University of Alberta

UBIHAND: A WEARABLE INPUT DEVICE FOR GESTURE RECOGNITION AND 3D INTERACTION

by

Farooq Ahmad ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring 2007

Your file Votre référence
ISBN: 978-0-494-29926-5
Our file Notre référence
ISBN: 978-0-494-29926-5

University of Alberta

Library Release Form

Name of Author: Farooq Ahmad

Title of Thesis: UbiHand: A Wearable Input Device for Gesture Recognition and 3D Interaction

Degree: Master of Science

Year this Degree Granted: 2007

Farooq Ahmad

Farooq Ahmad

Date: January 11, 2007

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **UbiHand: A Wearable Input Device for Gesture Recognition and 3D Interaction** submitted by Farooq Ahmad in partial fulfillment of the requirements for the degree of **Master of Science**.

Petr Musilek
Supervisor

Marek Reformat

Michael Bowling
External Examiner

Date: January 11, 2007

Abstract

The widespread adoption of mobile electronic devices and the advent of wearable computing have encouraged the development of compact alternatives to the keyboard and mouse. These include one-handed keyboards, digitizing tablets, and glove-based devices. This thesis describes a new wearable input device for gesture recognition, pointer control, and 3D interaction. It uses miniature wrist-worn wireless video cameras that track finger position.

For gesture recognition and text input, rapid finger movements are detected as keystrokes, and a Hidden Markov Model is used to correlate finger movements to keystrokes during a brief training phase, after which the user can type in the air as if typing on a standard keyboard.

The device can also be used as input interface for virtual reality. A hand model is used to analyze finger-tracking data and determine finger position in three dimensions, and a wrist-worn marker is used to track the hand within a virtual scene.

Acknowledgements

Completing this thesis would not have been possible without the encouragement and support of many people. First, I would like to thank my supervisor Petr Musilek for providing the perfect balance of direction and flexibility that allowed me to pursue new ideas without straying too far off track, and for his unfailing guidance throughout my degree. In my office at the FACIA lab, I have been surrounded by many interesting and helpful colleagues who have made the time I've spent working on this thesis seem to go by very quickly. I would especially like to thank Yifan, Armita, and Nitin, for their support, constructive criticism, and interesting discussions.

I have also benefited from my many interactions with the Computer Science department. Greg Kondrak was vital in helping me explore the field of natural language processing, and I would like to thank him for his confidence and guidance. Michael Bowling also suggested many valuable revisions to this thesis. In addition, I would like to thank the system administrators, and especially Grzegorz Dostatni, for keeping the labs up and running.

Finally, I would like to thank those closest to me for their encouragement throughout my degree. I would like to thank my parents for their tremendous support, my sisters for their enthusiasm and faith in me, and Nicole, for helping make the last year of my thesis the most rewarding.

Table of Contents

List of Tables

List of Figures

Chapter 1

Background and Motivation

1.1 Motivation

Mobile computing and wireless networking technology has advanced the integration of computing into everyday life. Globally, over 1 billion SMS messages are sent per day, and mobile phone use for messaging and Internet access continues to grow, motivating the need for simple and efficient portable input devices. Several technologies have been developed in order to meet this need. For example, handwriting recognition using pressure sensitive screens and a stylus has been successfully incorporated on many handheld computers. Onscreen keyboards and small 'thumb-boards' are another popular input method. However, input speed achieved with these devices is slow relative to standard keyboards [20]. The surface area available for input is even less on cellular phones, where 12 digit numerical keypads usually do double duty for text entry. Generally, when mobile phone keypads are used for text input, each key is mapped to three or four letters and predictive text entry algorithms are used to disambiguate key-presses. The application of predictive text entry methods to mobile phone text entry has significantly lowered the average number of keystrokes required per character (KSPC), from 3 KSPC to close to 1 KSPC. Nevertheless, key sequences for shorter words are often ambiguous, and the limited keypad size makes input speed slow relative to full-size keyboards. See [20] and [25] for a review of keypad based mobile input technologies.

An alternative to keypad-based input is based on virtual keyboards that do not rely on an input surface or physical keys to detect input. For example, the keyboard is projected onto a flat surface and key presses detected by infrared ranging sensors [32]. The increasing sophistication of mobile technology has also supported the emergence of multimedia, entertainment, and augmented reality applications for mobile computers. Augmented reality and wearable computers in particular require input interfaces that are lightweight and unobtrusive.

One barrier to the adoption of wearable computing devices is the absence of an input interface that allows the user to interact with the system while standing or moving. Typically, portable keyboards must be set or held in place to be used effectively. One approach to solving this problem is one-handed (split) keyboards. However, wearing a wrist-mounted split keyboard interferes with the use of the hands to do other tasks. Glove-based devices incorporating flex sensors within the glove fingers are less cumbersome, allowing free use of the hands. Another input method analyzes the data collected from tiny accelerometers worn on each finger to determine user input [9]. An even less obtrusive approach is virtual glove-based input devices, that use wrist mounted sensors to track hand and finger position. For example, the lightglove [16] uses infrared LEDs and sensors worn on the arm to track finger position and allows the control of electronic systems. The concept of using wrist mounted cameras as input devices is introduced in [40], where the fingers are tracked and used for input. Virtual glove-based devices enable the user to interact with a mobile computer while moving, and allow free use of the hands while not in use. However, the virtual glove devices mentioned above rely on a chorded input scheme to interpret finger movements. Since there are only 10 fingers available to represent all alphanumeric characters, several fingers must be moved simultaneously for each input. This requires the user to learn a new input method to use the devices.

The input system described in this paper can interpret non-chorded input, allowing the user to type as if using a standard keyboard, as each finger can be used to type several different characters. Probabilistic models of finger movement and language are learned and used to resolve ambiguous inputs. The use of cameras allows high-resolution tracking of the position of each finger at all times. In addition, environmental features can be tracked to determine the position of the hand itself.

The incorporation of vision-based ego-motion estimation provides for an intuitive method of pointer control by translating hand movements to pointer movements.

1.2 Overview of this thesis

The thesis is organized as follows. Chapter 1 provides a background on human computer interaction, image processing, gesture recognition, wearable computing, and 3D interaction. Chapter 2 provides a description of the hardware components of the prototype system, and an overview of the system organization. Chapter 3 describes the finger tracking, keystroke detection, and gesture recognition process. Chapter 4 introduces visual feature tracking for motion estimation and pointer control, and Chapter 5 describes how the device can be used for augmented reality interaction. In Chapter 6, experimental results obtained from the prototype system are discussed. Conclusions and possible extensions are explored in Chapter 7.

1.3 Background

This thesis describes an input device for text input, gesture recognition, and 3D interaction, and can best be described as falling under the field of Human Computer Interaction (HCI). However, Human Computer Interaction draws from a wide spectrum of research, and the work described here is no exception. The wearable input device described in this thesis is based on wrist worn cameras, and is designed for gesture recognition and 3D interaction. Thus the fields of computer vision, natural language processing, wearable computing, and augmented reality are all touched upon in this thesis, and so the relevant details of each field are described in this chapter. First, the basics of digital image capture and image processing are described. Since the field of computer vision is large, only the algorithms used in this thesis, including those used for object segmentation, edge detection, and feature tracking, are described. As hardware speed and image processing algorithms have progressed, much research has also been done towards higher-level recognition algorithms, including vision-based recognition of human gestures as a means of human computer interaction. A summary of gesture recognition research is presented, with particular emphasis on probabilistic recognition of human hand gestures.

Alternative methods of input are becoming increasingly relevant due to the emergence of wearable computing systems whose form factor does not mesh well with standard keyboard and mouse input systems. So, the current state of input technology for mobile and wearable computers is described next in the background. As wearable hardware technology matures, new applications are being developed to take advantage of the new smaller form factors. For example, wearable displays make augmented reality (AR) applications possible, which is the integration of virtual objects within a real scene. An overview of the field of AR is provided, as well as a review of current input devices for virtual and augmented reality applications.

1.3.1 Digital Image Processing

Digital image processing is a broad field of study that involves obtaining information from images using a computer. Information extraction can be low level, for example to clean up an image using convolutional filters, or high level such as recognizing objects within an image or determining the 3D structure of the scene. Digital image processing can be categorized into several classes: image processing, image analysis, computer vision, and machine vision. Image processing refers to operations that are applied to two-dimensional input images, where the end result is also two-dimensional output images. Image analysis/understanding is the higher level processing of images to extract meaning, for example recognizing and classifying objects within a scene. The term ”computer vision” usually refers to the extraction of 3D structure from one or more 2D images, and ”machine vision” is centered on the industrial applications of image processing. Before delving into the specifics of image processing algorithms, a brief summary of the digital image formation process is described next.

1.3.2 Cameras and Image Formation

Image formation involves the mapping of the three-dimensional world to a two dimensional image. Several camera models exist to describe this mapping; the most basic camera model is the pinhole or perspective model, based on a box with

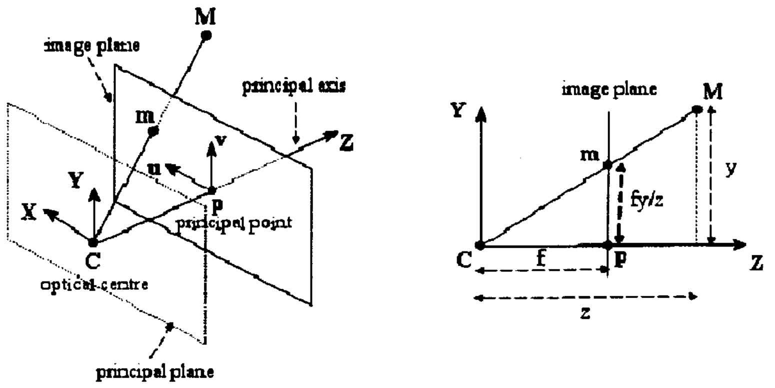

Figure 1.1: The pinhole camera model. (Source: [10])

a small hole in it on one side of the box that lets in light to create an image on the opposite side. The model is shown in Figure 1.1.

The pinhole camera is described in terms of the optical centre C (the pinhole), the optical axis z, and the focal length f, which is the distance between the pinhole and the image plane. Light enters the camera through the pinhole and hits the image plane at the back of the camera. Thus, given the size of an object on the image plane, the focal length, and the distance to the object, similar triangles can be used to determine the real world size of the object.

In particular, a 3D point (x,y,z) in the world will be mapped to the point (fx/z, fy/z) on the image plane. The relationship between 3D points and their image position is called the perspective equation.

Using matrices, the perspective equation can be expressed in homogenous coordinates as:

$$\begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (1.1)$$

Or expressed simply as $zm = pM$, where $M = (x, y, z, 1)$ are the homogenous coordinates of a 3D point, and $m = (fx/z, fy/z, 1)$ are the homogenous coordinates of the image point.

The General Projection Equation

In a more general case, world coordinates may not be exactly aligned with camera coordinates. Thus, the projection matrix will need to be multiplied by a matrix describing the rotation and translation of world coordinates relative to the camera. Both rotation and translation can be expressed in a single 4 x 4 matrix G containing a 3 x 3 rotation matrix R, and a 3 x 1 translation matrix t:

$$G = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{1.2}$$

In addition, camera parameters such as lens distortion and focal length will modify the appearance of objects within the image. The camera calibration matrix K describes the intrinsic camera parameters that determine how an image is projected onto the image plane. In general, the matrix will vary from camera to camera and will need to be calibrated for each one. Taking into consideration the camera reference frame relative to the world, G, and the intrinsic parameters, K, gives a more general form of the projection matrix P:

$$P = K[R|t] \tag{1.3}$$

Thus the image formation process is fully described by the following equation:

$$m = PM = K[R|t]M \tag{1.4}$$

1.4 Image Processing

Image processing is a subset of signal processing and is defined as any kind of operation in which the input and output are both images. Digital image processing involves the digitization of analog images into a discrete representation as a 2D array of real numbers. The image can thus be considered as a function over two variables, $I(x, y)$, where each element, or pixel, is a real valued number.

The exact digitization process that converts incoming light into a discrete representation using Complementary metaloxidesemiconductor (CMOS) or charge-coupled device (CCD) sensors is not described here. This thesis places greater focus on methods of processing the image to extract information once digitization has already taken place. Image processing operations can be categorized into 1) pixel-based, in which each pixel is modified independently of the values of the surrounding pixels; 2) local operations, in which pixel values are modified dependent

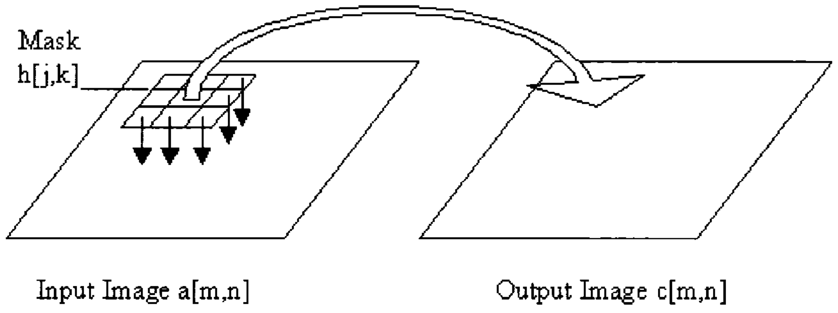

Figure 1.2: 2D image convolution

on the neighboring pixels and a weighted mask/window; and 3) global operations in which all pixel values in the image are taken into consideration. The operations that are most relevant to this thesis are local operations including 2D convolution and morphological operations, and they are described below.

