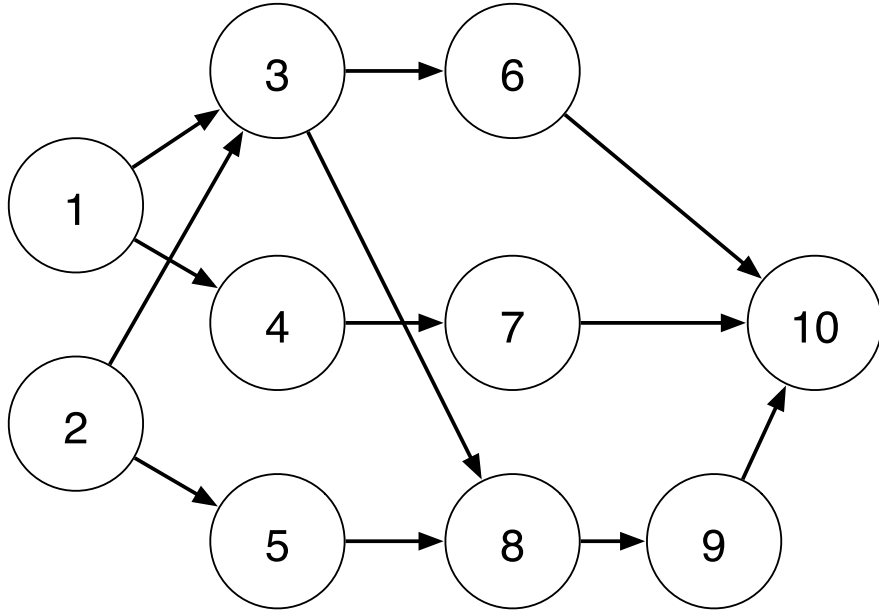


Figure 6.2: Example HMM used to represent the gesture language model.

We embed the language model into the HMM by associating nodes with individual GWords and assigning transition probabilities to represent the contextual constraints among GWords. Then the probability of observing the sequence in the model is:

$$P(\mathcal{W}) \doteq P(v_1, \dots, v_k) = \pi(v_1) \prod_{i=2}^k A(v_i|v_{i-1}) \quad (6.2.1)$$

As defined in Section 6.2.1, each state of the HMM models a specific GWord with its corresponding observation likelihood. Given an image sequence  $\mathcal{S}$  we can construct a candidate segmentation (Section 6.2.4) that splits the sequence into  $p$  subsequences  $\{\mathcal{S}_1 \dots \mathcal{S}_p\}$ . We match each of the subsequences to a GWord thus creating a Gesture Sentence  $\mathcal{W}$ . Assuming conditional independence of the subsequences given the segmentation and the observation likelihood of a subsequence only depends the corresponding GWord, the observation likelihood of the sequence is

$$P(\mathcal{S}|\mathcal{W}) = \prod_{i=1}^p B(\mathcal{S}_i|v_i) \quad (6.2.2)$$

Then, the overall probability of observing the Gesture Sentence is

$$\begin{aligned}
 P(\mathcal{W}|\mathcal{S}) &\propto P(\mathcal{W}) \cdot P(\mathcal{S}|\mathcal{W}) \\
 &= \pi(v_1) \prod_{i=2}^p A(v_i|v_{i-1}) \cdot \prod_{i=1}^p B(\mathcal{S}_i|v_i)
 \end{aligned} \tag{6.2.3}$$

### 6.2.3 Learning the Gesture Model

One distinct difference between our model and traditional discrete HMMs is that, the observation space is heterogeneous. Each low-level GWord has its own feature space and temporal characteristics. For example, a posture GWord can use direct image-based pattern matching to carry out the recognition; while a dynamic gesture which is modeled using a HMM may use a discrete feature space to characterize each frame and model the whole gesture as a sequence of discrete symbols. In other words, we do not have a uniform observation feature space in our model.

To resolve this problem, we assume that the observation likelihood  $B$  is trained separately by individual low-level gesture units (in Section 6.3 and Section 6.4 we discuss our implementations). We also assume that the observation probabilities of these units are normalized on the same scale to allow fair comparison among them. Thus, we only need to address the problem of learning and inference on the high-level gesture model. Specifically, we learn the parameters of the language model embedded in transition probabilities  $A$  and the prior  $\pi$ . We describe three basic techniques to learn the model: supervised, unsupervised, and hybrid.

#### Supervised Learning

Given a set of  $n$  labeled GWord sequences  $\mathcal{L} = \{\mathcal{W}_1 \dots \mathcal{W}_n\}$  with  $\mathcal{W}_i = \{v_{(i,1)}, \dots, v_{(i,m_i)}\}$  where  $m_i$  is the length of sequence  $\mathcal{W}_i$  and  $v_{(i,j)} \in \mathcal{V}$ . The GWord prior is given by

$$\pi(v_k) = \frac{\sum_{i=1}^n \delta(v_k, v_{(i,1)})}{n} \tag{6.2.4}$$

where  $\delta(\cdot)$  is the Kronecker delta function and  $v_k \in \mathcal{V}$ . The prior computes the probability that a Gesture Sentence begins with a certain GWord. The transition probability is given by the following equation.

$$A(v_l|v_k) = \frac{\sum_{i=1}^n \sum_{j=1}^{m_i-1} \delta(v_k, v_{(i,j)}) \cdot \delta(v_l, v_{(i,j+1)})}{\sum_{i=1}^n \sum_{j=1}^{m_i-1} \delta(v_k, v_{(i,j)})} \quad (6.2.5)$$

Intuitively, Equation 6.2.5 measures the transition probability from a GWord  $v_k$  to another GWord  $v_l \in \mathcal{V}$  by accumulating the number of GWord pairs  $v_k \rightarrow v_l$  and normalizing by the number of pairs beginning with  $v_k$ .

### Unsupervised Learning

Given a set of  $n$  unlabeled image sequences  $\mathcal{U} = \{U_1 \dots U_n\}$ . We generate an initial model  $M_0$  in a uniform fashion based on the structure of the HMM. We can use additional heuristics based on the specific application to refine the uniform initialization. We train the model using an EM-like [9] iterative algorithm.

1.  $\lambda \leftarrow \lambda_0$
2. Compute the best labeling (Section 6.2.4) for each sequence in  $\mathcal{U}$  based on the current model  $M$ .
3. Using the supervised learning algorithm (discussed previously), refine the model  $\lambda$ .
4. Repeat until a fixed number of iterations is reached or the change of the model in successive iterations is small.

Essentially, this algorithm is similar to traditional Viterbi-based learning of HMMs. The difference is that we use a separate labeling procedure instead of Viterbi search.

### Hybrid Learning

Given a set of labeled GWord sequences  $\mathcal{L}$  and a set of unlabeled image sequences  $\mathcal{U}$ . We generate an initial model  $\lambda_0$  using the labeled sequences with the supervised learning algorithm discussed above. Then, we refine the model in an iterative manner similar to the one used in unsupervised learning.

## 6.2.4 Inference on the Gesture Model

Given an image sequence  $\mathcal{S}$  of length  $m$  and an HMM with an embedded model, we construct the inference problem as the search for the best labeling  $\mathcal{L}$  of  $\mathcal{S}$  that maximizes the overall probability given in Equation 6.2.3. Formally, the inference problem is stated as

$$\{v_1^* \dots v_p^*\} = \arg \max_{\mathcal{W}=f(\mathcal{S})} P(\mathcal{W}) \cdot P(\mathcal{S}|\mathcal{W}) \quad (6.2.6)$$

where  $\mathcal{S} \doteq \{\mathcal{S}_1 \dots \mathcal{S}_p\}$ ,  $f(\mathcal{S}) = \{v_1 \dots v_p\}$  is a one-to-one mapping from a sequence segmentation to a Gesture Sentence, and  $p$  is unknown. Let  $g(\cdot)$  be the mapping from subsequence  $\mathcal{S}_i$  to a GWord  $v_i$ ; it is computed using the maximum-likelihood criterion:

$$g(\mathcal{S}_i) = \arg \max_{v_j \in \mathcal{V}} P(\mathcal{S}_i|v_j) \quad (6.2.7)$$

Theoretically, the inference problem in Equation 6.2.6 could be solved by an exhaustive search. However, the combinatorial complexity is high. Many algorithms, such as Viterbi search, N-Best search, and pruning have been proposed to resolve this problem efficiently [64, 100]. However, in our scheme, the fundamental differences in the three types of low-level gesture processors makes the optimization more difficult. For example, there is no uniform feature space nor homogeneous models. In addition, online processing is a prerequisite for human-computer interfaces. We have to make decisions and render visual feedback to the user with very little delay. Thus, we propose a sub-optimal, greedy algorithm.

Initialize the algorithm by setting  $\mathcal{S}_0 = \emptyset$ . At stage  $t$  in the algorithm processing, we search for the best transition from  $v_t$  to  $v_{t+1}$  which maximizes path probability, defined as the product of the transition probability  $A(v_{t+1}|v_t)$  (note at time  $t = 0$ ,  $A(v_1|v_0)$  is just the prior  $\pi(v_1)$ ) and the observation probability  $B(\mathcal{S}_{t+1}|v_{t+1})$ . The beginning of subsequence  $\mathcal{S}_{t+1}$  is set as the end of  $\mathcal{S}_t$ . To determine the end of the subsequence  $\mathcal{S}_{t+1}$  and thus make the greedy path choice, we incrementally increase the length of the subsequence until the path to one of the children  $c$  meets both of the following two conditions.

1. The observation probability of the child passes a threshold  $\tau_c$ . We discuss a supervised technique for learning the node thresholds below.

2. The path probability of  $c$  is highest among all of the children of node  $v_t$ . Formally,
 
$$c = \arg \max_{v_{t+1}} A(v_{t+1}|v_t) \cdot B(\mathcal{S}_{t+1}|v_{t+1}).$$

In Figure 6.3 we show a graphical depiction of a stage in the middle of the greedy algorithm. In the figure, at stage  $t + 1$ , child  $c_2$  of node  $v_t$  is chosen. We see that at the end of stage  $t + 1$  the end of sequence  $\mathcal{S}_{t+1}$  has been determined.

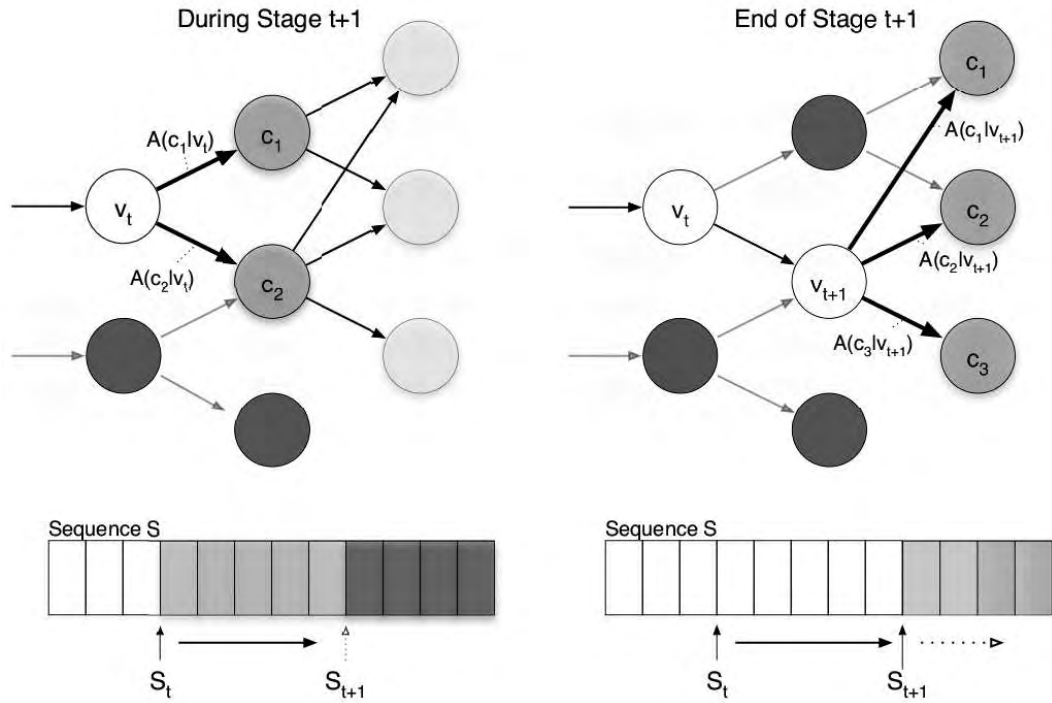


Figure 6.3: Graphical depiction of two stages of the proposed greedy algorithm for computing the inference on the gesture model. Black nodes are not on the best path and are disregarded, and white represents past objects on the best path. Dark gray nodes represent current candidate gestures to be evaluated.

Essentially, this greedy algorithm is similar to Viterbi search procedure. The difference is that, at each time  $t$ , instead of propagating the path to all possible states, we make a greedy choice and select the best one and discard other choices. This makes sure that we can render feedback to the user as soon as a new GWord is recognized, thus minimizing the response delay.

We learn the individual node thresholds using a supervised technique. Given a

set of labeled GWord sequences and segmented image sequence pairs  $(\mathcal{W}_i, \mathcal{S}_i) \in \mathcal{D}$ . We pose the problem of determining the threshold  $\tau_v$  for GWord  $v \in \mathcal{V}$  as finding the minimum observation probability for all occurrences of  $v$ :

$$\tau_v = \min_{(\mathcal{W}_i, \mathcal{S}_i) \in \mathcal{D}} \min_{v_i \in \mathcal{W}_i \text{ and } \delta(v_i, v)} B(\mathcal{S}_i | v) \quad (6.2.8)$$

First, we initialize all the thresholds to 0,  $\tau_v = 0, \forall v \in \mathcal{V}$ , to handle the case where  $v$  does not occur in  $\mathcal{L}$ . Then, for all GWords  $v \in \mathcal{V}$  we compute  $\tau_v$  according to Equation 6.2.8.

### 6.2.5 Three Low-Level Gesture Processors

As discussed in Section 6.1, there are three main classes of gestures in current virtual reality systems: static postures, dynamic gestures, and parameterized gestures. In our high-level gesture model presented earlier, each state corresponds to one of these three types of gesture processes.

#### Static Gesture

Static postures are based on the recognition of single discriminative frames of video. Hence, static postures simplify gesture processing by discarding all temporal information. For example, in the current literature, most alphanumeric symbols in ASL are represented as static postures [163]. Commonly used approaches to model the postures include appearance-based templates, shape-based models, and 3D model-based methods.

#### Non-parametric Dynamic

Non-parametric dynamic gestures capture temporal processes that carry only qualitative information; no quantitative information is present. Hence, these gestures are potentially more discriminative than static postures because of the additional temporal characteristics. For example, the ‘j’ and ‘z’ letters in the ASL have a temporal signature; i.e. the spatial trajectory of the finger over time is used to discriminate between the ‘i’ and the ‘j’. Hidden Markov models [100, 122, 149, 152], finite-state machines [58], and direct tempo-

rary template matching [13] are common methods used to model non-parametric dynamic gestures.

### Parametric Dynamic

Parametric dynamic gestures are the most complex among the three types because they not only incorporate a temporal dimension but also encode a set of quantitative parameters. For example, in explaining the height of a person using an outstretched hand, the distance between the ground and the hand gives a height estimate. Parametric hidden Markov models [139] have been proposed to model a single spatial variable. Segment model has also been discussed to represent a variable-length sequence of observation vectors in HMM-based speech processing [93]. However, most of the techniques are based on visual tracking, as are the ones presented in this dissertation.

The parametric dynamic gestures bring an added degree of difficulty to the recognition process because they can have too high a degree of temporal variability to be captured by a standard model like an HMM. For example, Figure 6.1 shows a composite gesture for grabbing, moving, and dropping a virtual object. In general, the moving gesture will appear quite arbitrary because the user has the freedom to navigate the entire workspace and also pause for variable amounts of time before dropping the object.

### Rescaling Observation Probability for Low-Level Gestures

As mentioned in Section 6.2.3, we need to scale the observation probability of all low-level GWords to allow fair comparison among them for inference on the gesture model. We take a straightforward way to scale all observation to a scalar between 0 and 1. We mapping all positive gestures to the range between  $\tau$  and 1, where  $\tau$  is the observation threshold defined in Section 6.2.3. For fair comparison, we will select a common  $\tau$  for all GWords. We use a linear function  $F : P \rightarrow P^* \doteq a * P + b$  to represent the transform from initial probability  $P$  to scaled likelihood  $P^*$ .

We compute the parameters ( $a$  and  $b$ ) for each GWord. First, we compute the minimum ( $P_{min}$ ) and maximum ( $P_{max}$ ) observation probability in the training data. We compute the parameters that maps  $P_{min}$  to  $\tau$  and  $P_{max}$  to 1.

$$P_* = \frac{P(1 - \tau) + \tau P_{max} - P_{min}}{P_{max} - P_{min}} \quad (6.2.9)$$

## 6.3 Experiments with the High-Level Gesture Model

In this section, we discuss an experiment that involves 10 low-level gesture GWords and 6 high-level gesture sentences. We analyze the proposed model for recognizing composite gestures by constructing a gesture set and the corresponding gesture model. We carry out the experiment on the 4DT system that has been presented in Chapter 5.

### 6.3.1 Gesture Set

The goal of the proposed framework is to facilitate the integration of different types of gestures (Section 6.2.5) and thus, natural interaction in virtual environments. To that end, we present an experimental gesture set with ten elements (GWords) with each of the three gesture types represented.

#### Low-Level GWords

The gesture set is designed to be used in general manipulative interfaces where actions such as selecting, grasping, and translating are required. Table 6.1 contains graphical depictions of each GWord. For dynamic gestures, we show three example images during the progress of the gesture.

- **Press.** Press is the static posture of a single finger activating the interface component.
- **Left.** Left is a dynamic, non-parametric motion of a finger to the left with respect to the interface component.
- **Right.** Right is a dynamic, non-parametric motion of a finger to the right with respect to the interface component.
- **Back.** Back is a dynamic, non-parametric retraction of the finger off the interface component.

- **Twist.** Twist is a clockwise twisting motion of a finger atop the interface component (dynamic, non-parametric).
- **Grab 1.** The first grabbing gesture is the dynamic, non-parametric motion of two fingers approaching the interface component open and closing once they have reached it.
- **Grab 2.** The second grabbing gesture is the dynamic, non-parametric motion of two fingers approaching the interface component open and remaining open upon reaching it.
- **Track.** Track is a parametric gesture that tracks two translational degrees-of-freedom.
- **Rotate.** Rotate is a parametric gesture that tracks one rotational degree-of-freedom.
- **Stop.** Stop is a static posture represented by an open hand atop the interface component.
- **Silence.** Silence is a static posture that acts as an ending point of all gesture sentences. In this experiment, we define silence as the scene when the user retracts his/her hand from the interaction component.

### Structure of Gesture Model

With the algorithms presented in Section 6.2, we construct and train an HMM model to be the *interaction language*. Figure 6.4 is a graphical depiction of the model. A simple Gesture Sentence is thus **Press** → **Left**: the user approaches an interface component with an outstretched finger and then swipes his or her finger to the left. For example, such a composite gesture could be used to delete an interaction component. A more complex Gesture Sentence involving all three types of low-level GWords is **Grab 1** → **Track** → **Stop**. This Gesture Sentence could be widely used in VR to grab and move virtual objects.


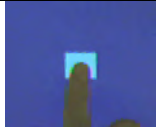
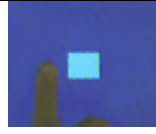



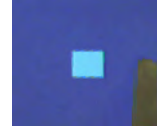
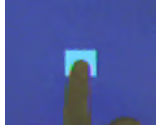

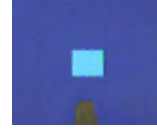
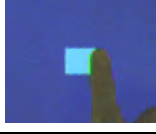

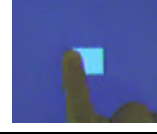



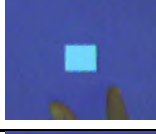


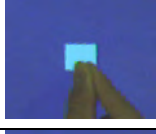
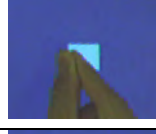
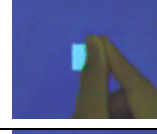
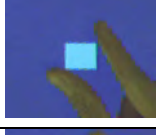



| GWord  | Stage 1   | Stage 2  | Stage 3   |
|--------|---|--|---|
| Press  |    |  |   |
| Left   |    |    |    |
| Right  |    |    |    |
| Back   |    |    |    |
| Twist  |    |    |    |
| Grab 1 |   |   |   |
| Grab 2 |  |  |  |
| Track  |  |  |  |
| Rotate |  |  |  |
| Stop   |  |  |   |

Table 6.1: Example images of basic GWords.

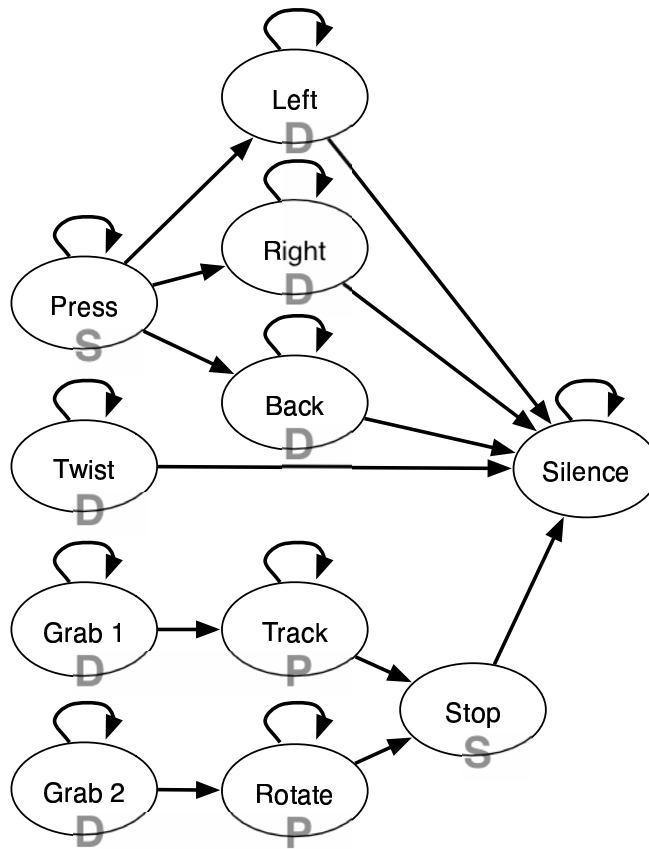


Figure 6.4: The gesture model we constructed for our experimental setup. The states are labeled as per the discussion in Section 6.3.1. Additionally, each node is labeled as either **P**arametric, **D**ynamic, or **S**tatic gestures.

### 6.3.2 Implementation of Low-Level GWords

As discussed in Section 6.2.5, we include three types of low-level gesture processing: static posture, non-parametric dynamic, or parametric dynamic. In this section we discuss the construction of these low-level processors for our experimental setup. However, from the perspective of the high-level gesture framework, the specific construction of the low-level processors is casual and independently.

## Processors for Static Postures

Static postures are based on the recognition of single discriminative frames of video. A multitude of potential methods exist in the literature for such recognition: SIFT keys [74] and Shape Contexts [8] for example. Exploiting the principle of local image analysis from the VICs paradigm (Chapter 4), we use three-layer neural networks (Section 5.1.4) to model the appearance of the postures.

Basically, given a pair of intensity images, we fix a local image neighborhood of  $128 \times 128$  pixels corresponding to the region in the image defined by its interface component mapping. As input to the network, we choose a coarse sub-sampling ( $16 \times 16$ ) and take non-overlapping pixel-neighborhood averages. We train a standard three-layer network with a real-value output indicating the posterior probability of the model matching the input.

## Processor for Non-parametric Dynamic Gestures

We model the dynamics of the motion of the finger using discrete forward HMMs, as discussed in Chapter 5. A 6-state forward HMM is used to model the dynamics of each gesture. The parameters of the HMM are learned via the standard forward-backward algorithm based on the recorded gesture sequences. The gesture recognition is based on the probability that each HMM generates the given gesture image sequence.

## Processors for Parametric Dynamic

The implementation of a parametric, dynamic processor is dependent on the task for which it is to be used. For example, in our gesture set, we require both a translational and a rotational processor. Again, many potential techniques exist for tracking the local motion of an image patch or pair of image patches. In our experiments, we used a *filtered-detection* algorithm: for each frame of video, we detect the feature(s) of interest and use a linear Kalman filter to model the dynamics of motion.

The feature detector is implemented using a neural network: the input is features extracted from sub-sampled image and the output is  $x$  and  $y$  position of the focus point of the gesture. For example, for translating, this focus point is defined as the image point

where the two grasping fingertips (thumb and index finger) meet. Assuming we can detect the same exact point every frame, tracking this grasping-point provides the two translational degrees-of-freedom. While it is difficult (or impossible) to detect exactly the same point every frame, in practice, the Kalman filter handles small variations in the point detection.

In our experiment, we use 25 hidden nodes for the neural network. The average tracking error is 3.07 pixels on the training data and 6.55 on the testing data.

### 6.3.3 Experimental Results Using Gesture Model

Table 6.1 shows our vocabulary of 6 possible composite gestures. To quantitatively analyze the model, we recorded a training set of 100 video sequences each corresponding to one of the 6 gestures. The length of the sequences vary from 30 to 90 frames (at 10 frames-per-second). These sequences are not used in training the low-level gesture units. For the supervised training, we manually labeled each frame of the video sequences with a GWord. For unsupervised learning, we initialized a uniform language model and used the algorithm in Section 6.2.3 to refine the model. After 2 iterations, the model converged.

We compare the language models after supervised and unsupervised learning in Tables 6.2 and Table 6.3, respectively. The models are presented as adjacency matrices such that each row represents the probability of transition from a GWord (leftmost column) to other GWords (or itself). It can be seen that the two models have similar structure. It shows that even without good heuristics or labeled data, our unsupervised learning algorithm can still capture the underlying language model from raw gesture sequences.

However, there are differences worth mentioning. For example, the prior for **Stop** from unsupervised learning is 0.03, but there are no sequences in the training corpus that begin with it. This is caused by the failure of the inference algorithm given a uniform language model. Second, we see a difference in the self-transition probability for the **Press** GWord. In the labeled data, we fixed the duration of **Press** to one frame, but with a uniform model, a static posture can last for several consecutive frame via self-transition.