1.4.1 Convolution

Convolution is a local operation that acts on a small part of an image at a time. The operator consists of a sliding window, called the support, kernel or mask, which is scanned across an image. The mask is simply a two dimensional matrix of real numbers. As the mask is moved across the image, each input image pixel that lies within the mask is multiplied by the window value at the corresponding location. The output pixel value corresponds to the sum of these multiplications. The window, along with the associated weights, is called the convolution kernel. Convolution operators can be used to sharpen or blur an image, or find image features such as edges or corners.

The convolution process is illustrated in figure 1.2. In formal terms, the output image $c[m, n]$ of convolving input image $a[m, n]$ with mask $h[m, n]$ is described by equation 1.5.

$$c[m, n] = a[m, n] \star h[m, n] = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} h[j, k] a[m - j, n - k] \qquad (1.5)$$

An example of a convolution-based operation is Gaussian smoothing. The Gaussian kernel approximates a two dimensional Gaussian curve centered at the kernel center. Typical kernel sizes are 3x3, 5x5 or 7x7. The operator is applied to

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 1.3: Masks for dilation(left) and erosion(right)

each pixel by multiplying the value of the surrounding pixels by the kernel values, and normalizing the result. In effect, each pixel takes on the values of nearby pixels, with the nearest pixel having the greatest effect. The result is an image where all edges are smoothed, and detailed features are blurred.

1.4.2 Image Morphology: Dilation and Erosion

Morphological operations are functions applied to images to aid segmentation and classification of objects within the image. A two dimensional structuring element is applied to a small region of the image at a time, analogous to the kernel used in convolution. In contrast to convolution however, set operations such as intersection and complement are applied to the input image and structuring element rather than sums of multiplications. The two principle morphological operations are dilation and erosion. They are typically applied to binary images.

Dilation is used to expand the boundaries of objects, and filling small holes within an objects interior. A mask is swept over the image, and the value of each output pixel is defined by the intersection of the mask with the input pixel and pixels surrounding it. For example, a 3x3 dilation mask is show on the left of figure 1.3. For a pixel to be set to 0, each pixel in the neighborhood of the pixel must be 0.

In contrast to dilation, erosion is used to shrink objects by eroding their boundaries. A 3x3 erosion mask is shown on the right of figure 1.3. For a pixel in the output image to be 1, the pixel and every one of the surrounding pixels must be 1. Otherwise, the output is 0.

More complex operations may also be created by combining dilation and erosion. The "'opening"' operation consists of erosion followed by dilation, and is used to filter out small specks in an image. "'Closing"' consists of dilation followed by erosion and is used to eliminate small holes or gaps within larger image blobs.

1.4.3 Feature Detection

The goal of many image processing applications is to detect, segment, and recognize meaningful parts of an image. Often the first stage in object segmentation and recognition involves detecting features in the image that aid in classification. For example, image features such as edges, lines, and corners can be used for recognition and tracking tasks.

Edge Detection

Edge detection is used to mark areas in an image where a large change in luminosity is present. Generally, edges demarcate the boundaries of objects that appear within an image, although edges will also appear within objects themselves. Thus edges provide a compact representation of scene structure, filtering out less relevant data. Several edge detection algorithms exist; the simplest method of edge detection is to find the image gradient using the first derivative of the image intensity. Taking a pixel value and subtracting the value of a neighboring pixel gives a rough estimate of the first derivative of the image at that point. Given a grayscale intensity image, horizontal edges and vertical edges are found in two passes; For horizontal edges each output pixel I'(x,y) corresponds to the intensity $I(x,y) - I(x-1,y)$, and for vertical edges $I'(x,y) = I(x,y) - I(x,y-1)$. This is equivalent to convolving the input image with the mask $\begin{bmatrix} -1 & 1 \end{bmatrix}$, and then $\begin{bmatrix} -1 & 1 \end{bmatrix}^T$. Pixels in the image derivative that are above a certain threshold are considered edge pixels. However, determining the correct threshold value will depend on the illumination and composition of objects in the scene.

Corner Detection

A common problem in computer vision is object tracking and camera motion estimation in an image sequence. In order to determine the motion of the camera relative to objects in the scene, visual features must be tracked robustly across frames. Corner features are an attractive image feature for tracking as they can be found quickly, and tracked robustly in two dimensions. The most commonly used corner detection algorithm is the Harris corner detector. The Harris corner detector is described in more detail in Chapter 4.

1.5 Gesture Recognition: Data Collection and Analysis

The goal of gesture recognition is to identify and classify human gestures and use them as a method for human computer interaction. It involves the process of converting a continuous signal into a set of discrete symbols, and shares much in common with handwriting and speech recognition. For example, speech recognition involves taking an acoustic signal, splitting it into words, and recognizing each of the words. Similarly, handwriting recognition and optical character recognition involves splitting an image into meaningful segments such as letters and words, and determining what each portion represents. In gesture recognition, the input signal is created by the continuous movement of one or more human body parts,

The principle elements of a gesture recognition system are the interface used for data input and the recognition algorithm used to classify the data.

- The interface describes what is measured and how the data is collected. For example the system may use full body tracking, head tracking, hand tracking, or the tracking of any other body part. Similarly, the sensors used may be mechanical, inertial, magnetic, ultrasonic, or vision based.

- Algorithms for gesture recognition can be based on simple heuristics, neural networks, or other machine learning techniques. The most successful ones are often based on learned probabilistic models, but there is a wide variation in model structure and how they are learned.

1.5.1 Interfaces for Human Motion Tracking

Mechanical, Inertial, Magnetic, and Ultrasonic Tracking

The oldest method of tracking human motion is to use mechanical trackers based on flexible goniometers. The goniometers are worn on each limb and measure joint bend angles, which are analyzed by kinematic algorithms to determine body position. A less obtrusive method of tracking human motion is to use small wireless accelerometers worn on the body parts to be tracked. Miniature accelerometers can provide 3D acceleration measurements at a high frequency, typically around 100Hz. The accelerometer data can then be integrated to determine position, but accelerometers are prone to drift, making accurate position determination difficult. Other systems use body worn ultrasonic transmitters with multiple receivers that analyze the time of flight of the sound to each receiver to determine position. Ultrasonic systems provide absolute position measurement, but the accuracy is limited by the speed of sound and measurement error due to signal variation. Finally, magnetic motion capture systems use sensors worn on the body to detect magnetic fields generated by three perpendicular magnetic field sources. Sensing accuracy is good, but such systems are expensive, and the magnetic field measurement is susceptible to interference from ferromagnetic materials.

Vision Based Marker Tracking

The most popular method of human motion determination is vision based tracking that tracks several markers worn on the body using one or more cameras. Markers may be active, for example blinking LED lights, or passive markers that are simply reflective balls or tape worn on several points on the body. In professional motion capture systems, around 40 markers are embedded in a body suit that is worn by an actor. Two or more cameras track the actor, and software is used to determine the 3D position of each marker and determine the most likely human body motion using kinematic constraints. Marker based systems are very accurate, giving $\pm$ 1mm accuracy, and the large number of markers overcomes any information missing due to marker occlusion. Marker based tracking systems, however, are expensive, need to be precisely calibrated in controlled illumination conditions and are typically used in a studio setting with fixed equipment.

Marker-less Tracking

Recent research focuses on determining body position without the need for special markers. Markerless motion capture involves segmenting the user from the background and determining body position using kinematic constraints.

1.5.2 Hand Tracking

While full body motion capture systems provide accurate position measurements, their cost and set up requirements make them unsuitable for routine human computer interaction. Instead, gesture recognition systems generally rely on tracking a few key body parts, such as the hand, head, or eyes. Hand tracking allows for the most expressive gestures, and requires less effort on behalf of the user than full body tracking. This makes it the most popular modality for gesture recognition. Hand tracking systems traditionally fall into one of two categories; 1) glove-based systems that rely on gloves instrumented with flex sensors to measure finger bend angles, and accelerometer or ultrasonic sensors for measurement of hand position; and 2) Vision based systems that either use one or more static cameras positioned in front of the user.

More recently wearable cameras have begun to be used for HCI. For example, the tinmith project [28] uses head mounted displays with integrated cameras to track gloves embedded with markers, and the gesture pendant [11] relies on a camera worn around the neck to track the hands.

Glove-Based

Glove-based hand motion capture utilizes gloves with embedded fiber optic sensors that measure the flexion and adduction of the fingers. Several companies offer gloves with embedded flex sensors. The least expensive models, such as the 5DT Dataglove, use one flex sensor for each finger, providing approximate bend angle measurements but no adduction measurements. Other models embed up to 24 sensors per hand, measuring two bend angles per finger, plus inter-finger adduction angles and wrist flexion. Generally, the gloves are meant to be used with 3D position sensors in order to measure the absolute position of the hand.

Vision Based

Vision based hand trackers generally rely on one or more fixed cameras observing the user. As long as the users hand is within the field of view of the camera, it can be tracked. While hand position can be tracked robustly, the accuracy is generally poor and precise tracking the pose of each finger is difficult. Such systems are prone to errors due to occlusion of part of the hand or fingers.

1.5.3 Gesture Recognition

As the gesture data is collected using one of the methods described above, it must be interpreted and translated into meaningful units. Gesture recognition can be divided into two phases;

1. splitting continuous gestures into meaningful units; and
2. recognition of the units and determining their symbolic value

The two steps can occur in two separate stages, with discretization followed by recognition, or the recognition system can be used to aid the segmentation stage. Early research into speech recognition typically approached the two tasks separately. For example, by splitting the acoustic signal where the amplitude dips, and then attempting to match each segment to a word. This required the utterance of each word to be separated by a pause. More recent research involves considering several possible segmentations, and performing segmentation and recognition in parallel using probabilistic algorithms.

Many processes can be modeled as a set of states, with an associated set of probabilities that describes the chances of transition from one state to another. Indeed, most speech and gesture recognition systems are based on probabilistic models. In particular, Hidden Markov models (HMM) find abundant use in speech and gesture recognition.

Hidden Markov Models for Gesture Recognition

A stochastic process in which future states are dependent only on the present state, that is, they are conditionally independent of states preceding the current state, are said to have the Markov property. This property is formally described by the equation:

$$Pr[X(t+h) = y|X(s) = x(s), \forall s \leq t] = Pr[X(t+h) = y|X(t) = x(t)], \forall h > 0 \tag{1.6}$$

Where $X(t+h)$ is the state at time $t+h$, independent of the state at any preceding time s.

In a Markov chain, the current state is observable; in this case the system can be described in terms of the states and the transition probabilities between them. However, in some processes, the current state is not directly observable. In this case, observable parameters may provide probabilistic clues as to which state the process is in. This kind of process can be modeled as a Hidden Markov Model (HMM).

Hidden Markov Models are described in terms of:

1. N, the number of states of the model
2. M, the number of distinct observation symbols per state
3. the set of observation probabilities $B = \{b_j(k)\}$ describing the probability of each kind of observation v_k within each state j where $B_j(k) = P[v_k|q_t = S_j], 1 <= j <= N, 1 <= k <= M$
4. the set of transition probabilities A that describe the probability of moving from one state to any other state. A is a matrix whose elements a_{ij} correspond to the probability $P[q_{t+1} = S_j|q_t = S_i], 1 <= I, j <= N$

In general, pattern recognition tasks that can be solved using hidden Markov models fall into three categories:

1) Computing the probability of a particular output sequence given the model parameters. Given a prior model of weather probabilities, for example, one may wish to determine the probability of having 10 sunny days in a row. This problem is easily solves using the forward algorithm [30].

2) Finding the most probable sequence of hidden states given a set of observations and the model observation and transition probabilities. In this problem, the

HMM parameters are known, and a series of observations must be classified using the model. For example, in the context of sign language recognition, a prior model of the probabilities of observing each sign, and transitioning from one hand sign to another may be known. Then, given a series of observations of hand positions, the most likely word needs to be found. This problem is slightly more difficult to solve than 1, and is solved using the Viterbi Algorithm discussed in Chapter 2 [30].

3) Parameter learning: Given a set of observation sequences, determine the most likely set of observation and transition probabilities. This problem requires optimizing the model parameters with respect to observations. and is thus the most difficult to solve. This problem is solved using the Baum Welch algorithm [30].

The problem addressed in this thesis, namely recognition of finger gestures, involves solving the second and third types of problem and they are described in more detail in Chapter 2.

Evaluation of Gesture Recognition Systems

Gesture recognition systems can be evaluated using the following criteria

- Input speed
- Usability and the learning curve required for new users
- Vocabulary, the number of discrete symbols that can be accurately recognized;
- Accuracy - the percentage of input gestures that are correctly identified;
- Robustness to gesture variability for a single person and variability among different individuals, and to signal transmission errors;
- Practical concerns such as size and cost

1.6 Mobile Computing

Since the invention of computers, their form factor has evolved at an amazing rate. The size and processing speed has developed rapidly, from large room sized computers of the sixties, to mini computers of the seventies, to portable computers that revolutionized the eighties and nineties, and most recently to smart phones and mobile devices that can be carried in a pocket. The changing form factor of computers continues to expand their applications and inclusion into everyday life. Today, mobile computers are pervasive, outnumbering their deskbound counterparts. While originally meant for voice communications, the role of mobile phones has expanded to data communications. Mobile phones are used for text messaging, games, surfing the Internet, multimedia applications, and scheduling. However, the small form factor of mobile phones makes interaction difficult.

1.6.1 Mobile Computing Text Entry Methods

The principle input device for mobile phones is a small 10-button keypad. For text input, multiple letters are assigned to each key, and characters are disambiguated probabilistically. Other input methods include handwriting recognition using a touchpad, miniature QWERTY keyboards, laser projected keyboards and voice recognition. A summary of mobile input technologies and their relative advantages is described in Tables 1.6.1 and 1.6.1.

A review of current input technologies for mobile devices can be found in [5].

1.6.2 Relative Advantages of the UbiHand Input Interface

The advantages of the input device described in this thesis relative to the technologies described in Tables 1.6.1 and 1.6.1 are:

- It is compact and portable; wearable cameras can be worn just like watches and turned on and off as needed;
- Keystroke recognition system allows one-to-many finger-key mappings and doesn't require chorded input;
- Input speed is fast relative to other portable input technology;
- Mouse pointer control is integrated into the same device; and

Table 1.1: Current input technologies for mobile devices

Type	Description/Example	Advantages	Disadvantages
1. Non touch-typing input			
Digitizing tablet/handwriting recognition	PALM Graffiti	Intuitive if regular handwriting can be used as input	Slow relative to touch typing; non-standard alphabets require training time
Voice recognition	Dragon Naturally Speaking; IBM Via Voice	Compact - microphone is only required input device	Error prone and imprecise; no pointer control
Gesture recognition (static camera)			Difficult to learn; slow; limited number of input symbols
2. Touch-typing input with a physical keyboard			
Full-size wireless QWERTY Keyboards and collapsible keyboards		Fast; easy to use; no learning curve	Large and cumbersome; can't be used while on the move
Wrist-worn Keyboards	Symbol Technologies keyboard and barcode scanner	Shallow learning curve	Large and obtrusive; Must be operated with one hand - slow input speed
PDA miniature QWERTY thumb keyboards	PocketPC, iPaq, Axim thumb keyboards	Compact; Learned quickly by experienced touch typers	Slow relative to standard keyboard; uncomfortable
Numeric keypads on cellular phones with predictive text entry	All cellular phones that allow text messaging; predictive text entry algorithms include Tegic T9, Eatoni Letterwise.	Compact; Predictive text entry can improve input speed	Slow relative to standard keyboard; multiple keypresses required per input letter

Table 1.2: Current wearable and virtual keyboard based input technologies

3.1 Touch-typing input with virtual keyboards			
Laser projected virtual keyboard	Image of a keyboard is projected onto a flat surface Canesta Electronic Perception; Vkey Virtual Keyboard	Compact relative to standard keyboard	Currently too large for cellular phones; poor accuracy; requires a flat surface
3.2 Wearable input devices			
Gloves with flex sensors	Fiber optic flex sensors in each finger record flex and interpret as input Vtype glove; Scurry Glove	Wearable	Cumbersome; require chorded input
LightGlove[4]	Lightglove virtual keyboard	Hands-free; wearable	Relatively large and obtrusive; Requires chorded input
FingeRings	Rings on each finger contain accelerometers that communicate movement information wirelessly FingeRings;	Wearable	Cumbersome; require accelerometers on each finger; requires chorded input

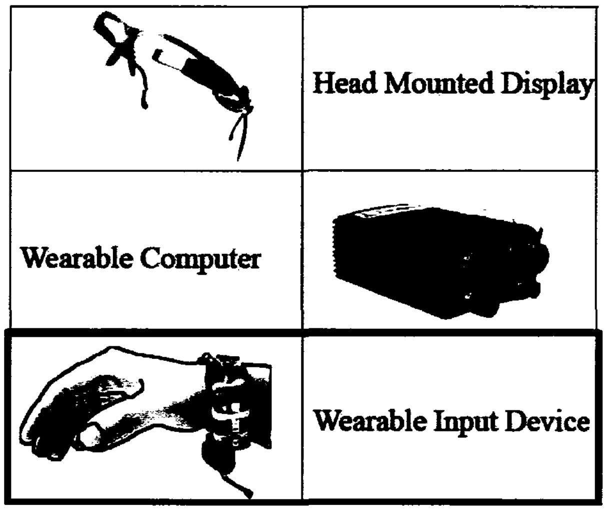

Figure 1.4: Components of a wearable computing system

- Use of CMOS camera technology allows cost, power and size savings as the industry advances. In addition, using a wrist worn camera allows high resolution tracking of finger position that is required for accurate gesture recognition and hand motion capture.

1.7 Wearable Computing

The shrinking size of modern computers has also facilitated the growth of wearable computing applications. In short, a wearable computer is an ultra portable computer system that consists of an input device, processor, and display that can be worn. The focus of wearable computing research is to make interacting with the computer as transparent and natural as possible. The elements of a complete wearable system are shown in figure 1.4.

Chapter 2

Hardware and Software Overview of the Prototype System

A prototype was built using an off-the-shelf miniature camera fastened to a watch strap, and is shown in figure 2.1. The wireless CMOS color camera is worn on the underside of the wrist, pointing towards the fingers, and is powered by a 9V rechargable battery that provides power for about 4 hours. The images are transmitted to a wireless receiver connected to a video capture card on a PC that processes the video stream in real-time. The fingertips are tracked continuously, and keystrokes are detected when one or more fingers deviates from its rest position above a distance and speed threshold.

Figure 2.1: The prototype system

2.1 Hardware Overview

2.1.1 Camera

The camera is a wireless CMOS color camera with 380 lines or resolution, weighing approximately 15g and the size is 1.5cm $\times$ 1.5cm $\times$ 1.5cm. It is powered by a 9V battery, giving about 4 hours of use on one charge. The field of view is approximately 45 degrees, providing a clear view of four fingers, but excluding the thumb. A lens with a wider field of view could be used to include the thumb. In terms of wearability, the camera is much smaller and lighter than current glove-based systems. However, the camera is still quite large and not as natural as, for example, wearing a watch. Nevertheless, there are currently several CMOS camera sensors on the market that are much smaller than the one used. For example, the Supercircuits PC208 measures 0.8 $\times$ 0.8 $\times$ 0.8cm, and is not very expensive ($129). The PC208 consumes 35mA at 12 V DC, giving a battery life of about 4 hours. The continued popularity of mobile camera phones continues to drive down the size and price of small CMOS camera sensors, so that the form factor is expected to improve.

2.1.2 Wireless Connection

In the prototype system, the video from the wrist-mounted cameras is sent wirelessly to a receiver connected to a PC. The video is sent at 2.4 GHz, with 380 lines resolution NTSC video at 30 fps. However, signal noise often causes errors in finger recognition. It would be preferable to do all of the processing on an on-board Digital Signal Processor (DSP), sending only the processed commands to a mobile or wearable computer. Several DSP chips specialized for video processing are available that could accomplish this task. For example, the Videocore DSP is a low power mobile DSP specialized for image processing that is able to do edge detection, segmentation, and tracking at 30fps on a mobile phone.

2.2 Software Overview

The system described in this thesis includes several components. The basic task performed by the software described here is to capture video from a wrist worn

Pointer Control	**Gesture Recognition**	**Augmented Reality Interaction**
Corner Detection Corner Tracking Robust Motion Estimation	Finger Tracking Keystroke Detection Keystroke Recognition	Finger Pose Estimation Hand Tracking Virtual Hand and Object Video Overlay

Table 2.1: Major software components

camera, and analyse the video for keystroke recognition, pointer control, and 3D interaction. The application is composed of several components written in C and C++. The low level vision code that handles grabbing images from the video capture card, and color thresholding is written in C. The higher level code for finger tracking, gesture recognition, and pointer control is written in C++. The graphical user interface is also written in C++, using X11 graphics primitives. Table 2.1 lists the major elements of each subsystem. These three tasks are more or less independent, and are described in order below.

2.2.1 Finger Tracking and Gesture Recognition

At the heart of the system is the finger segmentation and tracking algorithm. Using skin color segmentation and several filters, four fingers are segmented from the background, and the position of the fingertips and interfinger joints are tracked. This information is used for keystroke detection, which involves detecting rapid movements of one or more fingers from the rest position. In turn, keystroke observations are interpreted by the gesture recognition system, that classifies the detected keystrokes as characters or commands. The major subcomponents are:

- Skin Segmentation and Filtering
- Finger Tracking
- Keystroke Detection
- Gesture Recognition

Finger tracking and gesture recognition is detailed in Chapter 3.

2.2.2 Pointer Control

Features in the background (i.e., the non-finger component) are used for feature tracking for pointer control. Corner features are detected and tracked using a probabilistic algorithm, allowing the user to move his or her hand in order to control a mouse pointer in two dimensions. The major subcomponents are

- Corner detection
- Robust corner tracking
- Translating movement to pointer control

Pointer control is described in Chapter 4.

2.2.3 3D Interaction

An extension of the system allows it to be used for 3D interaction, and manipulation of virtual objects in augmented reality. For this purpose, a fiducial marker is worn on the wrist, and the system is used along with a second camera; a static or head worn camera is used to track the relative motion of the hand, and the wrist worn camera tracks the fingers. The major subcomponents are

- 6 degree-of-freedom (position and orientation) hand tracking using markers
- Finger pose estimation using inverse kinematics
- An augmented reality display that integrates virtual objects into a video stream

3D interaction is discussed in Chapter 5.

2.2.4 Feasibility

The feasibility of implementing the image processing algorithms on board a mobile processor is dependent on the computational complexity of the algorithms involved. Currently, all image processing takes place on a 1GHz Athlon PC in real time. However, the system is amenable to implementation on a low power mobile DSP; the

Processing Stage	**Complexity**
Skin Thresholding Each pixel is assigned as either skin color or non-skin color, depending on whether the pixel values fall within threshold values.	O(NM)
Filter - run length The skin segmentation is refined using a minimum run length threshold. Specifically, each column of the image is processed from top to bottom to find the bottommost skin pixel.	O(NM)
Filter - morphological operations Dilation and erosion are applied to the binary image to refine the skin segmentation. O(NM) - Due to the fact that there exists a simple algorithm for dilation and erosion with complexity independent of the size of the structuring element or kernel [12].	O(NM)
Filter - Gaussian convolution of contour The extracted one finger contour is smoothing using one-dimensional Gaussian convolution. The contour is N pixels long for any N*M image, so convolving the contour with a kernel of length K gives O(NK) complexity. A filter length of 5 is used in practice.	O(NK)

Table 2.2: Computational complexity for each stage of finger segmentation

computing requirements for finger tracking are described in Table 2.2.4. An image resolution of N x M pixels is assumed, where N is the width of the image and M is height. In practice, a resolution of 320 x 240 pixels was found to be more than sufficient for accurate finger tracking.

Chapter 3

Keystroke Detection and Recognition

3.1 Overview

Keystroke movements are assigned to characters during a training stage, and a Hidden Markov Model (HMM) is used for character recognition. The three stages of detecting and identifying keystrokes are described below. The information in the next two chapters is substantially similar to that provided in [1].

1. Hand Extraction and Fingertip Recognition

- The hand contour is extracted from the images by skin color discrimination;
- The fingertips are detected as the minima of the extracted hand contour

2. Keystroke Detection

- The finger rest position is recorded in an initialization stage;
- Fingertip position is tracked over time, and keystrokes are recognized as peaks in the deviation of the finger position from the observed rest position

3. Hidden Markov Model Based Character Recognition

- The correlation between finger movement and character/command input is observed during a brief training phase;

4. During operation, keystrokes are interpreted as characters and language statistics are used to disambiguate keystrokes.

3.2 Hand Extraction and Fingertip Detection

The video camera worn on the underside of each wrist continuously records the hand and fingers. The contour corresponding to the edge of the hand is extracted from the image stream by means of color segmentation.

3.2.1 Review of Skin Color Segmentation Algorithms

The approach to skin color segmentation taken in this thesis is naive thresholding, followed by morphological filtering and filtering algorithms that exploit the near vertical alignment of the fingers relative to the wrist worn camera, described further in section 3.2.3. Accurate skin color segmentation is an issue central to human gesture recognition, and other approaches to unconstrained skin segmentation are described below.

Skin segmentation should be robust to variations in skin color among different individuals, as well as variation due to changes in illumination, and much research has been done into determining the optimal color spaces and thresholds for skin segmentation. Color spaces are abstract representations of color consisting of tuples of numbers, usually made of three or four color components. For example, in the RGB color space, all colors are expressed as a linear combination of red, green and blue. In HSV color space, colors are represented by their Hue H, the color type, saturation S, the vibrancy of the color, and the value V, or brightness of the color. Generally, skin color segmentation is done in HSV color space. However, in [3], it was shown that the color space has no influence on skin color detection performance as long as the optimal skin color detector for that space is chosen, and that an invertible transformation between the compared color spaces exists.

Rather than relying on predefined thresholds for skin segmentation, using Gaussian mixture models of skin color generally provide better results. Skin color is described as one or more Gaussians in color space, where the parameters of the Gaussians are learned from examples using expectation maximization. Then, image pixels are classified as skin or non-skin according to Bayes rule:

$$P(skin|c) = \frac{P(c|skin)P(skin)}{P(c|skin)P(skin) + P(c|\sim skin)P(\sim skin)} \tag{3.1}$$

Where $P(c|skin)$ is the probability that a pixel is color c given that it is a skin pixel, and $P(c| \sim skin)$ is the probability that the pixel color is c given that it is not skin. Both are based on prior histograms of skin color. $P(skin)$ and $P(\sim skin)$ are a priori probabilities that skin will be found in the image at a particular location, and can be learned from examples.