During testing, we used the proposed greedy inference algorithm to analyze the video sequences. In Table 6.4, we present the recognition accuracy for the gestures for both gesture models. For each sequence, we compared its known composite gesture identity with

| Prior                  |      |      |      |       |       |       |      |      |     |      |      |
|------------------------|------|------|------|-------|-------|-------|------|------|-----|------|------|
| Word                   | L.   | R.   | B.   | Tw.   | G.1   | G.2   | Tr.  | Ro.  | Pr. | St.  | Si.  |
|                        | 0    | 0    | 0    | 0.167 | 0.167 | 0.167 | 0    | 0    | 0.5 | 0    | 0    |
| Transition Probability |      |      |      |       |       |       |      |      |     |      |      |
| Word                   | L.   | R.   | B.   | Tw.   | Gr.1  | Gr.2  | Tr.  | Ro.  | Pr. | St.  | Si.  |
| L.                     | 0.94 | 0    | 0    | 0     | 0     | 0     | 0    | 0    | 0   | 0    | 0.06 |
| R.                     | 0    | 0.93 | 0    | 0     | 0     | 0     | 0    | 0    | 0   | 0    | 0.07 |
| B.                     | 0    | 0    | 0.84 | 0     | 0     | 0     | 0    | 0    | 0   | 0    | 0.16 |
| Tw.                    | 0    | 0    | 0    | 0.93  | 0     | 0     | 0    | 0    | 0   | 0    | 0.07 |
| G.1                    | 0    | 0    | 0    | 0     | 0.94  | 0     | 0.06 | 0    | 0   | 0    | 0    |
| G.2                    | 0    | 0    | 0    | 0     | 0     | 0.94  | 0    | 0.06 | 0   | 0    | 0    |
| Tr.                    | 0    | 0    | 0    | 0     | 0     | 0     | 0.96 | 0    | 0   | 0.04 | 0    |
| Ro.                    | 0    | 0    | 0    | 0     | 0     | 0     | 0    | 0.95 | 0   | 0.05 | 0    |
| Pr.                    | 0.33 | 0.33 | 0.33 | 0     | 0     | 0     | 0    | 0    | 0   | 0    | 0    |
| St.                    | 0    | 0    | 0    | 0     | 0     | 0     | 0    | 0    | 0   | 0.7  | 0.3  |
| Si.                    | 0    | 0    | 0    | 0     | 0     | 0     | 0    | 0    | 0   | 0    | 1    |

Table 6.2: Parameters of the gesture model trained using supervised learning.

| Prior                  |      |      |      |      |      |      |      |      |      |      |      |
|------------------------|------|------|------|------|------|------|------|------|------|------|------|
| Word                   | L.   | R.   | B.   | Tw.  | G.1  | G.2  | Tr.  | Ro.  | Pr.  | St.  | Si.  |
|                        | 0    | 0    | 0    | 0.1  | 0.11 | 0.16 | 0    | 0    | 0.6  | 0.03 | 0    |
| Transition Probability |      |      |      |      |      |      |      |      |      |      |      |
| Word                   | L.   | R.   | B.   | Tw.  | G.1  | G.2  | Tr.  | Ro.  | Pr.  | St.  | Si.  |
| L.                     | 0.91 | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0.09 |
| R.                     | 0    | 0.88 | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0.12 |
| B.                     | 0    | 0    | 0.83 | 0    | 0.01 | 0    | 0    | 0    | 0    | 0    | 0.16 |
| Tw.                    | 0    | 0    | 0.0  | 0.95 | 0    | 0    | 0    | 0    | 0    | 0    | 0.05 |
| G.1                    | 0    | 0    | 0    | 0    | 0.82 | 0    | 0.14 | 0    | 0    | 0.02 | 0.02 |
| G.2                    | 0    | 0    | 0    | 0    | 0    | 0.77 | 0.04 | 0.15 | 0    | 0.04 | 0    |
| Tr.                    | 0    | 0    | 0    | 0    | 0.02 | 0    | 0.77 | 0.03 | 0    | 0.16 | 0.02 |
| Rt.                    | 0    | 0    | 0    | 0    | 0    | 0.01 | 0.03 | 0.90 | 0    | 0.06 | 0    |
| Pr.                    | 0.02 | 0.02 | 0.03 | 0    | 0    | 0    | 0    | 0    | 0.91 | 0    | 0.02 |
| St.                    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0.77 | 0.23 |
| Si.                    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    |

Table 6.3: Parameters of the gesture model trained using unsupervised learning.

| <b>Gesture Sentence</b> | <b>Supervised</b> | <b>Unsupervised</b> |
|-------------------------|-------------------|---------------------|
| Press → Left            | 97.3              | 97.3                |
| Press → Right           | 85.7              | 78.6                |
| Press → Back            | 88.9              | 90.4                |
| Twist                   | 96.4              | 96.4                |
| Grab 1 → Track → Stop   | 93.3              | 82.1                |
| Grab 2 → Rotate → Stop  | 97.9              | 97.9                |

Table 6.4: Recognition accuracy of the gesture model used in our experimentation.

the GWord output of the model. We consider the output correct if it matches the GWord sentence at every stage.

We can see from the results that the proposed high-level gesture modeling can effectively recognize compositions of low-level gestures with heterogeneous observation. Because of the multimodal nature of these gestures, it would be difficult for unimodal approaches to efficiently parse these continuous gestures.

These composite gestures would be impossible to recognize using traditional unimodal techniques, while our formulation takes advantage of high-level constraints to integrate fundamentally different low-level gesture units in a coherent probabilistic model.

However, the recognition accuracy for gesture **Press** → **Right** and gesture **Press** → **Back** are relatively poor. From visual inspection of the recognition algorithm’s output, we find that this is due to the greedy algorithm. The **Left**, **Right**, and **Back** are modeled with HMMs and trained with relatively long sequences (e.g. 20 frames). However, during inference, the greedy algorithm jumps to a conclusion based on an shorter subsequences (e.g. 7 frames). In our experiments, we see a bias toward the **Left** GWord for these incomplete subsequences.

The recognition results from the supervised and the unsupervised learning are comparable. This suggests that our approach to gesture recognition can perform well without a heuristic prior or manually labeled data. Hence, our method is less susceptible to the curse of dimensionality which, in our case, is that the amount of data (labeled, for supervised learning) required for learning generally increases exponentially with the number of GWords.

## 6.4 Modeling Gestures Across Multiple Users

As discussed in Chapter 2, although there are many gesture systems that handle dynamic gestures based on visual modeling, they are usually limited by a small vocabulary (usually less than ten gestures). Furthermore, most systems only deal with data from very few subjects. This lack of successful large-scale gesture systems can be attributed to the difficulty of collecting visual data, the absence of a robust and efficient gesture model, and lack of efficient algorithms.

In this dissertation, we have proposed an efficient and robust vision-based HCI platform, i.e., the VICs paradigm (Chapter 4) and 4DT system (Chapter 5). Through component mapping and robust hand segmentation, the system provides an excellent platform for visual data collection. We also investigate various methods to model both low-level gestures (including postures and dynamic gestures) and high-level composite gestures. These models are implemented using efficient algorithms aimed to achieve real-time processing, thus allowing our gesture interface to be integrated into average computation environments.

To further investigate the efficacy of our system, we carry out a human factors experiment that involves sixteen subjects and fourteen composite gestures. Basically, we intend to investigate our model's capability of capturing variances among users, efficacy of high-level modeling of continuous gestures, and the user's feedback regarding our gesture interface.

In the remainder of the section, we present our gesture set in Section 6.4.1. We discuss our experiment setup in Section 6.4.2. Section 6.4.3 presents our implementation of low-level gestures and high-level gesture model.

### 6.4.1 Gesture Vocabulary

To design a good gesture vocabulary is not easy. Besides standard sign languages (for example, American Sign Language), there is no standard vocabulary for manipulative and controlling gestures which is evident in the research literature. Our design is mainly based on three principles:

1. **Natural and intuitive:** After all, our goal is to allow intuitive interaction. Thus it would be unreasonable for user to adapt to cumbersome gestures. For example, we allow the user to pick a component and move it around by closing the thumb and the index finger to mimic an everyday routine.
2. **General and sufficient:** We aim to include those “core” gestures that are most common and crucial for carrying out general controlling tasks, such as moving, rotating and selecting.
3. **Easy to learn:** Given the lack of a common vocabulary for controlling gestures, it would be very difficult for average users to learn and remember many gesture commands. A tradeoff must be made between ease of learning and the power of the representation.

### Low-Level Gesture Vocabulary

Following our principles, we design a vocabulary that includes seven postures, three dynamic gestures, and three parameterized gestures. We present the details of each gesture in the following list:

1. **Push:** Push is the posture of a single finger resting on the center of the interface component intending to activate it. This definition is slightly different from intuition: pushing a button normally would involve moving the finger toward the button, pressing it and then retracting the finger, as modeled in Chapter 4. However, from our past experiences, we find it awkward to restrict the user to follow such a fixed temporal pattern. Also, we allow real-time gesture-based interaction and the dynamics of such gestures can vary significantly. For example, on a calculator application, the user can trigger a button, then move quick to a neighboring button from all possible directions and can move very fast. In our current system, we can only capture a few frames (sometimes fewer than three) corresponding to such a fast gesture. Therefore, we choose to represent such gestures using their characteristic frame. Same choices are made for Press-Left, Press-Right, Drop and Grab, which will be explained shortly.

2. **Press-Left:** Press-Left is the posture of a single finger placing on the left edge of the component to initiate a twisting anti-clockwise.
3. **Press-Right:** Similar to Press-Left, except the finger is on the right side to initiate clockwise twisting.
4. **Pick:** Pick is the posture to mimic a picking gesture with index finger and thumb half-closed.
5. **Drop:** Drop is a posture with index finger and thumb open totally to put down a component.
6. **Stop.** Stop is a static posture represented by an open hand atop the interface component.
7. **Grab.** Grab is a static posture with index finger and thumb close together intended to drag a component to change its size.
8. **Silence.** Silence is a posture that is defined either as an “empty scene” or any gesture that is different from our valid postures listed above.
9. **Twist.** This is a dynamic gesture to represent the processing of twisting the component clockwise.
10. **Twist-Anti.** Similar to Twist, except the direction of dialing is anti-clockwise.
11. **Flip.** Flip is a dynamic gesture that mimics the process of flipping a coin by picking it up and turning it over.
12. **Move.** Move is a parametric gesture that tracks two translational degrees-of-freedom.
13. **Rotate.** Rotate is a parametric gesture that tracks one rotational degree-of-freedom.
14. **Resize.** Resize is a also a parametric gesture that tracks two degree-of-freedom that indicate the change of the size of the component. The difference against Move is that, in Resize, the position of the component does not change.

Table 6.5 shows example images of each low-level gesture. For dynamic gestures, we show a snapshot of the gesture at different stages. For example, for Twist, we show the starting point when the finger is on the right side of the icon, then in the middle of the gesture when the finger has moved down to the center of the bottom, and finally the ending configuration.

## High-Level Language Model

Table 6.6 lists the nine composite gesture that are modeled in our gesture model. Note that all sentences end with Silence. An example gesture sentence to move a component is to **Pick** it, **Rotate** it, then **Drop** it, and to end with a **Silence**.

A graphical depiction of our gesture model is shown in Figure 6.5. In our model, we allow self-loops for most postures; this is reasonable because in reality, the user may keep the posture for a short period of time.

Our gesture vocabulary are intended for general-purpose manipulation and controlling of a virtual environment. Example applications of this vocabulary include controlling a traditional WIMP interface, exploring a virtual 3D graphical world, and manipulating a virtual 2D or 3D design studio. This vocabulary can also be extended for more specific application areas; for example, finger spelling can be added for writing text on the interface.

### 6.4.2 Experimental Setup

#### System Implementation

We use the 4DT system( Section 5.1) as the experimental platform. To investigate the efficacy of our scheme of gesture analysis with different camera configurations, we assembled two parallel systems, each running on a separate PC and controlling its own stereo pair. The lower stereo system is about half the height of the higher cameras and has higher spatial resolution. Figure 6.6 shows the setup of the system.

We implement a real-time data recording system to collect gesture data. We loosely synchronize the two separate recording processes by sending them synchronization messages through the X-Window server.





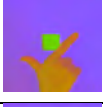
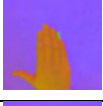


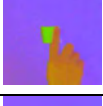

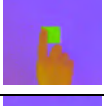
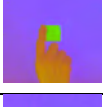
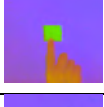
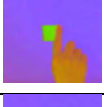

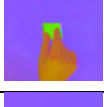
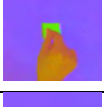
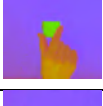

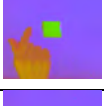
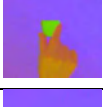
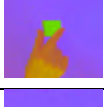
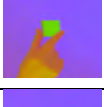



| GWord       | Category      | Stage 1   | Stage 2   | Stage 3   |
|-------------|---------------|---|---|---|
| Push        | Static        |    |   |   |
| Press-Left  | Static        |    |   |   |
| Press-Right | Static        |    |   |   |
| Pick        | Static        |    |   |   |
| Drop        | Static        |    |   |   |
| Stop        | Static        |    |   |   |
| Grab        | Static        |    |   |   |
| Silence     | Static        |   |   |   |
| Twist       | Dynamic       |  |  |  |
| Twist-Anti  | Dynamic       |  |  |  |
| Flip        | Dynamic       |  |  |  |
| Move        | Parameterized |  |  |  |
| Rotate      | Parameterized |  |  |  |
| Resize      | Parameterized |  |  |  |

Table 6.5: Example images of our gesture vocabulary.

| <b>Gesture</b> | <b>Sentence</b>                   |
|----------------|-----------------------------------|
| Pushing        | Push → Silence                    |
| Twisting       | Press-Right → Twist → Silence     |
| Twisting-Anti  | Press-Left → Twist-Anti → Silence |
| Dropping       | Pick → Drop → Silence             |
| Flipping       | Pick → Flip → Silence             |
| Moving         | Pick → Move → Drop → Silence      |
| Rotating       | Pick → Rotate → Drop → Silence    |
| Stopping       | Stop → Silence                    |
| Resizing       | Grab → Resize → Stop → Silence    |

Table 6.6: Composite gestures and the corresponding gesture sequences.