This provides reasonable skin color segmentation under constant illumination. In most settings, however, changes in illumination will result in changes skin color over time. Adaptive skin color segmentation addresses this by updating the skin color model at each time step. [42] gives a review of skin color extraction algorithms.

3.2.2 Finger Tracking Overview

In this thesis, a naive algorithm is used to provide an approximate segmentation of skin color pixels, which was refined using several filters. The threshold based skin and non-skin discrimination is quite rough, and the known shape and position of the fingers is exploited to provide more accurate finger tracking.

Finger tracking proceeds in several stages, the result of which is the image coordinates of the four fingertips and three interfinger joints that lie within the field of view of the wrist worn camera in each frame. The stages are as follows:

1. Thresholded skin segmentation
2. Application of morphological operators to the binary image
3. Spatial smoothing of the finger contours using one dimensional convolution
4. Finding contour maxima and minima
5. Time-domain filtering of fingertip position using a moving-average filter

3.2.3 Skin Color Thresholding

A rough first pass classifies each image pixel as either skin or non-skin depending on whether the pixel lies within an experimentally determined range in HSV color space. A maximum and minimum value are assigned to each color component, and if each component of a pixel lies between the thresholds, the output pixel value is set to 1. Otherwise, it is set to zero, so the output of the operation is a binary image. The thresholds were determined by trial and error on a single user, and would need to be refined for users whose fingers were differently colored. However, the thresholds are set quite wide, and generally the palm side of most peoples hands fall within the same range of pixels. On the other hand, the high permissivity of the threshold results in several misclassified pixels, and often small groups of background pixels are mistakenly identified as skin color pixels. In addition, shading sometimes causes parts of the finger to appear very dark and outside of the threshold range, especially in the creases at each joint. Morphological filtering in the next stage helps eliminate most misclassified background and finger pixels.

3.2.4 Morphological filtering - Dilation and Erosion

After thresholding, the image generally consists of a large black blob corresponding to each finger, with small holes within, and some much smaller blobs and speckled noise surrounding them. The erosion operator is applied to the binary image to remove the smaller background blobs corresponding to objects in the environment that are similar to skin color. The erosion operator also opens up light pixels within the finger that are misclassified as non skin color pixels. However, the holes are not large enough to split the finger into disconnected components. Next, the holes are filled by the application of the dilation operator. After these two morphological operations, the output image is a relatively good segmentation of the fingers from background, with some extraneous pixels remaining. The process of finding the edge contours of each finger using this binary image is described below.

3.2.5 Finding the Finger Contour

The finger position relative to the camera is exploited to find the edges of each finger. Namely, the fingers always begin at the top and are aligned nearly vertically in each image. So, starting from the top of the image, a minimum run length threshold and maximum gap length threshold is applied to each column of pixels. That is, starting at the topmost pixel, the column is extended until a span of more than X

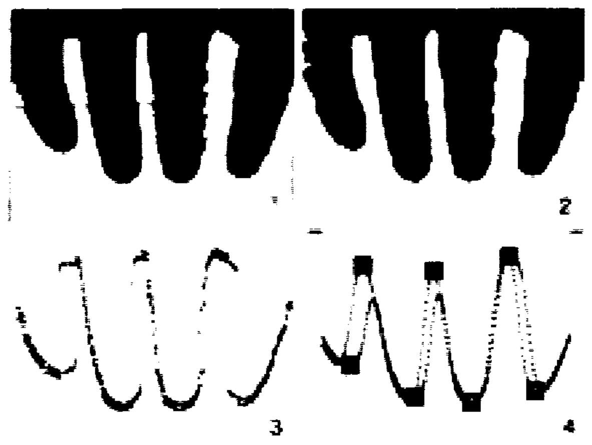

Figure 3.1: Stages in image preprocessing. The figure shows 1)the image after skin color detection, 2)after erosion and dilation, 3) the extracted edge contour and 4) the smoothed contour and extracted interest points

non-skin pixels is reached. A good value for X was determined experimentally as 20 pixels at an image resolution of 320x240. The bottommost pixel in this column is recorded as the edge of the finger. The run length thresholds provide robustness to noise caused by similarly colored objects in the environment.

3.2.6 Detecting Contour Maxima and Minima

The extracted edge contour described above is generally jagged due to noise, with several outlier pixels on each finger. The edge contour is smoothed using a 1D median filter to remove the discontinuities, so that the peaks and valleys of the extracted contours correspond to the fingertips and interfinger joints. The peaks and valleys are found by iterating through the contour and finding the alternating maximum and minimum Y coordinate values. The output is the image coordinates of the four fingertips and three interfinger joints. The preprocessing stages are shown in figure 3.1.

Four fingers and three phalanges are visible within the field of view of the camera, giving seven interest points to describe the hand and finger position. Figure 3.2 shows the fingertip and phalange detection, as well as the centroid of the interest points.

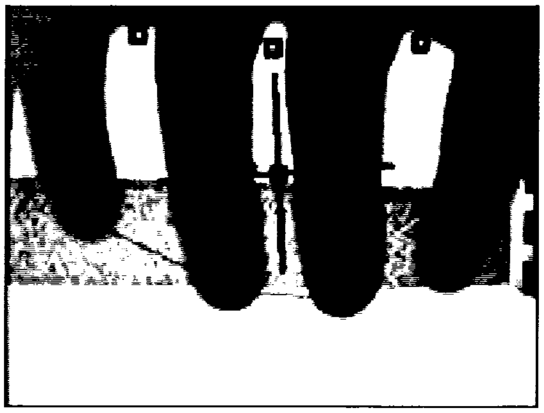

Figure 3.2: The seven interest points and detected orientation of the hand

3.3 Keystroke Detection

3.3.1 Initialization and Normalization

Observations of the system consist of the position and velocity of the four fingertips that are visible in the cameras field of view. Initialization of the system involves recording the rest position of each of the interest points. The coordinate system used for all measurements is defined by the centroid of the seven interest points and the alignment of the hand. The centroid is used as the origin of the coordinate system, and the orientation of the axes is determined by the angle of the ring and middle fingers relative to the image plane. Normalizing the position of the fingertips relative to this coordinate system accounts for shifts in the position and alignment of the camera over time.

3.3.2 Peak Detection

After normalization, the position in polar coordinates, r_i and θ_i of each fingertip relative to its rest position, , and speed v_i, is tracked. Thus at any particular moment of time, each of the four fingertips $\mathrm{x_i}(i = 1..4)$ is defined by a 3 dimensional vector $< r_i, \theta_i, v_i >$.

Keystrokes are detected as maxima in fingertip distances relative to their rest positions. Generally, more than one finger will move for each keypress and the distance reading for all fingers will peak at approximately the same time. Peaks that occur within a five frame window, corresponding to a time window of 0.17 seconds at 30 frames per second, are clustered and interpreted as a single keystroke.

There is considerable noise in the position readings, which would lead to spurious keystroke detections without smoothing the positions over time. A low-pass filter is applied to the fingertip positions to remove noise. For the system to be usable, keystrokes must be detected immediately and thus a causal filter with minimum lag is required. In addition, peak attenuation needs to be minimized in order to maintain accurate position measurements. A first order low-pass filter was used to attain the best compromise among noise reduction, peak attenuation, and delay characteristics.

3.3.3 Hidden Markov Model Based Character Recognition

Hidden Markov models are commonly used in speech recognition and have found growing application in vision research as well. HMMs are particularly well suited to tasks involving the interpretation of noisy, continuous observations into small symbol vocabularies, such as in hand-writing and gesture recognition. For example, HMMs have been used to recognize playback commands for a portable DVD player using embedded accelerometers [27]; head and hand gestures such as pointing, hand waving, and nodding [31], and American Sign Language gestures using a fixed camera [36]. Typically, in machine vision based gesture recognition, a static camera is placed in front of the user and records movement of the hands, arms and head. The recognition system then interprets the sequence of movements as a set of words or commands.

The input to a handwriting or gesture recognition system consists of a continuous stream of data that must be quantified, segmented into meaningful units, and interpreted as a sequence of discrete states. States can represent gestures, letters, words, or commands. In the case that the states are not directly observable and must be deduced, a hidden Markov model can be used to determine the most probable underlying state sequence from a series of observations.

The states in an HMM are inferred on the basis of the transition model and observation model that are associated with each state. The transition model consists of a table of probabilities of moving from one state to any other state. The observation

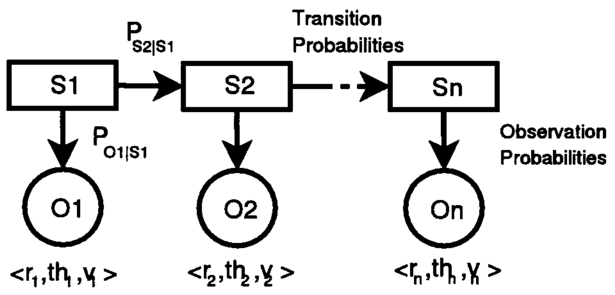

Figure 3.3: An HMM. For keystroke recognition, each state S is a character, the transition model is based on letter bigram probabilities, and the observation model contains finger position and speed vectors

model describes the relationship between system outputs and states, allowing the determination of the most probable state for any given output.

To summarize, HMMs require the following elements:

- A finite number of states Q;
- A set of output probabilities B, which may be discrete or continuous;
- A set of state transition probabilities A.

For keystroke recognition, each state in the HMM corresponds to a single alphanumeric character. At each detected keystroke, the most likely character is determined using the observation and transition models. As described above, the observation model relates system observables (fingertip positions and speeds) with hidden states (the characters that the finger movements represent). The observation model is learned during training by correlating each character in the training set with the output vector observed during its corresponding keypress. Often, two or more keys, such as the 'a' and 'z' keys, have similar observation vectors. The state transition model is used to disambiguate keystrokes. It is based on a language model that is derived from the analysis of a large corpus of text. The observation and transition models are described in detail below.

3.3.4 Observation Model

Observations consist of a set of vectors that describe the position and speed of each fingertip during the instant of a keypress. Each observation is made up of four vectors, one for each finger, each containing three elements; the position of the finger relative to its rest position in polar coordinates, and the speed of the fingertip. In the current implementation, the thumb is not consistently present in the cameras field of view, and is ignored.

Fingertip positions and speeds constitute a continuous observation space, so that the observation model must be expressed in terms of a probabilistic function. In general, the observation symbol probability distribution B = $\{b_j(k)\}$ is denoted by

$$b_j(k) = P\left(o_t = \mathbf{k} | q_t = j\right) \tag{3.2}$$

where $b_j(\mathbf{k})$ is the probability of observation **k** given that the current state is j. In the case of keystroke recognition, each observation **k** consists of 4 vectors, each containing the position and speed of a fingertip at the time of a keypress:

$$\mathbf{k} = \langle \mathbf{k_1}, ..., \mathbf{k_4} \rangle$$

where each $\mathbf{k_i} =< r_i, \theta_i, v_i >$ represents the fingertip position and speed relative to its rest position. Each state j corresponds to an alphanumeric character.

A 3D Gaussian distribution describes the probability of observing **k** for each character j.

$$P(o_t = \mathbf{k} | q_t = j) = N(\mathbf{k} - \mu_\mathbf{j}, \sigma_\mathbf{j}) \tag{3.3}$$

where $\mu_\mathbf{j} =< \mu_r, \mu_\theta, \mu_v >_j$, is the mean finger position and speed for character j; $\sigma_\mathbf{j} =< \sigma_r, \sigma_\theta, \sigma_v >_j$ is the variance; and N is the Gaussian function $\frac{1}{\sigma_j \sqrt{2\pi}} e^{-(\mathbf{k}-\mu_\mathbf{j})^2/2\sigma_\mathbf{j}^2}$

Thus given an observation vector **k**, the probability that it represents character j is inversely proportional to the distance between **k** and the vector corresponding to the character. The means and variances of the vectors associated with each character are learned during a short training stage, and continuously improved during operation. To shorten training time, the variances are set to uniform initial values for each finger.

During training, the user is asked to type a small paragraph of text. As the user types, keypress events are detected, and the current observation vector is associated with the next character in the training text. At the end of the training stage, each character has been typed several times and has several observations associated with

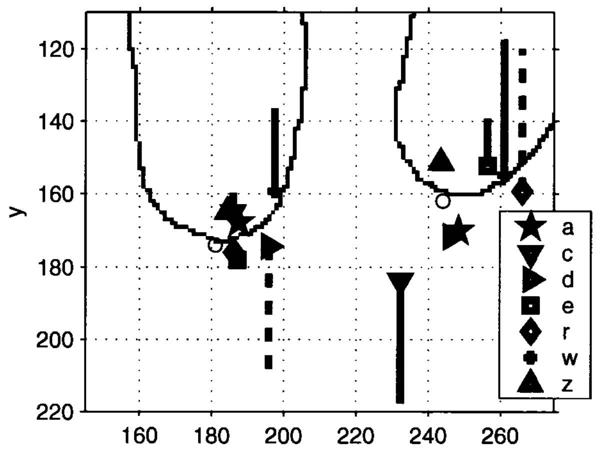

Figure 3.4: The mean position and speed vectors for the middle and index fingers of the left hand. The outline of the middle finger is on the left, and the index finger is on the right. Circles indicate the rest position, and other symbols represent letters. The length of lines corresponds to the finger speed magnitude during a keystroke.

it. On completion of training, each character is defined by the mean and variance of the set of observation vectors that was assigned to it.