### Subject Recruiting

We recruited volunteers by sending email to several email lists in the Whiting School of Engineering of the Johns Hopkins University, including the seminar email list of Engineering Research Center, the JHU Robotics email list, and graduate student lists of the Department of Computer Science. Overall, there are more than 250 members on these lists. Sixteen subjects participated in our experiment. Among them, seven are female.

### Data Collection

We formulate the problem of natural interaction using gestures as a dual-duplex learning process (Section 4.1). Before recording, we use two approaches for the subject to learn the gestures: text and video. For each gesture, besides giving some text explanation of the gesture, we display a short video of an expert user carrying out the gesture. Then the subject can mimic the gesture on the system till he or she feels confident and comfortable to record real-time sequences.

For each subject, we record five sequences for each low-level gesture as shown in Table 6.5. Then we record three sequences for each composite gestures. We split the dataset into training set and testing set.

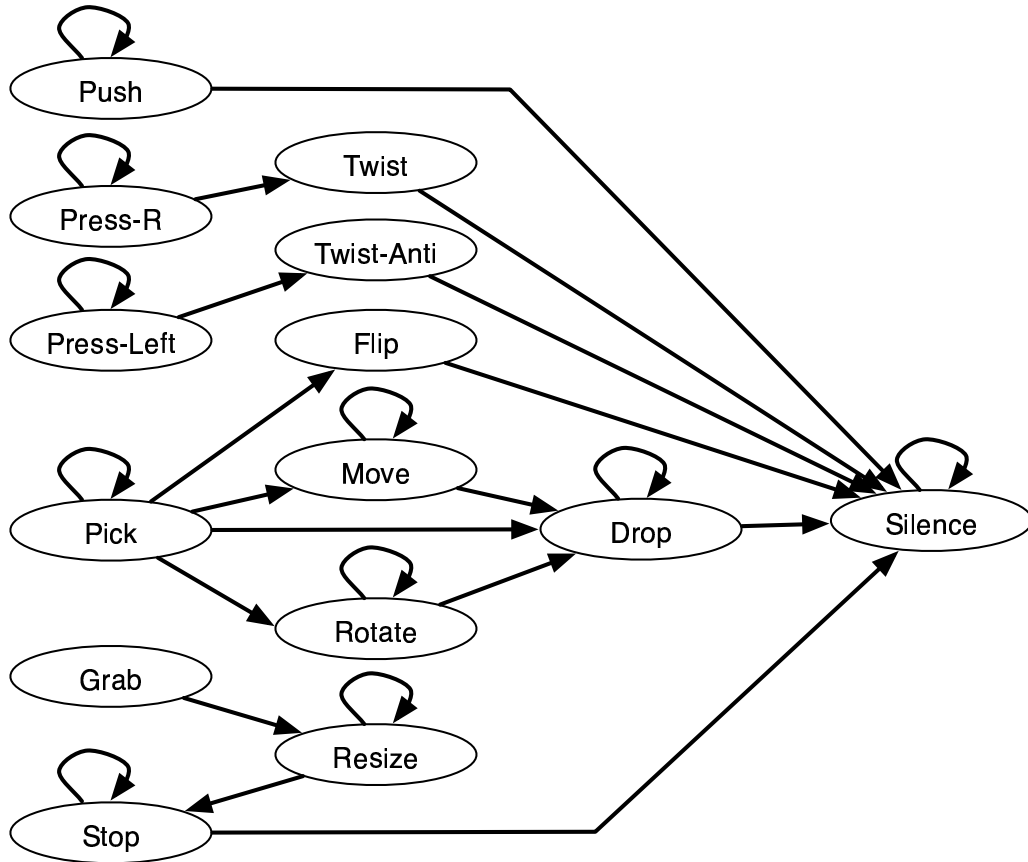


Figure 6.5: The gesture model for our large-scale gesture experiment.

### 6.4.3 Implementation of Low-Level Gestures

In this section, we explain our approaches to model three different low-level gestures: static, dynamic, and parameterized gestures.

#### Modeling Static Gestures

As mentioned in Section 5.2, our scheme of efficient motion capture provides rich visual cues for localized modeling of the appearance and motion of the hand. We propose two different ways to model postures based on our 3D appearance feature.

The first model is a 3-layer neural network. We train a neural network for each



Figure 6.6: Dual-stereo gesture system.

posture. The input is the raw 3D feature vector. The single real-value output indicates the confidence of positive recognition. During training and testing, we use data from other gestures as negative examples, for which the neural network ideally will output a very low confidence.

The second model is a histogram-based Bayesian classifier. First, we carry out VQ on the raw feature to reduce the high-dimensional data to a discrete symbol  $s_j$  in a vocabulary of size  $M$  (Section 5.2). Let  $N$  be the number of postures and  $v_i$  be the  $i$ th posture. Then from the training data, we compute a 2-D histogram  $H(v_i, s_j)$  for all the postures  $v_i$  over all symbols  $s_j$ . Basically, we accumulate the number of appearances of  $s_j$  in gesture  $v_i$ . We normalize  $H$  for both  $v_i$  and  $s_j$  to satisfy the probability axiom:

$$\begin{aligned} \sum_{i=1}^N H(v_i, s_j) &= 1, \text{ for all } j \in \{1, 2, \dots, M\} \\ \sum_{j=1}^M H(v_i, s_j) &= 1, \text{ for all } i \in \{1, 2, \dots, N\} \end{aligned} \quad (6.4.1)$$

For recognition of given symbol  $s$ , the class probability is simply represented as  $H(v_i, s)$  for each posture  $v_i$ . Assuming a uniform prior among all postures, then the classification is reduced to:

$$\begin{aligned} v_i &= \arg \max_{v_i} P(v_i|s) = \arg \max_{v_i} \frac{P(s|v_i)P(v_i)}{P(s)} \\ &= \arg \max_{v_i} P(s|v_i) = \arg \max_{v_i} H(v_i, s) \end{aligned} \quad (6.4.2)$$

### Modeling Dynamic Gestures

We use forward HMMs to model dynamic gestures (Section 5.3.1). Again, the input to the HMMs is discrete symbols computed via VQ on extracted appearance feature. As discussed in Section 6.3.3, one of the problems when we carry out inference on the our high-level model is that, the greedy algorithm might prematurely jumps to a conclusion based on incomplete video sequence. This can happen in standard HMMs, since the beginning part of the sequence can still match part the HMM model very well. To enforce the completeness of the observation, thus achieving coherent parsing of remaining GWords of the gesture sentence, we incorporate an extra parameter  $\rho_i = P(q_T = S_i)$  to model the probability of each state in the HMM ( $S_i$ ) to be the stopping state of a sequence of length  $T$  ( $q_T$ ).

To train this model parameter  $\rho$ , we follow the similar fashion of traditional Baum-Welch algorithm. Given a set of  $n$  training sequences, we denote  $T_i$  as the length of the length of the  $i$ th sequence, and  $q_{T_i}^i$  as the last state of the best state sequence found by Viterbi algorithm.

$$\rho_i = \frac{\sum_{i=1}^n \delta(S_i, q_{T_i}^i)}{n} \quad (6.4.3)$$

To incorporate this stopping state probability into HMMs for recognition, we only need to modify the probability that each HMM output a given sequence  $S$  as by multiplying the stopping-state probability:

$$P'(S|\lambda) = P(S|\lambda) * \rho(q_T) \quad (6.4.4)$$

where  $\lambda$  represents the HMM model;  $q_T$  is the last state; and  $P(S|\lambda)$  is the cumulative probability along the Viterbi path.

### Modeling Parameterized Gestures

In this experiment, our parameterized gestures are carried out on a 2D screen and involves relatively small occlusion and deformation. Furthermore, on the 4DT platform, the user’s hand is segmented from the background. This segmented hand region allows efficient 2D tracking.

We use region-based sum-of-square difference (SSD) tracking scheme proposed by Hager and Belhumeur [52]. Since we assume little illumination changes during a interaction session, we only need to model the geometric changes as:

$$f(x; u, \theta) = R(\theta)x + u \quad (6.4.5)$$

where  $\theta$  is the rotation angle and  $u$  is the translation.

We also implement a pyramid hierarchy of trackers at different levels of resolution to increase performance. Multiple iterations are carried out at each time frame to allow relative large inter-frame motion.

## 6.5 Results of Large-Scale Gesture Experiments

In this section, we present our experimental results for our human factor experiment which involves sixteen users and fourteen gestures. For most of the experiment, we report results for both stereo systems. For convenience, we will call them Higher Stereo and Lower Stereo respectively.

## 6.5.1 Experimental Results of Modeling Low-Level Gestures

### Recognition of Static Gestures

As discussed in Section 6.4.3, we use two different approaches to model static gestures: neural networks and histogram-based Bayesian classifier.

For neural networks, we use 20 hidden nodes. For each posture, we collect both positive and negative examples from the dataset. For each posture  $v$ , the negative examples are selected from data belonging to those postures other than  $v$ . As discussed in Section 6.4.2, we partition the whole dataset into a training set on which the model is trained, and a testing set. We test the trained model on the testing set. Table 6.7 and Table 6.8 shows the training and testing results for Higher Stereo and Lower Stereo, respectively. For each posture, we show the number of positive and negative frames in both training set and testing set. We also report the framewise accuracy for positive set, negative set and the two combined.

| Post. | Train Set |      | Train Accuracy |        |        | Test Set |      | Test Accuracy |        |        |
|-------|-----------|------|----------------|--------|--------|----------|------|---------------|--------|--------|
|       | Pos.      | Neg. | Pos.           | Neg.   | All    | Pos.     | Neg. | Pos.          | Neg.   | All    |
| Push  | 630       | 6874 | 100.00         | 100.00 | 100.00 | 244      | 2469 | 100.00        | 99.92  | 99.93  |
| Pr.-L | 94        | 7410 | 100.00         | 100.00 | 100.00 | 58       | 2757 | 94.83         | 100.00 | 99.89  |
| Pr.-R | 108       | 7396 | 99.07          | 100.00 | 99.99  | 54       | 2761 | 98.15         | 100.00 | 99.96  |
| Pick  | 3702      | 4091 | 99.95          | 100.00 | 99.97  | 1919     | 896  | 98.91         | 99.78  | 99.18  |
| Drop  | 147       | 7357 | 100.00         | 100.00 | 100.00 | 87       | 2728 | 94.25         | 100.00 | 99.82  |
| Stop  | 508       | 6996 | 100.00         | 100.00 | 100.00 | 293      | 2522 | 100.00        | 100.00 | 100.00 |
| Grab  | 116       | 7388 | 100.00         | 100.00 | 100.00 | 60       | 2755 | 91.67         | 100.00 | 99.82  |
| Sil.  | 2925      | 5175 | 100.00         | 99.96  | 99.98  | 1724     | 2815 | 99.88         | 98.93  | 99.56  |

Table 6.7: Posture recognition using neural networks for Higher Stereo.

For Bayesian classifiers, we use 96 clusters to carry out VQ. Table 6.9 and Table 6.10 shows the training and testing result.