Finger movement vectors that have a strong association with a particular character exhibit low variance, and are therefore most heavily weighted in the observation model described above. Similarly, finger movements that are not consistently observed for a specific character have a higher variance, and do not factor as strongly towards identification of the character. For example, if the index finger consistently moves to the same position for a certain character, but the position of the ring finger varies widely, the observation model for that character relies more heavily on the index finger observation. Figure 3.4 shows the observation vectors for the middle and index fingers for a subset of keys.

3.3.5 Character Disambiguation

The statistical regularities of language have been exploited for many purposes including spelling correction in word processors, automatic speech recognition, and predictive text entry in mobile phones. A language model can greatly aid in interpreting noisy or ambiguous text. For example, mobile phones are commonly used to send text messages and retrieve information from the internet. However, the limited number of keys on numerical keypads requires that several characters be mapped to one key, leading to ambiguous inputs. There are two main approaches to determining the desired text: dictionary-based approaches and statistical approaches. Dictionary-based predictive text entry involves searching a dictionary for words that are consistent with an observed sequence of key presses. Often several allowable words are found, and the user must make additional keypresses to select the correct one. In addition, the desired keys are not known until the entire word is typed. Statistical approaches to predictive text entry have attempted to address these issues by taking advantage of the statistics of word and letter sequences. Certain combinations of letters and words are more probable than others. Thus a probabilistic language model can be created by extracting the relative frequencies of letter and word combinations from a large body of text. With the aid of the language model, ambiguous inputs can be resolved based on the preceding sequence of words and letters.

The statistical regularity of the English language has been studied extensively since Shannon's seminal paper [34] describing a game that involves guessing the next letter of an incomplete selection of text. The predictability of the English language can be described in terms of entropy, which is the number of bits of data required to prescribe the next letter if the previous letters are known. For example, if each letter (and the space character) in the alphabet were equally likely, the entropy of the language would be $log_2(27)$= 4.76 bits per character (bpc). Shannon estimated the actual entropy of the English language to be between 0.6 and 1.3 bits per character. Since each letter is highly dependent on previous letters, fewer bits are required to encode each character. The best automated prediction methods have given an upper bound of 1.22 bpc [33]. This means that on average, the next letter can be predicted correctly approximately 40% of the time.

For the device described in this device, the most probable word corresponding to a sequence of keystroke observations is determined in a two stage process using both word frequency and letter pair (bigram) frequency. While a word is being typed, individual keystrokes are disambiguated based on the previous characters using a letter bigram model (LBM). This model increases the accuracy of keystroke recognition substantially relative to a naive system that relies only on observations. However, the recognition rate can be even further improved by incorporating word frequencies into the probability model. This is accomplished by considering the most likely character sequences determined by the letter bigram model, and rescoring them using a word frequency table. The two components of the language model are described below.

A first order hidden Markov model was used for letter disambiguation. The letter bigram transition model estimates the probability of observing each current input character l_n based on the previously observed character l_{n-1}:

$$P(l_n|l_{n-1}) = \frac{C(l_{n-1}l_n)}{C(l_{n-1})}; \tag{3.4}$$

where $C(l_{n-1}l_n)$ is the number of times that the bigram $l_{n-1}l_n$ is observed.

A selection of English text containing 200,000 words [26] was used to determine the letter bigram probabilities. Witten bell discounting [44] was used to account for zero frequency bigrams. The generated model is a table $A = a_{ij}$ of letter pairs and their corresponding probability. Each table entry a_{ij} contains the probability the letter j follows letter i.

After each detected keystroke, the Generalized Viterbi Algorithm [15] is used to determine the most likely character using the observation and transition models. The algorithm calculates the probability of each character by combining information from the two models, and orders the characters by probability. The most likely character is displayed on the screen.

The possible letter sequences can be considered as a trellis with weighted links between the candidate characters for each keystroke. Part of a possible trellis structure is shown in figure 3.5. The strength of the observation probabilities are represented by circle thickness, while the transition probabilities correspond to line thickness. While characters in the top row of the trellis have the highest observation probabilities, the best path through the trellis relies on both the observation and transition models. In this case the best path is d-r-a-w. As each keystroke is detected, the Viterbi algorithm identifies the most probable character by considering:

- The probability of each previous character i

- The transition probability a_{ij} from each previous character to each current character j
- $b_j(O_t) = P(\mathbf{O}|j)$, the probability of the current observation **O**, given that the current character is j

The above probabilities are used to determine the probabilities of every state sequence, or path of characters, at each time step. To do this, the quantity $\delta_t(i)$ is defined as the highest probability along a single path, which accounts for the first t observations and ends in state S_t. For the very first letter in a word, only the observation probability is considered and $\delta_t(i)$ is initialized as follows:

$$\delta_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N$$

where $\pi_i = P[q_1 = S_i]$ is the initial state distribution. Each letter of the alphabet is set to have an equal probability of appearing as the first letter, so that the first keystroke observation vector alone determines the first letter of a word.

For the following observations, the quantity $\delta_{t+1}(j)$ for each state j is determined by the most probable path to that state, as well as the observation probability.

$$\delta_{t+1}(j) = [max \delta_t(i) a_{ij}] b_j(O_{t+1}) \qquad (3.5)$$

An array $\Phi_t(j)$ is used to keep track of the actual state sequence path that maximized 3.5 for each t and j:

$$\Phi_t(j) = argmax[\delta_{t-1}(i) a_{ij}]$$

Finally, when a space token is observed, the sequence is terminated. The set of the m-most likely letter sequences, $\{w_m\}$, is determined is determined by backtracking through the array Φ. The Viterbi algorithm is explained in depth in [30].

The letter bigram model provides a good statistical model of letter sequences within words. However, by incorporating a higher level model of word probabilities the accuracy of the system can be further improved. So, each letter sequence $w_m =< l_1 ... l_n >$ generated by the generalized Viterbi algorithm is rescored according to a word unigram model (WUM). The word model is a table of word frequencies derived from analyzing a large section of text. The probability of word

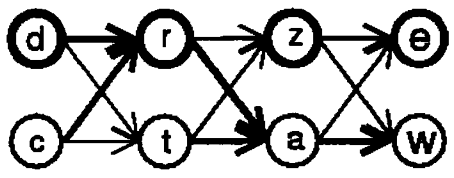

Figure 3.5: A portion of a character trellis. Circle boundary thickness represents observation probability and line thickness represents transition probability

w_m is defined as the number of times the word has been observed divided by the total number of tokens in a body of text.

$$P(w_m) = \frac{C(w_m)}{N} \tag{3.6}$$

where $C(w_m)$ is the count of word w_m, and N is the total number of tokens. The British National Corpus is a 100 million word sample of written and spoken British English derived from a wide variety of sources. In [23] the BNC is tokenized and composed into a list of words ordered by frequency. The word frequency list contains 939,028 unique words, including many proper names and colloquialisms. However, the rapid evolution of language on the internet continuously introduces new words and names that are often not included in large compiled lexica. One method of keeping an up-to-date lexicon is to integrate search query logs into the word list. Live updates of search queries are readily accessible online[1] and can be logged to generate a continuously evolving vocabulary.

The information from the letter bigram model LBM and word unigram model WUM is combined to determine the most probable word. The log probabilities derived from each model are added to determine the final score, and the word with the maximum score is displayed.

$$W^* = argmax_n[\log P(w_n|LBM) + \alpha \log P(w_n|WUM)] \tag{3.7}$$

α is a weighting applied to the word model to determine its relative bearing on the score. W^* is the most likely word as determined by combining the results of the Viterbi algorithm and word model.

The approach described above can be used to disambiguate continuous or discrete inputs that represent symbols that are probabilistically dependent on previous

[1]www.metaspy.com

or future symbols. For example, a similar approach can be used to disambiguate finger movements for glove-based input, virtual glove-based devices, or other types of interaction systems.

Chapter 4

Motion Estimation

4.1 Background

Camera motion estimation is the task of estimating the camera's motion relative to the viewed scene. Motion estimation can be used for motion dampening, video compression, and determination of 3D scene structure. A review of motion estimation algorithms is given in [38]. Motion estimation generally proceeds in two stages:

1. Identify salient features that can be tracked from frame to frame

2. Determine the motion of each feature

The approach used in this thesis relies on finding corner features, determining feature correspondences using Sum-of-Squares Difference (SSD) similarity within a local neighborhood, and using the Random Sample Consensus (RANSAC) algorithm to determine the most likely camera motion. Each stage is described in detail below.

4.2 Corner Detection and Clustering

The Harris corner detector is used to detect interest points in each frame. The detector scores pixels based on their "corner-ness", or having high gradients in two orthogonal directions. The derivation of the Harris detector is presented in [13] and [7].

Corner detection using the Harris Corner Detector proceeds as follows:

1. Find the vertical and horizontal image gradients by convolving the grayscale intensity image with the following kernels $\begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} -1 & 0 & 1 \end{pmatrix}^T$. This leaves us with the horizontal gradient image I_x and vertical gradient image I_y.

2. For each pixel in the intensity image, a local structure matrix C_{str} is found using I_x and I_y:

$$C_{str} = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \tag{4.1}$$

3. The eigenvalues of C_{str} are found by diagonalization of the symmetric matrix, which gives the following:

$$C_{str} = \begin{pmatrix} l_1 & 0 \\ 0 & l_2 \end{pmatrix} \tag{4.2}$$

The eigenvalues l_1 and l_2 can be interpreted in terms of image geometry as follows:

- For a perfectly uniform image, $C_{str} = 0$ and $l_1 = l_2 = 0$.

- For a perfectly black and white edge $l_1 > 0$, $l_2 = 0$, where the eigenvector associated with l_1 is orthogonal to the edge

- For a corner of a black square in a white background, $l_1 >= l_2 >= 0$

Regions of higher contrast correspond to higher eigenvalues. A corner is identified by two strong edges, thus a corner has been detected when the lower eigenvalue is larger than a threshold value.

Since a corner feature in the real world will result in a cluster of several highly scored pixels, a distance constraint is used to separate distinct corner features. First, all pixels exceeding a threshold value are classified as candidate corner pixels. Then, corner pixels are clustered using a simple clustering algorithm, described below.

For each corner pixel:

1. Assign the first pixel to a cluster, and set the cluster centroid to the (x,y) coordinates of that pixel

2. For every other pixel

   - If it is within distance $d = \sqrt{(x_c - x_p)^2 + (y_c - y_p)^2}$ of any other cluster, assign the pixel to the closest cluster, and update the centroid of the cluster $C = \frac{N*[x_c,y_c]+[x_p,y_p]}{(N+1)}$
   - Otherwise, start a new cluster

N is the number of pixels belonging to a certain cluster, (x_c, y_c) are the coordinates of the centroid of the cluster, and the distance d was set to 30.

The pixel at the centroid of each cluster is assigned as the representative pixel for that cluster, and the surrounding 15x15 block of pixels is assigned as a feature point. The feature points are then ranked in order of minimum eigenvalue, which corresponds to corner strength as described in [35] and the forty best corners are chosen.

4.3 Corner Tracking

In each successive frame, the best forty corner features are chosen. Each of the 15 x 15 pixel corner features from the previous frame is then matched to corners within a 50 by 50 pixel window using a SSD similarity measure. Finding the SSD involves aligning the two feature vectors, subtracting the pixel values of one from the other, and summing the squares of each pixel difference. Then, each feature from the current frame is matched to the most similar feature (i.e. minimum SSD) in the previous frame within a 50 x 50 window. Symmetric matches, pairs of features that match both forwards and backwards, are selected as correspondence pairs.

4.4 Motion Determination Algorithm

While the majority of correspondences detected and assigned using SSD within a local neighborhood will be correct, some matches will be incorrect. To filter out the spurious matches, the RANSAC algorithm is used to determine the most probable camera motion. In the current implementation, the image motion is fit to a two

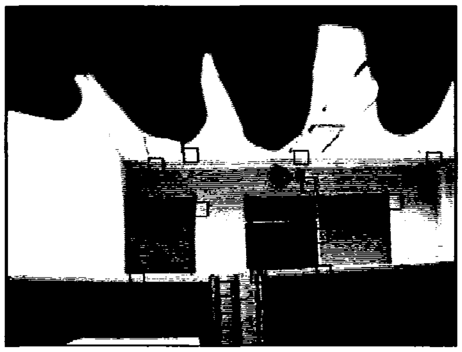

Figure 4.1: Corner tracking in pointer control mode

dimensional movement model; camera motion perpendicular to the image plane is ignored. The RANSAC algorithm determines the most likely movement as follows:

1. Select one of the feature movement vectors at random and use it as the movement model.
2. Count K, the number of movement vectors that fit the model within a given tolerance.
3. If K is bigger than the minimum desired consensus K_{min}, exit successfully
4. Otherwise, repeat the above steps for I iterations or until a correct model is found

In practice, the parameters K_{min} and the number of iterations I are set to attain a balance between accuracy and computation time. The higher K_{min} and I are set, the more likely the RANSAC algorithm will determine the correct motion. The relationship between parameter values and the probability of finding the true correspondence is described in [14].