It's evident that neural networks achieve better performance than histogram-based classifiers. Through iterative learning, neural networks achieve overall recognition ratio over 98% for all postures. For histogram-based approaches, the Higher Stereo has a much better recognition ratio than the Lower Stereo. One of the reasons is that for the lower cameras,

| Post. | Train Set |      | Train Accuracy |        |        | Test Set |      | Test Accuracy |        |        |
|-------|-----------|------|----------------|--------|--------|----------|------|---------------|--------|--------|
|       |           |      |                |        |        |          |      |               |        |        |
| Push  | 530       | 5569 | 100.00         | 100.00 | 100.00 | 291      | 2070 | 100.00        | 100.00 | 100.00 |
| Pr.-L | 90        | 6009 | 100.00         | 100.00 | 100.00 | 52       | 2309 | 96.15         | 100.00 | 99.92  |
| Pr.-R | 106       | 5993 | 100.00         | 100.00 | 100.00 | 58       | 2303 | 100.00        | 100.00 | 100.00 |
| Pick  | 2913      | 3186 | 100.00         | 100.00 | 100.00 | 1580     | 781  | 99.37         | 98.85  | 99.20  |
| Drop  | 144       | 5955 | 97.92          | 100.00 | 99.95  | 81       | 2280 | 96.30         | 100.00 | 99.87  |
| Stop  | 418       | 5681 | 100.00         | 100.00 | 100.00 | 243      | 2118 | 100.00        | 100.00 | 100.00 |
| Grab  | 110       | 5989 | 100.00         | 100.00 | 100.00 | 56       | 2305 | 92.86         | 99.91  | 99.75  |
| Sil.  | 2196      | 4311 | 100.00         | 99.93  | 99.95  | 1373     | 2361 | 99.63         | 97.63  | 98.37  |

Table 6.8: Posture recognition using neural networks for Lower Stereo.

| Post.   | Train Frames | Train Accuracy | Test Frames | Test Accuracy |
|---------|--------------|----------------|-------------|---------------|
| Push    | 630          | 96.98          | 344         | 100.00        |
| Press-L | 94           | 100.00         | 58          | 94.83         |
| Press-R | 108          | 99.07          | 54          | 98.15         |
| Pick    | 3572         | 96.95          | 1919        | 97.50         |
| Drop    | 147          | 100.00         | 87          | 98.85         |
| Stop    | 508          | 99.80          | 293         | 100.00        |
| Grab    | 116          | 98.28          | 60          | 95.00         |
| Silence | 6842         | 98.90          | 3802        | 98.68         |
| Overall | 12017        | 98.30          | 6617        | 98.39         |

Table 6.9: Posture recognition using histogram-based classifier for Higher Stereo.

| Post.   | Train Frames | Train Accuracy | Test Frames | Test Accuracy |
|---------|--------------|----------------|-------------|---------------|
| Push    | 530          | 98.87          | 291         | 91.07         |
| Press-L | 90           | 100.00         | 52          | 96.15         |
| Press-R | 106          | 100.00         | 58          | 96.55         |
| Pick    | 2913         | 97.05          | 1580        | 96.52         |
| Drop    | 144          | 93.06          | 81          | 91.34         |
| Stop    | 418          | 99.29          | 243         | 100.00        |
| Grab    | 110          | 100.00         | 56          | 89.29         |
| Silence | 4631         | 86.18          | 2897        | 83.50         |
| Overall | 8942         | 91.67          | 5258        | 89.04         |

Table 6.10: Posture recognition using histogram-based classifier for Lower Stereo.

higher spatial resolution provides a richer description of the appearance shape, thus using 96 clusters to capture the whole shape spectrum leads to more errors. To verify this, we also carry out an extra experiment using 128 clusters. The recognition on the test set is 94.44%, which is better than current ratio of 89.04%. However, its recognition accuracy is still lower than neural networks.

Many of the failed frames in our test are caused by the variance of the shape of the hand across all users. For example, the Grab posture (two fingers close) of several female subjects is very close to the Push (one finger) gesture of a few male subjects because of the differences in the sizes of the fingers.

### Recognition of Dynamic Gestures

We train our HMM models with stopping state modeling (Section 6.4.3) to model three dynamic gestures: Twist, Twist-Anti and Flip. In testing, we include both positive sequences as well as negative sequences, such as incomplete gesture sequences and other arbitrary sequences. To compare our extended HMMs with traditional HMMs without explicit stopping state modeling, we also carry out training and testing experiments using standard HMMs. Table 6.11 and Table 6.12 shows the results for Higher Stereo.

| <b>Gesture</b> | <b>Sequences</b> | <b>Standard</b> | <b>Extended</b> |
|----------------|------------------|-----------------|-----------------|
| Twist          | 54               | 96.30           | 100.00          |
| Twist-Anti     | 47               | 93.62           | 100.00          |
| Flip           | 48               | 100.00          | 100.00          |
| Overall        | 147              | 96.64           | 100.00          |

Table 6.11: Training accuracy of dynamic gestures using standard HMMs and extended HMMs for Higher Stereo.

In Table 6.13 and Table 6.14, we shows the training and testing results for Lower Stereo.

It can be seen that, compared to standard HMMs, the extended HMMs with stopping state modeling rejects negative sequences quite successfully. This is especially crucial in parsing continuous gestures, because it prevents premature decision on incomplete sequences and thus ensures that the parsing of the following gesture works on the appropriate

| <b>Gesture</b> | <b>Sequences</b> | <b>Standard</b> | <b>Extended</b> |
|----------------|------------------|-----------------|-----------------|
| Twist          | 27               | 81.48           | 85.19           |
| Twist-Anti     | 29               | 93.10           | 93.10           |
| Flip           | 28               | 96.43           | 96.43           |
| Negative       | 578              | 79.58           | 98.79           |
| Overall        | 665              | 81.05           | 97.89           |

Table 6.12: Testing accuracy of dynamic gestures using standard HMMs and extended HMMs for Higher Stereo.

| <b>Gesture</b> | <b>Sequences</b> | <b>Standard</b> | <b>Extended</b> |
|----------------|------------------|-----------------|-----------------|
| Twist          | 53               | 96.23           | 100.00          |
| Twist-Anti     | 45               | 100.00          | 100.00          |
| Flip           | 46               | 100.00          | 100.00          |
| Overall        | 144              | 98.61           | 100.00          |

Table 6.13: Training accuracy of dynamic gestures using standard HMMs and extended HMMs for Lower Stereo.

| <b>Gesture</b> | <b>Sequences</b> | <b>Standard</b> | <b>Extended</b> |
|----------------|------------------|-----------------|-----------------|
| Twist          | 29               | 86.21           | 86.21           |
| Twist-Anti     | 26               | 88.46           | 88.46           |
| Flip           | 28               | 89.29           | 92.86           |
| Negative       | 539              | 84.97           | 98.52           |
| Overall        | 622              | 85.37           | 97.27           |

Table 6.14: Testing accuracy of dynamic gestures using standard HMMs and extended HMMs for Lower Stereo.

video segment.

## Recognition of Parameterized Gestures

We use three-layer hierarchy of SSD tracking scheme to track the 2D translation/rotation of parameterized gestures: Move, Rotate and Resize. The scale factor between neighboring layers is 0.5. Since we have stereo video streams, we implement two parallel trackers (left and right tracker) separately on these two videos and compare their results.

For Higher Stereo, we use a template of  $150 \times 150$  pixels. Table 6.15 shows the tracking results based on per frame and per sequence criteria. These statistics include raw

tracking errors (in pixels), and difference of the tracked parameters (rotation angle and translational amounts) projected to the 4DT screen, which has a resolution of  $1280 \times 1024$ .

| Gest.  | Frame | Error |       | Differences |       |       | Seq. | Differences |       |       |
|--------|-------|-------|-------|-------------|-------|-------|------|-------------|-------|-------|
|        |       | Left  | Right | Angle       | X     | Y     |      | Angle       | X     | Y     |
| Move   | 2735  | 5.50  | 4.84  | 0.025       | 6.32  | 4.67  | 56   | 0.035       | 8.16  | 7.32  |
| Rotate | 2533  | 6.01  | 5.66  | 0.057       | 10.47 | 12.63 | 58   | 0.074       | 11.12 | 14.84 |
| Resize | 2038  | 6.67  | 5.50  | 0.031       | 7.45  | 7.70  | 63   | 0.033       | 9.14  | 8.93  |
| Total  | 7306  | 6.00  | 5.31  | 0.038       | 8.07  | 8.28  | 177  | 0.047       | 9.48  | 10.39 |

Table 6.15: Results of tracking parameterized gestures for Higher Stereo.

It can be seen that, the tracked parameters by the two separate trackers are very close. In the worst case, the difference is about three degree in rotation and 12.63 pixel in translation. In parsing the gesture, we compare the template matching error of the two trackers and choose the parameters of the tracker with less error.

Compared to Move and Resize (which tracks 2D translation), Rotate has greater raw tracking errors and left/right tracker differences. This is partly because when the user rotate his or her hand, more occlusion and shape changes will occur. One solution to this problem is to use extra cameras to disambiguate the occlusion, such as a camera installed above the center of the screen [92].

In Table 6.16, we also show the statistics of average tracked parameters for both left and right trackers.

| Gesture | Left Tracker |       |       | Right Tracker |       |       |
|---------|--------------|-------|-------|---------------|-------|-------|
|         | Angle        | X     | Y     | Angle         | X     | Y     |
| Move    | 0.067        | 56.85 | 33.62 | 0.075         | 57.11 | 39.14 |
| Rotate  | 0.417        | 20.52 | 15.90 | 0.427         | 14.92 | 17.12 |
| Resize  | 0.063        | 66.64 | 45.38 | 0.071         | 60.37 | 52.47 |

Table 6.16: Average amount of parametric changes of parameterized gestures for Higher Stereo.

These results show that, for pure rotation, both trackers correctly report a significant change in rotation angle while a relatively low translation. While for Move and Resize, the rotation is very low and the translation is much higher than Rotate. Recall in

| Gest.  | Frame | Error |       | Differences |       |       | Seq. | Differences |       |       |
|--------|-------|-------|-------|-------------|-------|-------|------|-------------|-------|-------|
|        |       | Left  | Right | Angle       | X     | Y     |      | Angle       | X     | Y     |
| Move   | 1958  | 6.94  | 7.09  | 0.038       | 12.24 | 8.66  | 43   | 0.044       | 12.77 | 13.14 |
| Rotate | 1283  | 7.57  | 8.91  | 0.127       | 16.29 | 13.86 | 34   | 0.148       | 17.42 | 13.14 |
| Resize | 1065  | 6.69  | 7.15  | 0.037       | 12.68 | 11.05 | 40   | 0.048       | 12.01 | 13.50 |
| Total  | 4306  | 7.07  | 7.61  | 0.064       | 13.56 | 10.80 | 117  | 0.076       | 13.86 | 11.82 |

Table 6.17: Results of tracking parameterized gestures for Lower Stereo.

our high-level gesture model discussed in Section 6.4.1, both Rotate and Move can follow a Pick in a valid gesture sentence. The ambiguity is resolved by checking whether Move or Rotate returns a higher confident that is based on the measurement of respective tracked parameters.

For Lower Stereo, we use a template of  $250 \times 250$  pixels. We show the tracking results in Table 6.17 and Table 6.18.

| Gesture | Left Tracker |       |       | Right Tracker |       |       |
|---------|--------------|-------|-------|---------------|-------|-------|
|         | Angle        | X     | Y     | Angle         | X     | Y     |
| Move    | 0.058        | 51.16 | 14.39 | 0.075         | 52.01 | 17.96 |
| Rotate  | 0.289        | 25.19 | 18.39 | 0.329         | 17.14 | 22.25 |
| Resize  | 0.060        | 59.60 | 28.31 | 0.067         | 53.62 | 36.19 |

Table 6.18: Average amount of parametric changes in parameterized gestures for Lower Stereo.

## 6.5.2 Experimental Results of Modeling Composite Gestures

In this experiment, we use supervised learning to learn the high-level gesture model discussed in Section 6.4.1. Then we test this learned model on a separate dataset.

Since we have two approaches to model postures, i.e., neural networks and Bayesian classifiers, we also perform two sets of experiments using different posture models. We will present the result in the remainder of this section.

## Experimental Result Using Neural Networks

For High Stereo, we use over 150 labeled sequences to train the HMM model. We train a similar gesture model for Lower Stereo based on over 170 gesture sequences.

Table 6.19 shows the recognition results on our testing set. When the GWord sequence generated by the greedy algorithm does not match the ground truth sequence, which is obtained by manually labeling, we will declare the recognition to have failed.

| Gesture       | Higher Stereo |        | Lower Stereo |        |
|---------------|---------------|--------|--------------|--------|
|               | Sequences     | Ratio  | Sequences    | Ratio  |
| Pushing       | 35            | 97.14  | 34           | 97.06  |
| Twisting      | 34            | 100.00 | 31           | 96.77  |
| Twisting-Anti | 28            | 96.42  | 27           | 100.00 |
| Dropping      | 29            | 96.55  | 27           | 92.59  |
| Flipping      | 32            | 96.89  | 31           | 96.77  |
| Moving        | 35            | 94.29  | 32           | 96.88  |
| Rotating      | 27            | 92.59  | 29           | 93.10  |
| Stopping      | 33            | 100.00 | 31           | 96.77  |
| Resizing      | 30            | 96.67  | 29           | 100.00 |
| Total         | 283           | 96.47  | 271          | 96.68  |

Table 6.19: Gesture recognition results using neural network processors.

These results show that our high-level gesture model recognizes continuous gesture over a large group of users very accurately. The recognition accuracy is comparable to our first experiment discussed in Section 6.3.3.

Since all our gesture sentences begin with a static gesture. An early mistake in recognizing the starting posture will cause error in the recognition of the rest of the sequence. Thus, the correct recognition of these postures is crucial to the successful parsing of the whole sentence. In fact, many of the failing sentences are caused by incorrect posture recognition in the first stage.

## Experimental Result Using Bayesian Classifiers

In this experiment, we use histogram-based classifiers to recognize postures. Table 6.20 shows the results for both Higher Stereo and Lower Stereo.

| Gesture       | Higher Stereo |       | Lower Stereo |       |
|---------------|---------------|-------|--------------|-------|
|               | Sequences     | Ratio | Sequences    | Ratio |
| Pushing       | 35            | 97.14 | 34           | 85.29 |
| Twisting      | 34            | 85.29 | 31           | 83.87 |
| Twisting-Anti | 28            | 85.71 | 27           | 81.48 |
| Dropping      | 29            | 86.21 | 27           | 85.19 |
| Flipping      | 32            | 90.63 | 31           | 80.65 |
| Moving        | 35            | 91.43 | 32           | 81.25 |
| Rotating      | 27            | 88.89 | 29           | 93.10 |
| Stopping      | 33            | 81.82 | 31           | 90.32 |
| Resizing      | 30            | 96.67 | 29           | 93.10 |
| Total         | 283           | 87.99 | 271          | 85.98 |

Table 6.20: Gesture recognition results using the Bayesian classifier.

It's evident that these results are worse than those using neural networks. Again, most of the errors are caused by the posture classifier. As discussed in Section 6.5.1, the posture recognition accuracy of histogram-based classifiers is lower than neural networks. Thus, when embedded into high-level language model, the system reports lower recognition ratio.

Based on the overall evaluation of the results for both low-level and composite gestures, we conclude that Higher Stereo with neural network-based posture modeling achieves the best performance. This is partly because of the better learning and modeling capability of multi-layer neural networks.

### 6.5.3 User Feedback Regarding Our Gesture Interface

During the experiment, we also ask the subjects to fill in a feedback form giving their opinion of our gesture system.

All subjects agree that our gesture vocabulary is easy to remember and learn. We also asked the subjects about which gestures are not very intuitive and can be improved. Six subjects answered that all gestures are intuitive. Flip received five votes, Rotate and Pick received two votes, and Twist received one. It can be seen that dynamic gestures (Flip, Rotate and Twist) have lower acceptance. This is reasonable considering that in daily life,

we carry out such tasks (e.g., picking a pen above a desk) without a clear form or rules. However, overall, the subjects are satisfied with our gesture vocabulary.

Another aspect of gesture system is its convenience for prolonged usage. This aspect is often ignored in current research of designing new user interfaces. For each subject, our experiment lasts about thirty minutes. We asked the subjects about the degree of fatigue using the gesture system, compared to using a mouse in a traditional GUI. Eight subjects (50%) think that the gesture system is comparable to a mouse. Six subjects think they feel more tired and two subjects feel less tired than using a mouse. One reason is that all our subjects are daily users of traditional GUIs, so switching to a gesture interface requires practice and adaptation to work efficiently and fluently. Also the amount of physical movement of the hands on our flat screen (about  $40cm \times 30cm$ ) is much larger than on a small mouse pad. Thus, continuously moving the hands on such a large physical space can cause more fatigue. Although this might limit the application of gesture-based interfaces for normal human being and for such relatively simple tasks, there are still many situation where such interfaces are preferred. For example, people with visual disabilities can use gestures to communicate with computers.

Our last investigation is about the overall performance of such a gesture-based interface. We asked the subjects how they feel about the interface compared to traditional GUIs with a mouse. Seven subjects (44%) think our gesture-based interface is more convenient and comfortable, while seven think they are comparable. Only two subjects think our system are more awkward because they have to first learn how to use. But they agree that, after spending a short amount of time practicing, the system is comparable to traditional GUIs with a mouse. Overall fourteen subjects (88%) consider using natural gestures is comparable or more convenient than traditional mouse-based GUIs.

## 6.6 Conclusions

We have presented an approach to recognize composite gestures. The composite gestures consist of three different types of low-level units (GWords): static, posture-based primitives; non-parametric dynamic gestures; and parametric, dynamic gestures. We con-

struct a coherent model by combining the GWords and a high-level language model in a probabilistic framework which is defined as an HMM. We have proposed unsupervised and supervised learning algorithms; our results show that even with a random initialization, the gesture model can learn the underlying gesture language model. By combining the model and the greedy inference algorithm, our method can model gestures composed of GWords with heterogeneous observation.

Our approach allows the inference of composite gestures as paths through the HMM and uses the high-level constraints to guide the recognition of composite gestures. However, the proposed greedy inference algorithm will make locally optimal decisions since it is operating online.

We make two experiments to evaluate our gesture modeling techniques. The first vocabulary consists of 10 low-level GWords and 6 composite gestures. The second experiment involves 16 users, 14 GWords and 9 composite gestures. Our experimental results show the robustness and efficiency of our approaches in modeling both low-level gestures and high-level composite gestures across a relatively large group of users. This large-scale experiment shows the promising of integrating our vision-based gesture modeling techniques into general practical applications.

In our current implementation, for both low-level gestures and the high level gesture model, we train common models for all the subjects. As we mentioned, many of the recognition mistakes are caused by the variances among different users. These variances include disparity in shape and motion dynamics. We expect that, using a user-specific model or a weighted sum of several groups of models can reduce the recognition error caused by such discrepancy among a large group of users. We intend to address these problems in future research and compare the results with our current implementation.

## Chapter 7

# Conclusions

In this dissertation, we have presented several techniques for applying computer vision to natural and intelligent human-computer interaction. An overview of the literature reveals many open problems in both multi-modal interfaces and vision-based HCI. We first propose a novel approach to integrating visual tracking into haptics systems. Traditional haptic environments require the user to be attached to the haptic device at all times, even though force feedback is not always being rendered. We designed and implemented an augmented reality system, called VisHap, that uses visual tracking to seamlessly integrate force feedback with tactile feedback to generate a “complete” haptic experience. Such a multi-modal framework allows the user to interact with combinations of virtual and real objects naturally, thereby combining active and passive haptics. The flexibility and extensibility of our framework is promising in that it supports many interaction modes and allows further integration with other augmented reality systems.

We also presented our novel vision-based HCI methodology, which is called Visual Interaction Cues (VICs). In VICs, we approach gesture recognition without globally tracking and modeling the user. Instead, we use techniques of component mapping to limit the analysis to a localized space in the projection in video cameras. In the local regions of the video images, gesture recognition is solved by modeling the spatio-temporal pattern of visual cues that correspond to gestures. We show that the paradigm is applicable in both conventional 2D interface settings and unconventional 3D virtual and augmented reality

settings.

One of the biggest problems in vision-based gesture analysis is efficient data collection. We designed and implemented the 4DT system which makes use of system calibration and skin modeling to allow robust capture of hand motion. We also proposed an efficient feature extraction scheme that describes the appearance and motion of the hand. Based on this scheme, we have been able to collect many visual datasets for gesture analysis. We have designed and implemented various gesture modeling approaches, such as neural networks, Hidden Markov Models and Bayesian classifiers. Our experiments show that our motion capture scheme and feature extraction scheme allows efficient and robust gesture modeling.

Since gestures are in essence a language, with individual low-level gestures analogous to a word in conventional languages, a high-level gesture model is essential for robust and efficient recognition of continuous gesture. To that end, we have proposed a high-level gesture model that integrates low-level gestures into a single, coherent probabilistic framework. In our model, each low-level gesture is called a GWord and a composite gesture is a sentence composed of a sequence of GWords. The parameters of the model can be learned using supervised or unsupervised techniques. A greedy inference algorithm allows online recognition of continuous gestures. We have designed a large-scale gesture experiment that involves sixteen subjects and fourteen GWords. The experiment shows the robustness and efficacy of our system in modeling a relative large gesture vocabulary involving many users. Most of the users also consider our gesture system is comparable or more natural and comfortable than traditional user interface with a mouse.

## 7.1 Future Work

In the future, we intend to advance our research into the following directions:

1. **Vision and Haptics Interface:** In Chapter 3, we have presented a multimodal interaction system that combines vision and haptics to improve the fidelity of haptic experiences. We intend to incorporate such tools as HMDs to enhance the visual feedback to the user. Also more interaction modes with virtual and real objects need to be explored. We intend to develop more applications and carry out human factors

experiments to evaluate this multimodal interface.

2. **3D Interaction:** We have present VICs as a novel paradigm for general vision-based interaction. We have also discussed several interaction models, including 2D mirror and 3D-2D projection. We plan to investigate the efficacy of VICs in 2.5D and 3D augmented reality. This will allow wider applications of vision-based interaction, including 3D virtual environments and medical applications.
3. **Visual Cues Extraction:** On the 4DT system, we have discussed techniques for efficient hand segmentation and motion capture. Although our 3D appearance and motion features can effectively model both static and dynamic gestures, we intend to investigate more robust ways to locally characterize the hand motion. For examples, it is ideal if the features are robust against the variances of the hand size and shape across users, variation of the camera configuration, and the changes of lighting conditions.
4. **Gesture Modeling:** In this dissertation, we have discussed various approaches for modeling both low-level and high-level gestures. We have also proposed a greedy algorithm to carry out inference on a high-level gesture model. We intend to investigate more precise and efficient methods to recognize continuous gestures. Our current gesture vocabulary consists of nine high-level gesture sentences. We also want to test our model on a larger gesture set. One possible application is to automatically recognize continuous sign language.

# Bibliography

- [1] D. Alexander and B. Buxton. Statistical Modeling of Colour Data. *Int. Journal of Computer Vision*, 44(2):87–109, 2001.
- [2] V. Athitsos and S. Sclaroff. Estimating 3D Hand Pose from a Cluttered Image. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 432–439, 2003.
- [3] R. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(11):1–38, 1997.
- [4] R. Azuma, Y. Baillet, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.
- [5] S. Basu and A. Pentland. Motion Regularization for Model-based Head Tracking. In *Proc. Int. Conf. Pattern Recognition*, 1996.
- [6] M. Beaudouin-Lafon. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. *Proc. of CHI*, 2(1):446–453, 2000.
- [7] D. Becker and A. Pentland. Using a Virtual Environment to Teach Cancer Patients T'ai Chi, Relaxation, and Self-Imagery. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 1996.
- [8] S. Belongie, J. Malik, and J. Puzicha. Matching Shapes. In *Proc. Int. Conf. Computer Vision*, pages 454–463, 2001.

- [9] J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, Technical Report, International Computer Science Institute, 1997.
- [10] O. Bimber, B. Frohlich, D. Schmalstieg, and L. Encarnacao. The Virtual Showcase. *IEEE Computer Graphics and Applications*, 21(6):48–55, 2001.
- [11] M. Black and Y. Yacoob. Tracking and Recognizing Rigid and Non-rigid Facial Motions Using Local Parametric Models of Image Motion. *Int. Journal of Computer Vision*, 25(1):23–48, 1997.
- [12] M. Blattner and E. Gliner. Multimodal Integration. *IEEE Multimedia*, 3(4):14–24, 1996.
- [13] A. Bobick and J. Davis. The Recognition of Human Movement Using Temporal Templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.
- [14] A. Bobick and A. Wilson. A State-based Technique for the Summarization of Recognition of Gesture. In *Proc. Int. Conf. Computer Vision*, pages 382–388, 1995.
- [15] A. Bobick and A. Wilson. A State-based Approach to the Representation and Recognition of Gesture. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, 1997.
- [16] R. Bolt. Put That Three: Voice and Gesture at the Graphic Interface. *Proc. ACM SIGGRAPH*, pages 33–35, 1980.
- [17] J. Bouquet. Camera calibration toolbox for matlab.
- [18] G. Bradski. Computer Vision Face Tracking for Use in a Perceptual User Interface. *Intel Technology Journal*, 1998.
- [19] M. Brand, N. Oliver, and A. Pentland. Coupled Hidden Markov Models for Complex Action Recognition. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 994–999, 1997.

- [20] C. Bregler. Learning and Recognizing Human Dynamics in Video Sequences. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 568–574, 1997.
- [21] C. Bregler and J. Malik. Tracking People with Twists and Exponential Maps. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 8–15, 1998.
- [22] D. Burschka, G. Ye, J. Corso, and G. Hager. A Practical Approach for Integrating Vision-Based Methods Into Interactive 2D/3D Applications. Technical report, CIRL Lab, Johns Hopkins University, 2005.
- [23] Q. Cai and J. Aggarwal. Human Motion Analysis: A Review. *Journal of Computer Vision and Image Understanding*, 73(3):428–440, 1999.
- [24] J. Cassell. *A Framework for Gesture Generation and Interpretation*. Computer Vision in Human-Machine Interaction, 1998.
- [25] M. Cavusoglu, D. Feygin, and F. Tendick. A Critical Study of The Mechanical and Electrical Properties of The Phantom Haptic Interface and Improvements for High Performance Control. *Presence*, 11(5):555–568, 2002.
- [26] Y. Cheung. A Competitive and Cooperative Learning Approach to Robust Data Clustering. Technical report, Dep. Computer Science, Hong Kong Baptist University, 2003.
- [27] J. Corso, D. Burschka, and G. Hager. Direct Plane Tracking in Stereo Image for Mobile Navigation. In *Proc. Int. Conf. on Robotics and Automation*, volume 875-880, 2003.
- [28] J. Corso, D. Burschka, and G. Hager. The 4DT: Unencumbered HCI With VICs. In *Proc. IEEE CVPR Workshop on Human-Computer Interaction*, 2003.
- [29] J. Corso, G. Ye, D. Burschka, and G. Hager. Software Systems for Vision-Based Spatial Interaction. In *Proc. 2002 Workshop on Intelligent Human Augmentation and Virtual Environments*, pages D–26 and D–56, 2002.