4.5 Pointer Movement and Button Presses

After determining the most probable magnitude and direction of motion, the movement vector is scaled and applied to the pointer position. During training, the finger position corresponding to a mouse button click is recorded. As long as this finger position is maintained, the virtual button is held in the depressed position.

Chapter 5

3D Interaction

5.1 Background: Virtual and Augmented Reality Interaction

5.1.1 Virtual Reality

Virtual reality describes an immersive artificial environment that entirely replaces the visual and auditory sensations of the real world with a synthetic one. Generally, a head worn display provides slightly different images of a 3D scene to each eye, preserving stereo parallax, and head tracking allows the user to view the virtual world naturally by constantly updating the displays as the head moves. An alternative to head mounted displays is CAVE Automatic Virtual Environments (CAVE) [5], in which four walls surrounding the user display a 3D scene that is updated according to the users movements.

5.1.2 Augmented Reality

In contrast to the complete immersion provided by virtual reality, augmented reality superimposes virtual objects within the real world. A camera is used to capture the environment, and the 3D structure of the scene and camera motion is determined in real-time. Using this information, synthetic objects can be inserted into the video stream and made to appear as part of the environment.

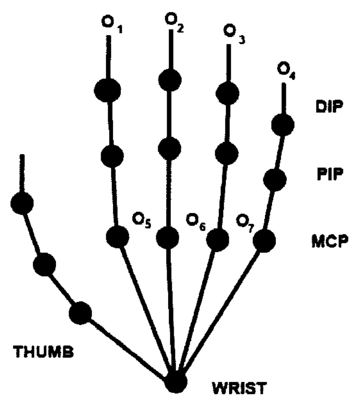

Figure 5.1: The simplified hand model

5.1.3 Virtual and Augmented Reality Interaction

Currently, one of the principle methods of virtual reality interaction is by using a keyboard and mouse or joystick. These methods allow one to easily navigate within the environment, but complex manipulation tasks with virtual objects is difficult. For example, picking up an object, and moving, rotating, or deforming the object in three dimensions is difficult using traditional 2D input devices.

A more natural method of 3D interaction is to track the position of the hands and fingers, allowing the hands themselves to be used as input devices. Observations of the hand and fingertip position within the video stream are interpreted using a kinematic model to estimate the flexion and adduction of each finger, and the wrist. Some of the information in this chapter is also described in [2].

5.2 Hand Model

The human hand is a highly articulated structure, with 27 degrees of freedom. Articulated objects are often modeled using linkages and joints, and described by the lengths of each link and degrees of freedom of each joint. Each finger contains the following joints: metacarpophalangeal (MCP), proximal interphalangeal (PIP) and distal interphalangeal(DIP). A simplified model of the hand joints and bones, that does not include the bones of the palm or wrists, is shown in Figure 5.1 below.

As described in [8], each of the four fingers has four degrees of freedom; the DIP and PIP each have one, and the MCP has two, including one for flexion and one for abduction. The thumb has five degrees of freedom; one for the interphalangeal (IP) joint, and the MCP and trapeziometacarpal (TM) joint each have two for flexion and abduction. The remaining six degrees of freedom are due to the position (3 DOF) and orientation (3 DOF) of the wrist. The high number of degrees of freedom makes tracking and visual pose estimation of the hand difficult. Like glove-based approaches, the device described in this paper separates the task into two subtasks; hand position estimation and finger pose estimation. Fortunately, human finger motion is highly constrained. A simplified hand motion model is described in the next section.

5.3 Simplified Finger Kinematic Model

As described above, each finger has three joints, with a total of four degrees of freedom. However, in the normal range of hand motion, each finger joint is not moved independently of the other joints. In [19], hand motion constraints are divided into two types: Type I Constraints refer to the limits of the range of finger motion due to hand anatomy.

$$\begin{aligned} 0 < q_{mcp-y} &< 90 \\ 0 < q_{pip-y} &< 110 \\ 0 < q_{dip-y} &< 90 \\ -15 < q_{mcp-x} &< 15 \text{ (adduction/abduction)} \end{aligned}$$

The above comparisons describe the range of possible bend angles for the first (mcp), second (pip), and third (dip) joints of the fingers, and are approximately the same for each finger. For most people, the mcp joint can actually bend slightly backwards, to a minimum of approximately -30 $^\circ$ relative to the top of the hand, not 0 $^\circ$. The subscript y refers to bending along the major axis of the hand, and x refers to the side-to-side (adduction/abduction) motion possible at the mcp of each finger.

Type II constraints are the limits imposed on joints during motion, reflecting the dependence of finger bend angles. They can be divided into interfinger constraints, those that relate the bend angles of one finger to the others, and intrafinger constraints between joints of the same finger. Interfinger constraints are not very strong, and thus only intrafinger constraints are considered here. In [19], the following commonly adopted constraint is used that relates DIP angle to PIP angle for each finger:

$$q_{DIP} = \frac{2}{3} q_{PIP} \tag{5.1}$$

In natural hand movements, the PIP bend angle remains approximately the same as the MCPx bend angle, allowing the adoption of an additional constraint:

$$q_{PIP} = q_{MCP-y}. \tag{5.2}$$

A similar constraint is adopted by glove-based devices, which contain a single flex sensor per finger, and in several vision based hand trackers[8]. No type II finger adduction/abduction constraints are assumed.

Due to the position of the camera, additional constraints can be assumed regarding wrist position relative to the camera. Since the camera is worn just under the wrist, the camera remains aligned with the fingers regardless of wrist roll. However, the wrist may pitch and yaw relative to the camera. Thus the number of degrees of freedom captured by the system is two for the adduction/abduction and bend angles of each finger (8DOF), plus wrist pitch and yaw for a total of 10 DOF.

5.4 Finger Position Estimation with Inverse Kinematics

As described in Chapter 3 and shown in figure 5.2 below, the observations of finger position include fingertip positions and inter finger joint position in two dimensional image coordinates.

From these 7 observations, we can deduce a reasonably good 10 DOF estimate of hand position by applying human finger bend angle constraints. The simple inverse kinematic model is described below.

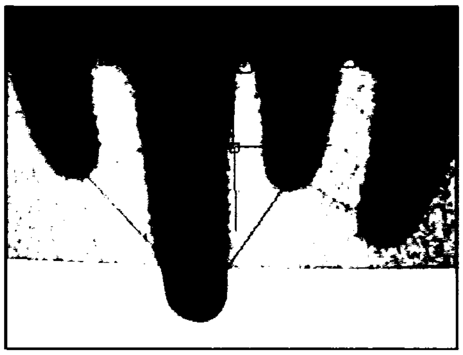

Figure 5.2: Fingertip position observations

Using the distance v between the fingertip and MCP joint in the image (see figure 5.3), the MCP, PIP and DIP bend angles can be estimated for each finger. When the fingers are straightened, the distance v between the MCP and fingertip position will be near zero, as can be seen from figure 5.3. At the point of maximum fingertip MCP distance, the MCP angle is approximately 60 degrees. For all intermediate distances, the MCP angle, and consequently the PIP and DIP angles, vary linearly with distance. The relationship can be described with the following equation:

$$K_1 v_y = q_{MCP-y} = q_{PIP-y} = \frac{3}{2} q_{DIP-y} \tag{5.3}$$

Where $K_1 = [60^\circ - 0^\circ]/[240 pixels - 0 pixels] = \frac{1}{4}$; and 240 pixels is the image height.

Finger Adduction/Abduction is described similarly as a proportion to horizontal image distance:

$$K_2 v_x = q_{MCP-x} \tag{5.4}$$

Where $K_2 = [30^\circ - 0^\circ]/[80 pixels - 0 pixels]$, 30 ° is the range of maximum horizontal MCP adduction/abduction, and 80 pixels is the corresponding change in fingertip movement in image coordinates. The model is illustrated in figure 5.3.

Of course, these parameters would need to be adjusted for varying camera field of views and image resolutions. Hand motion capture and the generated model is shown in figure 5.4.

Wrist pitch is estimated on the basis of average y coordinate value of the MCP position of all fingers; a pitch of 0 ° was found to correspond to an average MCP position of approximately 30 pixels from the top of the image, with positive pitch

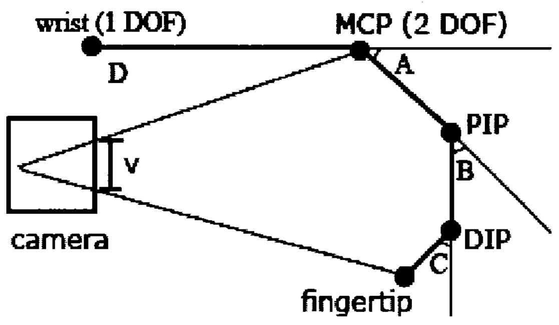

Figure 5.3: Finger pose estimation using a simple hand model

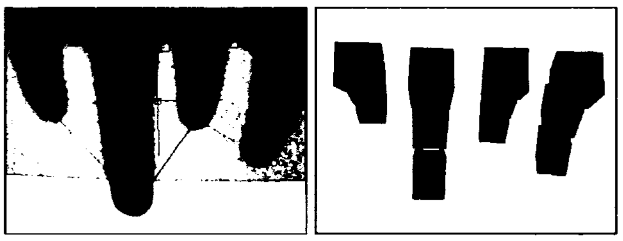

Figure 5.4: Finger motion capture and generated 3D model

corresponding to lower MCP positions and negative pitch corresponding to higher values. The field of view of the camera used limited wrist pitch measurements to a range of -15 ° to 15 °.

5.5 Hand Tracking using ARToolkit

For interacting with a virtual object, hand position must be tracked as well. The most popular method of tracking hand position in three dimensions is small magnetic sensors such as those sold by Polhemous or ascension. These types of sensors are commonly bundled with glove-based hand motion trackers. Another alternative is to use gloves embedded with reflective markers that can be tracked using multiple cameras. Such systems use two or more calibrated cameras to determine hand and finger pose in 3 dimensions. Ultrasonic sensors provide a less expensive option, but also less accurate, and generally provide position information but lack orientation information. The approach used in this thesis involves tracking wristworn ARToolkit markers, whose position and orientation can be tracked using a camera. The concept of using handworn ARToolkit markers for hand tracking is discussed

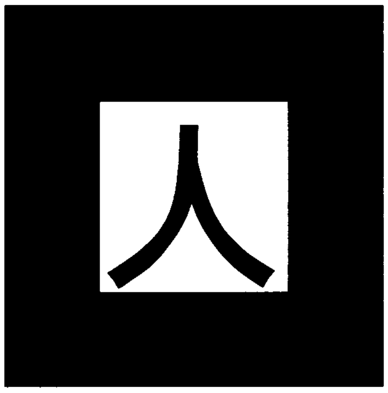

Figure 5.5: An ARToolkit marker

in [41] and [29]. In [41], users wear a glove that includes ARToolkit markers as well as a pressure sensitive foil worn within the thumb for input. Similarly in [29], gloves with metallic contacts on the fingers are worn along with ARToolkit markers that are tracked by a headworn camera. The gloves allow contact between the thumb and one of the fingers to be used as binary input buttons. This thesis follows a similar approach, except that rather than requiring pinch gloves, input is by means of high-resolution finger tracking based on wrist worn cameras.

5.6 3D Hand Position Estimation

The ARToolkit[21] is a marker-tracking library based on simple fiducial markers that are designed to be tracked in three dimensions. ARToolkit markers consist of a black square border, and a simple shape within the border that is used for template matching when several markers are visible at the same time. A sample ARToolkit marker is shown in figure 5.5.

A single camera is used to track the marker, and the ARToolkit returns a 6 DOF position and orientation matrix representing marker position relative to the camera. A complete description of the ARToolkit library can be found in [21]. ARToolkit can also handle the case when more than one marker is visible at the same time. In this case, each of the markers orientation and position is returned. Multiple markers provide a mechanism for interaction, allowing one marker to be used for the display of virtual objects, and the other to be worn or held as a gestural input device. For

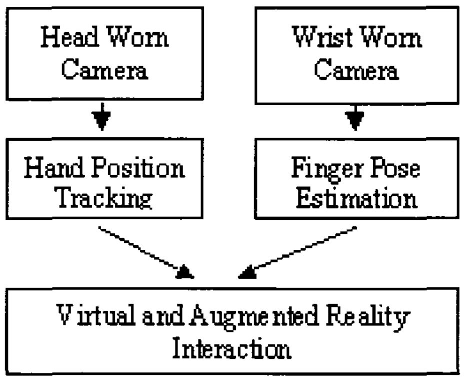

Figure 5.6: Augmented reality interaction overview

example, in [22], a set of virtual objects are overlaid over one marker, while a second marker can be used as a paddle to push virtual objects around. In this thesis, we extend the paradigm to infuse additional functionality into the second marker, by tying it to the UbiHand finger tracking system. When an ARToolkit marker is worn on the users wrist, a virtual hand is overlaid over the users hand. Finger tracking information from the wrist worn camera is used to align the virtual fingers with the users actual fingers as they move. In addition, the relative position of the wrist worn marker to another static marker is continuously tracked. This allows the user to interact with and manipulate virtual objects using their own hand.

5.7 Augmented Reality Interaction - Virtual Objects and Collision Detection

The hand and finger pose estimation is integrated to a complete model of hand and finger position. This information is used to display a virtual hand over the users hand in the field of view of the headworn camera as shown in figure 5.7.

Virtual objects are overlaid onto other static markers that are visible within the field of view of the head worn camera. The position of both the wrist worn marker as well as the static markers relative to the head worn camera is tracked in real time. Collision detection between virtual objects and the hand is determined by a simple proximity test; if a fingertip comes within a certain distance of the centroid of a virtual object, a collision is registered, triggering the virtual object to respond in some way. For example, in figure 5.7, a fingertip can be used to push a virtual robot, that

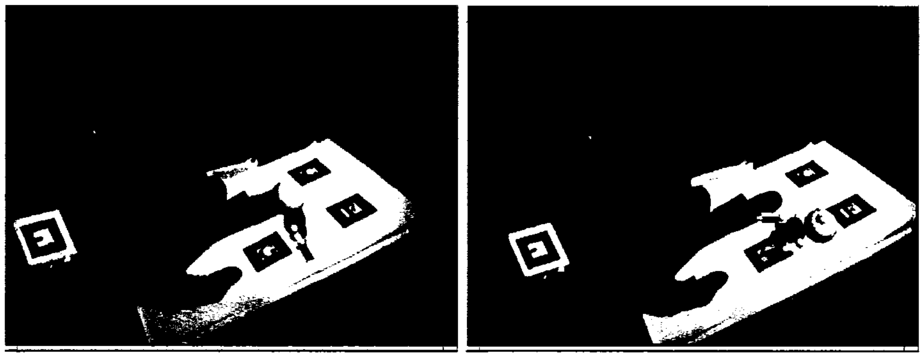

Figure 5.7: Augmented reality interaction

falls if it has been touched, and begins to walk faster when it gets up. An overview of the system organization is illustrated in Figure 5.6, and the experimental setup is described in more detail in Chapter 6.

Chapter 6

Experimental Results

A prototype system was implemented on a single wrist-worn color wireless camera. The camera captures images at a frame rate of 30 fps at a resolution of 320x240 pixels. The receiver was connected to a PC equipped with a video capture card, and processing was done on a 1533 MHz AMD Athlon system. The prototype system is shown in figure 2.1. During the evaluation, the camera was worn on the left hand and the training data was displayed on a monitor. Testing of the keystroke input and pointer control modes was done in a well-lit indoor environment.

6.1 Typing

The letter bigram and word unigram models were derived from the British National Corpus as described in section 3.3.5. For the purposes of training and testing one-handed input, a selection of text that can be typed using only the left hand was extracted by selecting words consisting exclusively of characters from the following set

$$Q, W, E, R, A, S, D, F, Z, X, C, V$$

A touch-typist was trained on a 300 word test set over a period of one hour. Each character was typed approximately 25 times, and the mean and variance of each finger movement vector was calculated after the completion of the training stage.

After training, a 200 word test file was used to measure recognition accuracy and speed. In the testing stage, after each keystroke is detected, the most likely character was displayed on the screen. Finger movements incorrectly recognized as

Table 6.1: Character recognition accuracy

Language Model	No Language Model	Letter Bigram	Letter & Word Model
Accuracy [%]	67	84	90

keystrokes (false positives) were classified as errors, while undetected keystrokes were retyped. The right hand was used to click a mousebutton to signal the completion of each word, at which time the most likely word was displayed on the screen.

6.1.1 Character Recognition Accuracy

Table 6.1 describes the character recognition accuracy with and without the help of the language models. As the table shows, the observation model alone provides a character recognition rate of 67%. Character recognition errors are a result of the ambiguity and high variance of keystroke movements. In general, sets of characters that are typed with the same finger on a qwerty keyboard, such as {e,d,c} exhibit similar observation vectors and are easily confused. Of the incorrectly recognized words, 16 were due to character substitution errors, and 6 were due to spurious keystrokes. Several methods could be used to improve the accuracy of the observation model. For example, including angular direction rather than just speed v in the observation model provides more information to help discern keystrokes. Nevertheless, while the observation model alone does not always recognize the correct character, the letter bigram transition model improves the recognition rate substantially to 84%. The word model further improves the character recognition rate. Thus in some cases, the character is misidentified immediately after a keystroke, but is corrected as soon as the word is completed. In the current implementation, the end of each word is indicated with a mouse button press. Another alternative is to assign a chorded finger movement to represent a space character, or a thumb movement if a camera with a wider field of view is used. In practice, usability can be improved by displaying the three most probable words and allowing the user to choose the correct one in very ambiguous cases. In addition, explicitly including the probability of misdetected keystrokes in the observation model is expected to further improve the accuracy of the system.

6.1.2 Input Speed

A typing speed of 14 words per minute was achieved on the prototype system. The smoothing filter described in section 3.3.2 is one source of lag, introducing a delay of approximately three frames (0.1 s) between a keystroke and its detection. Missed keystrokes, which need to be retyped, are another source of delay. In the current implementation, a detected keypress is indicated by displaying the most likely character on the screen. The absence of the tactile feedback is another possible limitation to input speed on virtual keyboard and glove-based devices. Nevertheless, the input speed for the reduced character set tested here is comparable to other input methods for portable devices. In [43], the input speed and accuracy for several mobile phone text entry methods is compared. The addition of acoustic feedback after each detected keypress is expected to improve input speed.

6.2 Pointer control

The speed and control of pointer movement were tested qualitatively. The finger position assigned to pointer control mode was an open hand as shown in figure 4.1 and was detected consistently. In each frame, 20 corner features were selected to be tracked. On average, 70% of the selected features are matched from one frame to another. Accurate control can be achieved with smooth hand movements. However, rapid hand movements hinder accurate feature tracking, resulting in pointer position lag. In addition, feature correspondence errors sometimes result in erratic pointer motion. Several improvements can be made to enhance performance. For example, using more unique feature descriptors will improve feature matching between frames. In addition, while corner features are appropriate for tracking in indoor environments, other feature descriptors are more suitable for less structured outdoor environments. Inertial measurements can also be used for ego-motion estimation, as in [27]. Integrating vision and inertial sensor based methods can be used to further improve accuracy. An overview of sensor fusion techniques for motion estimation is given in [37].

6.3 Augmented reality interaction

When used as a tool for AR interaction, the system consists of two major components: finger pose estimation and marker based hand tracking. Both components were first tested independently, and then as a unit using an AR based game.

6.3.1 Finger Pose Estimation

The accuracy and robustness of finger pose estimation was tested to determine the accuracy at varying bend angles. The fingers were tracked as described in Chapter 3, and a 3D model of the fingers was displayed in a window. The constant multiplier K in equation 5.3 that is used to obtain the bend angles of the virtual fingers was fine-tuned to minimize the discrepancy between actual and virtual bend angles. The maximum and minimum bend angles of the first joint of the index finger for accurate recreation are described in Table 6.3.1.

Minimum flexion occurs when the hand is fully opened. The fingers are able to bend backwards to the point where they are out of the field of view of the wrist worn camera. Maximum flexion occurs when the hand is in a fist. Maximum Abduction for the index finger occurs when the fingers are brought together, and maximum adduction occurs when the fingers are spread apart so the horizontal distance between each finger is at a maximum.

The limitations on minimum flexion are due to the fact that once the knuckle angle dips below zero $^\circ$, the fingertip position is out of range of the cameras field of view. Similarly, there are limitations on maximum measurable adduction since finger adduction greater than 25 $^\circ$ for the index finger moves the tip of the finger past the horizontal edge of the image. In this case, however, a camera with a wider field of view, or worn further back on the wrist would eliminate this limitation.

	Physiological Maximum (Degrees)	**Maximum Measurable** (Degrees)
Minimum Flexion (Hand Opening)	-30	0
Maximum Flexion (Hand Closing)	90	60
Maximum Abduction	-15	-10
Maximum Adduction (Fingers Outstretched)	35	25

Table 6.2: Minimum and maximum measurable bend angles for the first joint of the index finger

The limitations on finger abduction are due to the fingertip recognition algorithm itself; when abduction is at a minimum amount, all fingers are touching one another, and detecting the maxima of the finger contours becomes inaccurate. Finally, the limitation on flexion measurement is due to ambiguity in the fingertip position when the hand is in a fist. Once the finger is bent past a certain angle, fingertip position begins to move back up in the image, which the algorithm interprets as decreasing flexion rather than increasing flexion. Resolving this ambiguity would require a more specific finger tracking algorithm that could differentiate between views of the front and back of each fingertip.

Within the bend angle limits prescribed above, however, finger position estimation is quite accurate for natural finger bend angles. That is, when comfortably opening and closing the hands, the flexion and adduction of each finger can be determined from the observations of fingertip and phalange positions.

6.3.2 Tool Selection and Usage

Simple gestures were used to switch between various virtual tools, such as pencil, eraser, and scissors. For example, quickly moving the index finger upwards and back down turned on pencil mode, and a downards pinky keystroke started

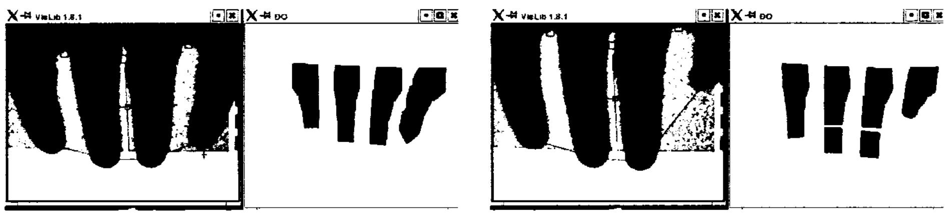
Figure 6.1: Augmented reality pencil and eraser

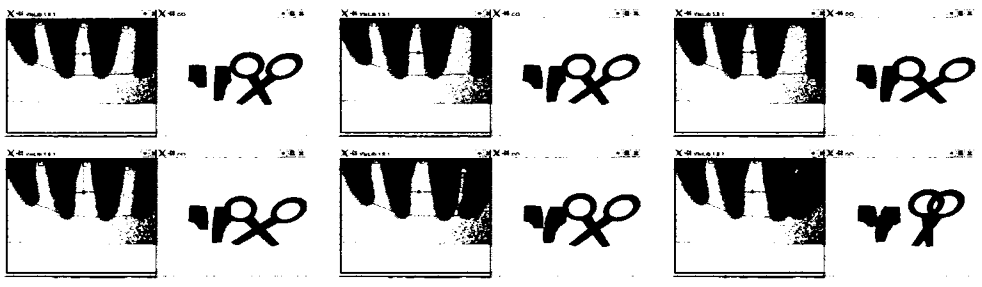
Figure 6.2: Augmented scissors tool. There is a one frame delay between finger movement and model update

eraser mode. Pencil and eraser mode simply changed the appearance of the index finger of the virtual hand. In scissors mode, a pair of scissors was overlaid over the index and middle fingers, and opened in closed in response to their angles of adduction/abduction.

3D drawing functions were not implemented so it is difficult to determine the usability of the augmented reality tools for 3D design. However, from a qualitative perspective, controlling the pencil and eraser using the index finger seemed quite natural, as did moving the fingers in a cutting motion to emulate scissors.

6.3.3 Marker tracking for hand position tracking

A single 3x3 cm ARToolkit marker was worn on the wrist and used for tracking hand position. The relative lengths of each part of the hand were set based on measurements of the users hand. Then, the size of the virtual hand with respect to marker size was calibrated manually, until the position and size of the overlaid virtual hand matched the users hand.

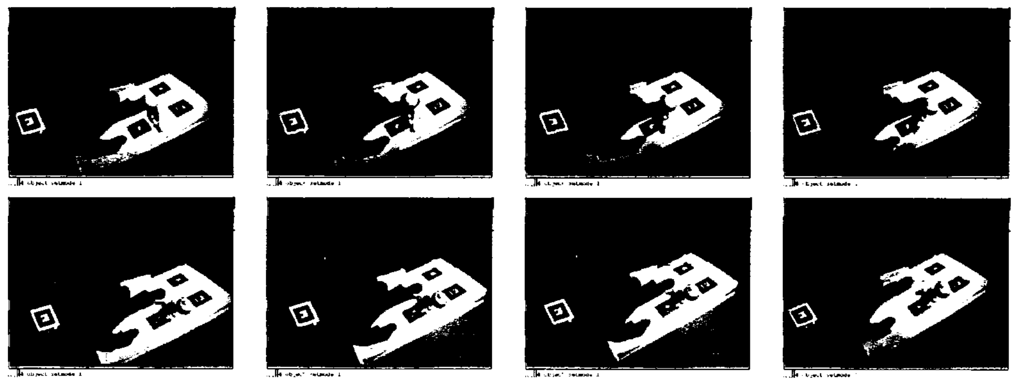

Figure 6.3: Interacting with an augmented reality character

6.3.4 Interaction with Virtual Objects

A simple augmented reality game was created to test the system. Figure 6.3 illustrates the setup. Since an HMD was not available, a webcam was worn as a pendant, and the user viewed the scene in a monitor. A set of markers printed on a single letter-sized sheet of paper was used as the game platform upon which a virtual robot was overlaid. The robot walks randomly across the platform, turning whenever it reaches an edge of the piece of paper. The wrist worn marker is used to determine hand position and orientation, and a virtual hand is drawn over the users hand. The object of the game is to knock over the robot by pushing it over with a finger. Whenever the robot is knocked over, it gets back up and starts walking at a faster pace. Some screenshots of the game that show the virtual robot being knocked down are shown in figure 6.3.