- [30] J. Corso, G. Ye, and G. Hager. Analysis of Multi-Modal Gestures with a Coherent Probabilistic Graphical Model. *Virtual Reality*, 2005.
- [31] J. Crowley, F. Berard, and J. Coutaz. Finger Tracking as an Input Device for Augmented Reality. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 195–200, 1995.
- [32] C. Cruz-Neira, D. Sandin, and T. Defanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. *Proc. ACM SIGGRAPH*, pages 135–143, 1993.
- [33] Y. Cui and J. Weng. Hand Sign Recognition from Intensity Image Sequences with Complex Background. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 88–93, 1996.
- [34] A. Van Dam. Post-WIMP User Interfaces. *Communications of the ACM*, 40(2):63–67, 1997.
- [35] J. Davis and A. Bobick. The Representation and Recognition of Action Using Temporal Templates. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 928–934, 1997.
- [36] J. Davis and M. Shah. Recognizing Hand Gestures. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, volume 1, pages 331–340, 1994.
- [37] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, 2000.
- [38] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, Inc, 2001.
- [39] A. Elgammal, V. Shet, Y. Yacoob, and L.S. Davis. Learning Dynamics for Exemplar-based Gesture Recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 571–578, 2003.
- [40] O. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

- [41] D. Forsyth and J. Ponce. *Computer Vision*. Prentice Hall, 2003.
- [42] H. Fuchs, S. Pizer, L. Tsai, and S. Bloomberg. Adding a True 3-D Display to a Raster Graphics System. *IEEE Computer Graphics and Applications*, 2(7):73–78, 1982.
- [43] A. Galata, N. Johnson, and D. Hogg. Learning Variable-Length Markov Models of Behavior. *Journal of Computer Vision and Image Understanding*, 81(3):398–413, 2001.
- [44] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley Professional, 1995.
- [45] W. Gao, G. Fang, D. Zhao, and Y. Chen. Transition Movement Models for Large Vocabulary Continuous Sign Language Recognition. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 553–558, 2004.
- [46] D. Gavrilu. The Visual Analysis of Human Movement: A Survey. *Journal of Computer Vision and Image Understanding*, 73(1):82–98, 1999.
- [47] D.M. Gavrilu and L.S. Davis. Towards 3-d Model-based Tracking and Recognition of Human Movement. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 1995.
- [48] T. Gevers. Color Based Object Recognition. *Pattern Recognition*, 32(3):453–464, 1999.
- [49] T. Gevers. Robust Histogram Construction From Color Invariants. In *Proc. Int. Conf. Computer Vision*, pages 615–620, ICCV 2001.
- [50] H. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan. Multi-Modal System for Locating Heads and Faces. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 88–93, 1996.
- [51] E. Hadjidemetriou, M. Grossberg, and S.K. Nayar. Histogram Preserving Image Transformations. *Int. Journal of Computer Vision*, 45(1):5–23, 2001.

- [52] G. Hager and P. Belhumeur. Efficient Region Tracking with Parametric Models of Geometry and Illumination. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(10):1125–1139, 1998.
- [53] G. Hager and K. Toyama. XVision: A Portable Substrate for Real-Time Vision Applications. *Journal of Computer Vision and Image Understanding*, 69(1):23–37, 1996.
- [54] C. Hardenberg and F. Berard. Bare-Hand Human-Computer Interaction. In *Proc. Workshop on Perceptive User Interfaces*, 2001.
- [55] J. Hare, M. Karam, P. Leus, and M. Schraefel. iGesture: A Platform for Investigating Multimodal, Multimedia Gesture-based Interactions. *ACM Multimedia*, 2004.
- [56] V. Hayward and M. Cruz-Hernandez. Tactile Display Device Using Distributed Lateral Skin Stretch. In *Proc. 8th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, volume 2, pages 1309–1314, 2000.
- [57] J. Heikkila and O. Silven. A Four-step Camera Calibration Procedure with Implicit Image Correction. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.
- [58] P. Hong, M. Turk, and T. Huang. Gesture Modeling and Recognition Using Finite State Machines. In *Proc. IEEE Int. Conf. Gesture Recognition*, 2000.
- [59] B. Horn, H. Hilden, and S. Negahdaripour. Closed Form Solution of Absolute Orientation Using Orthonormal Matrices. *Journal of the Optical Society*, 5(7):1127–1135, 1988.
- [60] T. Horprasert, D. Harwood, and L. Davis. A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection. In *Proc. ICCV FRAME-RATE Workshop*, 1999.
- [61] T. Horprasert, D. Harwood, and L. Davis. A Robust Background Subtraction and Shadow Detection. In *Proc. Asian Conf. Computer Vision*, 2000.

- [62] B. Insko, M. Meehan, M. Whitton, and F. Brooks. Passive Haptics Significantly Enhances Virtual Environments. Technical Report 0, Technique Report, University of North Carolina at Chapel Hill, 2001.
- [63] Y. Ivanov and A. Bobick. Recognition of Visual Activities and Interactions by Stochastic Parsing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.
- [64] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1999.
- [65] J. Johnson, T. Roberts, W. Verpland, D. Smith, C. Irby, and M. Beard and K. Mackey. The Xerox Star: A Retrospective. *Computer Graphics*, 22(9):11–26, 1989.
- [66] M. Johnston and S. Bangalore. Finite-state Multimodal Parsing and Understanding. *Proceedings of COLING*, 2000.
- [67] M. Johnston and S. Bangalore. Finite-state Methods for Multimodal Parsing and Integration. In *Finite State Methods in Natural Language Processing*, 2001.
- [68] M. Jones and J. Rehg. Statistical Color Models with Application to Skin Detection. *Int. Journal of Computer Vision*, 46(1):81–96, 2002.
- [69] K. Kahol, P. Tripathi, and S. Panchanathan. Automated Gesture Segmentation From Dance Sequences. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 883–888, 2004.
- [70] A. Kendon. *Current Issues in the Study of Gesture*. The Biological Foundations of Gestures: Motor and Semiotic Aspects, J.-L. Nespoulous, P. Peron, and A. R. Lecours, Eds., Lawrence Erlbaum Assoc., 1986.
- [71] R. Kjeldsen, A. Levas, and C. Pinhanez. Dynamically Reconfigurable Visioin-Based User Interfaces. *Machine Vision and Applications*, 16(1):6–12, 2004.
- [72] H. Koike, Y. Sato, Y. Kobayashi, H. Tobita, and M. Kobayashi. Interactive TextBook and Interactive Venn Diagram:; Natural and Intuitive Interfaces on Augmented Desk

- System. *SIGGHI Conference on Human Factors in Computing Systems*, pages 121–128, 2000.
- [73] R. Lau, G. Flammia, C. Pao, and V. Zue. WebGalaxy-Intergrating Spoken Language and Hypertext Navigation. In *Proc. Eurospeech*, pages 883–886, 1997.
- [74] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.
- [75] C. Lu, G. Hager, and E. Mjolsness. Fast and Globally Convergent Pose Estimation from Video Images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000.
- [76] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003.
- [77] P. Maes, T. Darrell, B. Blumberg, and A. Pentland. The ALIVE System: Wireless, Full-Body Interaction with Autonomous Agents. *MultSys*, 5(2):105–112, 1997.
- [78] S. Malassiotis, N. Aifanti, and M. Strintzis. A Gesture Recognition System Using 3D Data. In *Proc. First Int. Symposium on 3D Data Processing Visualization and Transmission*, pages 190–193, 2002.
- [79] D. Marr. *Vision*. W. H. Freeman and Company, 1982.
- [80] T. Massie and J. Salisbury. The Phantom Haptic Interface: A Device for Probing Virtual Objects. In *Proc. Symposium on Haptic Interfaces for Virtual Environment and Teleoperator*, 1994.
- [81] C. McDonald, G. Roth, and S. Marsh. Red-Handed: Collaborative Gesture Interaction with a Projection Table. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 773–778, 2004.
- [82] S. Mckenna and K. Morrison. A Comparison of Skin History and Trajectory-Based Representation Schemes for the Recognition of User-Specific Gestures. *Pattern Recognition*, 37:999–1009, 2004.

- [83] D. Minnen, I. Essa, and T. Starner. Expectation Grammars: Leveraging High-Level Expectations for Activity Recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 626–632, 2003.
- [84] T. Moran, E. Saund, W. Melle, A. Gujar, K. Fishkin, and B. Harrison. Design and Technology for Collaborage: Collaborative Collages of Information on Physical Walls. In *Proc. ACM Symposium on User Interface Software an Technology*, 1999.
- [85] G. Mori and J. Malik. Estimating Human Body Configurations using Shape Context Matching. In *Proc. European Conf. Computer Vision*, pages 666–680, 2002.
- [86] T. Mori, Y. Segawa, M. Shimosaka, and T. Sato. Hierarchical Recognition of Daily Human Actions Based on Continuous Hidden Markov Models. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 779–784, 2004.
- [87] K. Murphy. A Brief Introduction to Graphical Models and Bayesian Networks. Technical report, University of Brithish Columbia, 1998.
- [88] C. Myers and L. Rabiner. A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected Word Recognition. *The Bell System Technical Journal*, 60(7):1389–1409, 1981.
- [89] J. Nespoulous and A. Lecours. *Gestures: Nature and Function*. The Biological Foundations of Gestures: Motor and Semiotic Aspects, J.-L. Nespoulous, P. Peron, and A. R. Lecours, Eds., Lawrence Erlbaum Assoc., 1986.
- [90] K. Nickel and R. Stiefelhagen. Pointing Gesture Recognition based on 3D-Tracking of Face, Hands and Head Orientation. In *Proc. Workshop on Perceptive User Interfaces*, pages 140–146, 2003.
- [91] Z. Obrenovic and D. Starcevic. Modeling Multimodal Human-Computer Interaction. *IEEE Computer*, 37(9):65–72, 2004.
- [92] K. Oka, Y. Sato, and H. Koike. Real-Time Fingertip Tracking and Gesture Recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, 2002.

- [93] M. Ostendorf, V. Digalakis, and O. Kimball. From HMMs to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition. *IEEE Trans. Speech and Audio Processing*, 4(5):360–378, 1996.
- [94] D. Pai. Multisensory Interaction: Real and Virtual. *Proc. Int. Symposium on Robotics Research*, 2003.
- [95] V. Parameswaran and R. Chellappa. View Invariants for Human Action Recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 613–619, 2003.
- [96] V. Pavlovic, R. Sharma, and T. Huang. Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.
- [97] A. Pentland and A. Liu. Modeling and Prediction of Human Behavior. *Neural Computation*, 11(1):229–242, 1999.
- [98] S. Phung, A. Bouzerdoum, and D. Chai. Skin Segmentation Using Color Pixel Classification: Analysis and Comparison. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(1):148–154, 2005.
- [99] F. Quek. Unencumbered Gesture Interaction. *IEEE Multimedia*, 3(3):36–47, 1996.
- [100] L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [101] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.
- [102] A. Ramamoorthy, N. Vaswani, S. Chaudhury, and S. Banerjee. Recognition of Dynamic Hand Gestures. *Pattern Recognition*, 36:2069–2081, 2003.
- [103] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The Office of the Future: A Unified Approach to Image-based Modeling and Spatially Immersive Displays. In *Proc. ACM SIGGRAPH*, 1998.

- [104] J. Rehg and T. Kanada. Visual Tracking of High DOF Articulated Structure: An Application to Human Hand Tracking. In *Proc. European Conf. Computer Vision*, pages 35–46, 1994.
- [105] J. Rehg and T. Kanade. DigitEyes: Vision-Based Human Hand Tracking for Human-Computer Interaction. In *Proc. IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 16–22, 1994.
- [106] J. Rehg and T. Kanade. Model-based Tracking of Self-Occluding Articulated Objects. In *Proc. Int. Conf. Computer Vision*, pages 612–617, 1995.
- [107] S. Roweis and Z. Ghahramani. A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2):305–345, 1999.
- [108] M. Salada, J. Colgate, M. Lee, and P. Vishton. Fingertip Haptics: A Novel Direction in Haptic Display. In *Proc. 8th Mechatronics Forum International Conference*, pages 1211–1200, 2002.
- [109] M. Salada, J. Colgate, M. Lee, and P. Vishton. Validating a Novel Approach to Rendering Fingertip Contact Sensations. In *IEEE Virtual Reality Haptics Symposium*, pages 217–224, 2002.
- [110] C. Schmandt. Spatial Input/Display Correspondence in a Stereoscopic Graphic Workstation. *Computer Graphics*, 17(3):253–261, 1983.
- [111] J. Segen and S. Kumar. Fast and Accurate 3D Gesture Recognition Interface. In *Proc. Int. Conf. Pattern Recognition*, 1998.
- [112] J. Segen and S. Kumar. Gesture VR: Vision-based 3D Hand Interface for Spatial Interaction. In *Proc. ACM Int. Conf. Multimedia*, pages 455–464, 1998.
- [113] J. Segen and S. Kumar. Shadow Gestures: 3D Hand Pose Estimation Using a Single Camera. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 479–485, 1999.