Using the device for augmented reality interaction is quite intuitive. Virtual tools overlaid over images of the actual environment and the users own hands make spatial navigation much simpler than within a completely virtual environment. The main difficulties include errors in marker tracking, and properly maintaining alignment of the virtual hand with the users hand.

Chapter 7

Conclusions

The field of wearable computing is in its infancy. The natural integration of computing power into body-worn articles allows constant, immediate access to information and communication. The implications of continually available computing are not obvious, but may revolutionize information sharing and communication in a similar fashion as the Internet. Indeed, the technology of wearable computing is a natural extension to the Internet, allowing always-on access to information that will complement our human memories. However, there are still several technological hurdles that must be overcome before wearable computers will be accepted on a large scale.

The first requirement for an acceptably unobtrusive wearable computer system is a compact wearable display. Recent progress in small head mounted displays and optical see through displays indicates that the technology is nearing fruition.

Performance requirements include sufficient network speeds to provide location based information to mobile users in real time, and mobile computers with the ability to quickly process and display context sensitive information. Current wireless networking speed and mobile computer processing power is sufficient for the task.

The next requirement is an unobtrusive and easy-to-use method of input that can be used on the move. So far, mobile human-computer interaction research has focused on methods of inputting text into mobile computers that require the users focused attention. Natural methods of input, that allow seamless interaction with a wearable computer are still scarce, and it is this third requirement that is addressed in this thesis.

7.1 Contributions of this thesis

This thesis presents a new method of interacting with wearable computers, based on hand gesture recognition. The input device is multi-modal; the algorithms described in this thesis describe input methods for text input, pointer control and 3D interaction. Further more, wrist worn cameras provide for a compact and unobtrusive method of hand gesture recognition. The main contributions of this thesis are as follows:

- Development of algorithms for robust finger segmentation and keystroke detection
- Application of a Hidden Markov Model for recognizing and disambiguating hand gestures to allow non-chorded keystroke input
- Proof-of-concept development of vision based feature tracking as a method for hand motion estimation and pointer control
- Development of a prototype system that demonstrates the wrist worn camera as a feasible input device for multi-modal wearable HCI
- Description of how such a device can be used for 3D hand motion capture and augmented reality interaction.

The concept of using wrist worn finger trackers for input has been described in two main research papers; The first is Andrew Vardy's paper [40], that introduced the idea of wearing a camera under the wrist to track the fingers. His system could recognize seven possible gestures based on the relative height of each finger in the camera image. Bruce Howard's Lightglove[16] takes a similar approach, but relies on several infrared range sensors worn under the wrist rather than a camera. Prototypes of the Lightglove have been created, and the company is near to commercializing the product.

The main requirements that will enable a wrist worn input device to gain popularity are size, comfort, cost, functionality, accuracy, and ease-of-use. The requirements are discussed below, along with how this work has addressed these concerns.

7.1.1 Size, Comfort and Cost

Given that most mobile computers cost a few hundred dollars, it is reasonable to assume that cost for an input peripheral should not exceed \$100. Given the decreasing costs of CMOS image sensors and mobile DSPs, it is realistic to assume that a commercial version of the system described in this thesis can be developed and sold in this price range.

In addition, for users to accept and use a new input device, it should be as small as possible and as close in appearance as possible to currently worn accessories. Once wearable displays are indiscernible from sunglasses, they are more likely to be worn. Similarly, once wearable input looks and feels like a normal wristwatch, it is more likely to be accepted.

The prototype developed meets this requirement, demonstrates a great improvement in wearability relative to glove based devices, and could be further miniaturized with newer technology. CMOS image sensors that meet the size requirements are available on the market today, and the algorithms used for finger segmentation and gesture recognition have been designed to be amenable to implementation on a mobile DSP.

7.1.2 Functionality

There is a pressing need for an input device that provides fast text input and accurate pointer control for mobile computers. The main requirement for a wearable input device is the ability to input text quickly and accurately.

The finger gesture recognition and disambiguation system described in this thesis provides a method for input that allows rapid input of text at close to 1 keystroke per character. This feature removes the requirement of using slow and hard-to-learn chorded input schemes, and represents a significant extension to glove-based and virtual-glove based input devices which generally limit input to a small vocabulary of gestures. While the text input speed of the system currently lags that possible with a standard keyboard, the performance is comparable to other portable input methods.

An intuitive method of pointer control based on hand movement has been integrated into the device as well, which allows for fast switching between typing and pointer control. Using inertial sensors for improved motion estimation is the most likely approach for a practical implementation.

7.1.3 Accuracy and Robustness

Consumers are resistant to adopt any input technology that is not 100% predictable. For example, open vocabulary speech recognition systems have not been widely adopted even though recognition accuracy rates for continuous speech has surpassed 98% for some commercial systems. Attaining perfect accuracy is the most challenging aspect of non-tactile human computer interaction. Another shortcoming of current speech recognition systems that prevents widespread consumer acceptance is the difficulty of correcting mistakes. For gesture recognition systems to succeed, a fast method of correcting recognition mistakes is vital.

This thesis has put forward the idea of using a probabilistic language model to allow a larger vocabulary of understood gestures, as well as increasing recognition accuracy for smaller vocabularies. Refinements to the recognition algorithms, such as using higher level language models, would further improve performance.

7.1.4 Ease of Use, Training Time

One significant barrier to the adoption of gestures or speech for HCI is the time required to train the recognition system. The device described in this thesis minimizes the learning curve by allowing QWERTY finger-to-key assignments rather than requiring chorded input. Another usability improvement is that it allows text input while on the move; the hands do not have to be held in a specific position during typing, so that input can be entered while the user is standing or walking with their hands at their sides. This makes it a natural fit for a wearable computer system.

7.2 Future Work

Several improvements can be made to each element of the system. For example, hand and finger recognition can be made more robust by implementing adaptive skin color classification. Work has been done to accurately detect varying ranges of skin color and perform accurate skin color extraction in all illumination conditions. Furthermore, contour tracking algorithms using deformable templates or dynamic contours [17] could be applied to more accurately track the edges of each finger. The Condensation (Conditional Density Propagation) algorithm [18] is a probabilistic algorithm that can be used to track contours in cluttered environments. Alternately, in [39], the circular Hough transform is used to track fingertip position for a virtual drawing application. Incorporating an infrared light, and using an infrared filtered camera lens is another approach to finger segmentation that would be robust to illumination changes. For example, in [4], a flashing infrared LED is used to segment objects based on distance from the light, and a similar approach could be used here by integrating a flashing IR LED beside the wrist worn camera.

Many improvements to pointer control are also possible. The feature tracking approach described in this thesis uses a two dimensional model of camera movement. Structure-from-motion techniques can be used to extend motion tracking to three dimensions. In [6], real-time structure-from-motion is achieved by tracking corner features and determining 3D position using a Kalman filter. Using more distinctive features rather than corners, as in [24], could enable more accurate tracking, and smoother pointer control. A less computationally expensive approach is to use inertial sensors such as accelerometers to aid motion estimation. This approach also has the advantage of working in low light conditions.

Overall, the approach of using wrist worn cameras as a wearable input device is promising, and could be developed into a practical realization by incorporating some of the suggestions discussed.

Bibliography

[1] F. Ahmad and P. Musilek. A keystroke and pointer control input interface for wearable computers. In *Fourth IEEE International Conference on Pervasive Computing and Communications*, pages 593–600, 2006.

[2] F. Ahmad and P. Musilek. Ubihand: a wearable input device for 3d interaction. In *ACM SIGGRAPH 2006 Research posters*, 2006.

[3] A. Albiol, L. Torres, and E. J. Delp. Optimum color spaces for skin detection. In *Proceedings of IEEE International Conference on Image Processing*, pages 122–124, 2001.

[4] R. Bolle, J. Connell, N. Haas, R. Mohan, and G. Taubin. Us5631976: *Object imaging system.* Technical report, IBM Exploratory Computer Vision Group, 1997.

[5] C. Cruz-Neira, Daniel J. Sandin, T.A. DeFanti, R.V. Kenyon, and J.C. Hart. The cave: audio visual experience automatic virtual environment. In *Communications of the ACM*, pages 64 – 72, 1992.

[6] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision*, volume 2, page 1403, 2003.

[7] K. G. Derpanis. *The Harris Corner Detector.* Technical report, York University, 2004.

[8] A. Erol, G. Bebis, M. Nicolescu, R. Boyle, and X. Twombly. A review on vision-based full dof hand motion estimation. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, page 75, 2005.

[9] M. Fukumoto and Y. Suenaga. FingeRing: A full-time wearable interface. In *Conference on Human Factors in Computing Systems*, pages 81–82, 1994.

[10] A. Fusiello. *Elements of Geometric Computer Vision: Pin-hole camera geometry*, 2005.

[11] M. Gandy, T. Starner, J. Auxier, and D. Ashbrook. The gesture pendant - a self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *Proceedings of the 4th IEEE International Symposium on Wearable Computers*, page 87, 2000.

[12] J. Gil and R. Kimmel. Efficient dilation, erosion, opening, and closing algorithms. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1606 – 1617, 2002.

[13] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, volume 15, pages 147–151, 1988.

[14] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521623049, 2000.

[15] T. Hashimoto. A list-type reduced constraint generalization of the viterbi algorithm. In *IEEE Transactions on Information Theory*, volume 33, pages 866–876, 1987.

[16] B. Howard and S. Howard. Lightglove: Wrist-worn virtual typing and pointing. In *Fifth International Symposium on Wearable Computers*, page 172, October 2001.

[17] M. Isard and A. Blake. *Active Contours.* Springer, 1998.

[18] M. Isard and A. Blake. Condensation conditional density propogation for visual tracking. In *International Journal of Computer Vision*, volume 29, pages 5–28, 1998.

[19] J.Lin, Y. Wu, and T.S. Huang. Modeling the constraints of human hand motion. In *Proceedings of the Workshop on Human Motion (HUMO'00)*, page 121, 2000.

[20] M. Kalsch and M. Turk. *Keyboards without Keyboards: A Survey of Virtual Keyboards.* Technical report, University of California, Santa Barbara, 2002.

[21] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality. In *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, 1999.

[22] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual object manipulation on a table-top ar environment. In *Proceedings of the International Symposium on Augmented Reality (ISAR 2000)*, pages 111 – 119, 2000.

[23] A. Kilgarriff. Putting frequencies into the dictionary. In *International Journal of Lexicography*, pages 135–155, 1996.

[24] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110, 2004.

[25] I. MacKenzie and R. Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. In *Human Computer Interaction*, volume 17, pages 147–198, 2002.

[26] M. Marcus, B. Santorini, and M. Marcinkiewicz. Penn treebank corpus of wall street journal texts.

[27] J. Mntyjrvi, J. Kela, P. Korpip, and S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *International Conference on Mobile and Ubiquitous Multimedia*, volume 83, pages 25–31, 2004.

[28] W. Piekarski, B. Avery, B. Thomas, and P. Malbezin. Integrated head and hand tracking for indoor and outdoor augmented reality. In *Proceedings of the IEEE Virtual Reality Conference*, page 11, 2004.

[29] W. Piekarski and B. Thomas. Using artoolkit for 3d hand position tracking in mobile outdoor environments. In *Proceedings of the First IEEE International Workshop on the Augmented Reality Toolkit*, 2002.

[30] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[31] G. Rigoll, A. Kosmala, and S. Elckeler. High performance real time gesture recognition using hidden markov models. In *Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human Computer-Interaction*, volume 1371, pages 69–80, 1997.

[32] H. Roeber, J. Bacus, and C. Tomasi. Typing in thin air: the canesta projection keyboard - a new method of interaction with electronic devices. In *CHI 2003 Extended Abstracts on Human Factors in Computing Systems*, pages 712–713, 2003.

[33] R. Rosenfeld. A maximum entropy approach to adaptive statistical modeling. *Computer Speech and Language*, 10:187–228, 1996.

[34] C. Shannon. Prediction and entropy of printed english. In *Bell System Technical Journal*, pages 50–64, 1951.

[35] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2–11, 1994.

[36] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *Proceedings of the International Workshop on Face and Gesture Recognition*, pages 189–194, 1995.

[37] D. Strelow and S. Singh. Online motion estimation from image and inertial measurements. In *Proceedings of the 11th International Conference on Advanced Robotics*, 2003.

[38] T.Y. Tian, C. Tomasi, and D. J. Heeger. Comparision of approaches to egomotion computation. In *Proceedings of CVPR*, pages 315–320, 1996.

[39] N. Ukita and M. Kidode. Wearable virtual tablet: Fingertip drawing on a portable plane-object using an active infrared camera. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 169–176, 2004.

[40] A. Vardy, J.A. Robinson, and L-T Cheng. The wrist-cam as an input device. In *Proceedings of the Third International Symposium on Wearable Computers*, pages 199–202, 1999.

[41] S. Veigl, A. Kaltenbach, F. Ledermann, G. Reitmayr, and D. Schmalstieg. Two-handed direct interaction with artoolkit. In *Proceedings of the First IEEE International Workshop on the Augmented Reality Toolkit*, 2002.

[42] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *Proceedings of Graphicon 2003*, 2003.

[43] D. Wigdor and R. Balakrishnan. A comparison of consecutive and concurrent input text entry techniques for mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 81–88, 2004.

[44] I. Witten and T. Bell. The zero frequency problem: Estimating the probabilities of novel events in adaptive text compression. In *IEEE Trans. Information Theory*, pages 1085 – 1094, 1991.