- [114] R. Sharma, T. Huang, V. Pavlovic, Y. Zhao, Z. Lo, S. Chu, K. Schulten, A. Dalke, J. Phillips, M. Zeller, and W. Humphrey. Speech/Gesture Interface to a Visual Computing Environment for Molecular Biologists. *IEEE Computer Graphics and Applications*, 20(2):29–37, 2000.
- [115] Y. Shi, Y. Huang, D. Minnen, A. Bobick, and I. Essa. Propagation Networks for Recognition of Partially Ordered Sequential Action. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 862–869, 2004.
- [116] M. Shin, L. Tsap, and D. Goldgof. Gesture Recognition Using Bezier Curves for Visualization Navigation from Registered 3-D Data. *Pattern Recognition*, 37(5):1011–1024, 2004.
- [117] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983.
- [118] J. Siskind. Visual Event Perception. In *Proceedings of the NEC Research Symposium*, 1998.
- [119] P. Smyth. Belief Networks, Hidden Markov Models, And Markov Random Fields: A Unifying View. *Pattern Recognition Letters*, 18:1261–1268, 1997.
- [120] Q. Stafford-Fraser and P. Robinson. Brightboard: a Video-augmented Environment Papers: Virtual and Computer-augmented Environments. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 134–141, 1996.
- [121] T. Starner, B. Leibe, D. Minnen, T. Westyn, A. Hurst, and J. Weeks. The Perceptive Workbench: Computer-Vision-Based Gesture Tracking, Object Tracking, and 3D Reconstruction for Augmented Desks. *Machine Vision and Applications*, 14(1):59–71, 2003.
- [122] T. Starner, J. Weaver, and A.P. Pentland. Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

- [123] I. Sutherland. A Head-Mounted Three Dimensional Display. *Proceedings of the Fall Joint Computer Conference*, pages 757–764, 1968.
- [124] M. Swain. Color Indexing. *Int. Journal of Computer Vision*, 7(1):11–32, 1991.
- [125] C. Tomasi, S. Petrov, and A. Sastry. 3D Tracking = Classification + Interpolation. In *Proc. Int. Conf. Computer Vision*, pages 1441–1448, 2003.
- [126] J. Triesch and C. Malsburg. Robust Classification of Hand Postures against Complex Backgrounds. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 170–175, 1996.
- [127] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
- [128] M. Turk and G. Robertson. Perceptual User Interfaces. *Communications of the ACM*, 43(3):33–34, 2000.
- [129] A. Utsumi and J. Ohya. Multiple-Hand-Gesture Tracking Using Multiple Cameras. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 473–478, 1999.
- [130] R. Vertegaal. Attentive User Interfaces. *Communications of the ACM*, 46(3):31–33, 2003.
- [131] E. Vidal, F. Thollard, C. Higuera, F. Casacuberta, and R. Carrasco. Probabilistic Finite-State Machines-Part I. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(7):1013–1025, 2005.
- [132] E. Vidal, F. Thollard, C. Higuera, F. Casacuberta, and R. Carrasco. Probabilistic Finite-State Machines-Part II. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(7):1026–1039, 2005.
- [133] P. Viola, M. Jones, and D. Snow. Detecting Pedestrains Using Patterns of Motion and Appearance. In *Proc. Int. Conf. Computer Vision*, pages 734–741, 2003.

- [134] C. Vogler and D. Metaxas. ASL Recognition Based on a Coupling Between HMMs and 3D Motion Analysis. In *Proc. Int. Conf. Computer Vision*, pages 363–369, 1998.
- [135] C. Wagner, S. Lederman, and R. Howe. A Tactile Shape Display Using RC Servomotors. In *Proc. 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator*, pages 354–355, 2002.
- [136] I. Weiss and M. Ray. Model-based Recognition of 3d Objects from Single Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):116–128, 2001.
- [137] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical report, Technique Report, University of North Carolina at Chapel Hill, 2004.
- [138] P. Wellner. Interacting with Paper on the DigitalDesk. *Communications of the ACM*, 36(7):87–96, 1993.
- [139] A. Wilson and A. Bobick. Parametric Hidden Markov Models for Gesture Recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.
- [140] A. Wilson and N. Oliver. GWindows: Robust Stereo Vision for Gesture-Based . In *Proc. Workshop on Perceptive User Interfaces*, pages 211–218, 2003.
- [141] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time Tracking of the Human Body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–784, 1997.
- [142] Y. Wu, G. Hua, and T. Yu. Tracking Articulated Body by Dynamic Markov Network. In *Proc. Int. Conf. Computer Vision*, pages 1094–1101, 2003.
- [143] Y. Wu and T. Huang. Capturing Articulated Human Hand Motion. In *Proc. Int. Conf. Computer Vision*, pages 606–611, 1999.
- [144] Y. Wu and T. Huang. View-independent Recognition of Hand Postures. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 88–94, 2000.

- [145] Y. Wu and T. Huang. Hand Modeling, Analysis, and Recognition. *IEEE Signal Processing Magazine*, 18(3):51–60, 2001.
- [146] Y. Wu, J. Lin, and T. Huang. Capturing Natural Hand Articulation. In *Proc. Int. Conf. Computer Vision*, volume 2, pages 426–432, 2001.
- [147] L. Xu, A. Krzyzak, and E. Oja. Rival Penalized Competitive Learning for Clustering Analysis, RBF net and Curve. *IEEE Trans. Neural Network*, 4:636–649, 1993.
- [148] M. Yamamoto, A. Sato, and S. Kawada. Incremental Tracking of Human Actions from Multiple Views. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 2–7, 1998.
- [149] J. Yamato, J. Ohya, and K. Ishii. Recognizing Human Actions in Time-sequential Images Using Hidden Markov Model. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 379–385, 1992.
- [150] G. Ye, J. Corso, D. Burschka, and G.D. Hager. VICs: A Modular Vision-Based HCI Framework. In *Proceedings of International Conference on Computer Vision Systems*, pages 257–267, 2003.
- [151] G. Ye, J. Corso, and G. Hager. Gesture Recognition Using 3D Appearance and Motion Features. In *Proc. CVPR Workshop on Real-Time Vision for Human-Computer Interaction*, 2004.
- [152] G. Ye, J. Corso, and G. Hager. VICs: A Modular HCI Framework Using Spatio-Temporal Dynamics. *Machine Vision and Applications*, 2004.
- [153] G. Ye, J. Corso, and G. Hager. *Real-Time Vision for Human-Computer Interaction*, chapter Visual Modeling of Dynamic Gestures Using 3D Appearance and Motion Features. Springer-Verlag, 2005.
- [154] G. Ye, J. Corso, G. Hager, and A. Okamura. VisHap: Augmented Reality Combining Haptics and Vision. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 3425–3431, 2003.

- [155] G. Ye and G. Hager. Apperance-Based Visual Interaction. Technical report, Department of Computer Science, Johns Hopkins University, 2002.
- [156] P. Yin, I. Essa, and J. Rehg. Asymmetrically Boosted HMM for Speech Reading. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 755–671, 2004.
- [157] Y. Yokilohji, J. Kinoshita, and T. Yoshikawa. Path Planning for Encountered-type Haptic Devices That Render Multiple Objects in 3D Space. In *Proc. IEEE Virtual Reality*, pages 271–278, 2001.
- [158] Y. Yokokohji. WYSIWYF Display: A Visual/Haptic Interface to Virtual Environment. *Presence*, 8(4):412–434, 1999.
- [159] T. Yoshikawa and A. Nagura. A Touch/force Display System for Haptic Interface. *Presence: Teleoperators and Virtual Environments*, 10(2):225–235, 2001.
- [160] R. Zeleznik, K. Herndon, and J. Hughes. SKETCH: An Interface for Sketching 3D Scenes. In *Proc. 23rd Annual Conf. Computer Graphics and Interactive Techniques*, pages 163–170, 1996.
- [161] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
- [162] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer. Visual Panel: Virtual Mouse Keyboard and 3D Controller with an Ordinary Piece of Paper. In *Proc. Workshop on Perceptive User Interfaces*, 2001.
- [163] H. Zhou, D. Lin, and T. Huang. Static Hand Postures Recognition based on Local Orientation Histogram Feature Distribution Model. In *Proc. CVPR Workshop on Real-Time Vision for Human-Computer Interaction*, 2004.

# Vita

## EDUCATION

**Ph.D. Candidate in Computer Science**, The Johns Hopkins University, 2000-  
Current

**M.S.E. in Computer Science**, The Johns Hopkins University, 2002

**B.E in Computer Science**, *Cum Laude*, Tsinghua University, 1998.

Thesis: Face Detection under Complex Background

## PROFESSIONAL EXPERIENCES

**Research Assistant** - Dr. Gregory Hager, CIRL Lab, Summer 2001-Current

Project: Participated in the NSF VICs project. Investigated efficient and robust vision-based techniques to model dynamic gestures as well as composite gestures for human-computer interaction.

1. Proposed a novel and efficient shape/motion descriptor to capture hand motion.
2. Investigated such methods as neural network, extended HMM, finite-state machines to model dynamic gestures.
3. Proposed efficient algorithm to model composite gestures using graphical model and HMM. Composite gestures are sequences of heterogeneous gestures including static, dynamic, and parameterized gestures.

4. Designed and implemented a flexible vision-based HCI platform. Carried out a human factor experiment on this platform. The experiment involved sixteen subjects and twelve gestures.

**Research Assistant** - Dr. Allison Okamura, Haptic Exploration Lab, Summer 2002- Spring 2003

Project: Designing and implementing the VisHap augmented reality system, which uses visual tracking to seamlessly integrate force feedback with tactile feedback to generate a “complete” haptic experience.

**Research Assistant** - Dr. Guangyou Xu, Computer Vision Lab, Tsinghua University, 1997 - 2000

Project: Participating in face detection, recognition and identification project. Designing and implementing a face identification system with a database of 500 people. The system consists of such key modules such as constrained camera rectification, face detection, facial features extraction, and efficient matching.

**Paper Reviewer:** Journal of the Image and Vision Computing, 2004, IEEE CVPR, 2003.

**Software Project Manager** - Wingon Software Inc., 1999.

Project: Lead a team of five engineers to design and implement a MIS system serving the municipal administrative offices of Beijing. Designed the entire database and system framework.

**Software Engineer** - Bluebox Inc., 1998-1999.

Project: Lead a team to design and implement a business management system. PowerBuilder, SQL, MS Visual Studio and InstallShield were used

**Software Engineer** - NaSoft Inc., 1998.

Project: Joined a team of fifteen engineers to implement the Bank Credit Registration System. This distributed system mainly consisted of such modules as server database management, bank operator interface, Internet access support, and middleware. Such tools as SQL, Sybase PowerBuilder, Visual Basic and

HTML were used during development. The system is now serving the People's Bank of China and its over 400 branches and affiliates.

## TEACHING EXPERIENCES

**Teaching Assistant**, Intermediate Java Programming, Dr. Dwight Wilson, Fall 2001. Duties included holding weekly office hours, grading homework and projects.

**Teaching Assistant**, Algorithms, Dr. Baruch Awerbuch, Spring 2001. Duties included holding weekly office hours, providing weekly assignments and two exams, preparing and posting solutions for homework assignments, grading assignments and exams, taking and posting lectures notes, and holding review sessions.

**Teaching Assistant**, Computer Vision, Dr. Gregory Hager, Fall 2000. Duties included holding weekly office hours, grading homework assignments.

## HONORS AND AWARDS

**Excellent Graduate Award**, Tsinghua University, 1998

**Excellent Student Scholarship**, Tsinghua University, 1994,

**Zheng Geru Scholarship**, Tsinghua University, 1996 and 1997.

## PUBLICATIONS

### Journal Articles and Book Chapters

1. J. Corso, G. Ye, and G. Hager. Analysis of Multi-Modal Gestures with a Coherent Probabilistic Graphical Model. *Virtual Reality*, 2005.
2. G. Ye, J. Corso and G. Hager, Visual Modeling of Dynamic Gestures Using 3D Appearance and Motion Features, in *Real-Time Vision for Human-Computer*

Interaction, edited by B. Kisacanin, V. Pavlovic and T.S. Huang, Springer-Verlag, 2005.

3. G. Ye, J. Corso, D. Burschka, and G. Hager. VICs: A Modular HCI Framework Using Spatio-temporal Dynamics. *Machine Vision and Applications*, 16(1), pages 13-20, 2004.

### **Conference Publications and Workshops**

1. G. Ye and G. Hager. Robust Modeling of Heterogeneous Gestures Using Localized Parsers, IBM Symposium on User Interface and Signal Processing Technologies, 2005.
2. G. Ye, J. Corso, G. Hager, and A. Okamura. VisHap: Augmented Reality Combining Haptics and Vision. In *Proceedings of 2003 IEEE International Conference on Systems, Man & Cybernetics*, October 2003.
3. G. Ye, J. Corso, D. Burschka, and G. Hager. VICs: A Modular Vision-Based HCI Framework. In *Proceedings of 3rd International Conference on Computer Vision Systems (ICVS)*, April 2003. Pages 257-267.
4. G. Ye, J. Corso and G. Hager. Gesture Recognition Using 3D Appearance and Motion Features. In *Proceedings of CVPR 2004 Workshop on Real-time Vision for Human-Computer Interaction*, 2004.
5. J. Corso, G. Ye, D. Burschka, and G. Hager. Software Systems for Vision-Based Spatial Interaction. In *Proceedings of 2002 Workshop on Intelligent Human Augmentation and Virtual Environments*. Chapel Hill, North Carolina, October 2002. Pages D-26 and D-56. Poster Presentation.

### **Technical Reports**

1. D. Burschka, G. Ye, J. Corso, and G. Hager, A Practical Approach for Integrating Vision-Based Methods Into Interactive 2d/3d Applications, CIRL Lab Technical Report CIRL-TR-05-01, The Johns Hopkins University, 2005.

2. G. Ye, J. Corso, and G. Hager, A. Okamura, Augmented Reality Combining Haptics and Vision, CIR Lab Technique Report, The Johns Hopkins University, 2003.
3. G. Ye and G. Hager, Appearance-based Visual Interaction, CIR Lab Technique Report, The Johns Hopkins University, 2002.
4. G. Ye, H. Zhang, and G. Xu, Face Identification: Research and System Implementation, Computer Vision Lab Technique Report, Tsinghua University, 2000.
5. G. Ye, L. Wang, and G. Xu, Projective Rectification for Face Detection, Computer Vision Lab Technique Report, Tsinghua University, 2000.
6. G. Ye and G. Xu, Face Detection under Complex Background, Bachelor Degree Thesis, Department of Computer Science, Tsinghua University, 1998.

## **RESEARCH INTERESTS**

Computer Vision

Pattern Recognition

Machine Learning

Human-Computer Interaction

## **AFFILIATIONS**

Student Member of IEEE, Computer Society