Template Based Gesture Recognition

by

Darren Phi Bang Dang

B.S., Electrical Engineering and Applied Science, Caltech (1994)

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996
[June 1996]

Signature of Author .....
Department of Electrical Engineering
and Computer Science
May 13, 1996

Certified by .....
Tomás Lozano-Pérez
Professor, Department of Electrical Engineering
and Computer Science
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Template Based Gesture Recognition

by

Darren Phi Bang Dang

Submitted to the Department of Electrical Engineering
and Computer Science
on May 13, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Throughout the history of computer science, there has always been efforts to make interactions with computers more natural and less intrusive. The invention of the mouse and windows environment has revolutionized the user's capability in interacting with the computer. Touch screens have also brought a new dimension into user interfaces, but a more radical interface is called for in an age where human computer interactions are not limited to computer screens. A human computer interface which uses machine vision to recognize hand gestures is, therefore, necessary to provide a more natural human computer interaction. This thesis will present a solution to recognizing pointing gestures on a planar surface. The system can be used to replace a one button mouse and is a step towards providing a more natural human-computer interface.

Thesis Supervisor: Tomás Lozano-Pérez
Title: Professor, Department of Electrical Engineering
and Computer Science

About the Author

Darren Dang received the degree of Bachelor of Science with honors in Electrical Engineering and Applied Science from the California Institute of Technology (Caltech) in June of 1994. Following graduation he joined the M.I.T. Artificial Intelligence Laboratory to do research in computer vision.

Acknowledgments

First and foremost, I would like to thank my thesis advisor, Tomás Lozano-Pérez, for his endless support and for spending many long hours reading revisions of this thesis. I feel very fortunate to have him as a mentor and friend. His thorough knowlegde of computer vision, AI, and computer science in general has been a tremendous asset. I would also like to thank my academic advisor, Albert Meyer, for his advice and guidance. I especially would like to thank all the members of the HCI group for an enjoyable atmosphere. I also wish to thank all of the individuals who make the Artificial Intelligence Laboratory an outstanding research environment. I owe much to my family for their love and encouragement. My years at Caltech and MIT would not have been possible without them. For Thanh, I would like express deep appreciation for giving me thoughtful comments on my thesis and for many wonderful meals. This work has been supported by DARPA under Rome Laboratory contract F30602-94-C-0204.

To my family

Contents

List of Figures

Chapter 1

Introduction

In the early stages of computer development, scientists used punch cards to communicate with computers. Later, with time shared systems users could type in their instructions. The invention of the mouse and window based systems brought about the point-and-click environment and made computers accessible to a wide range of people. Currently, speech recognition systems are being developed so we can talk to our computers. At the same time, computer vision researchers are also exploring ways to allow humans to interact with computers using hand and body gestures.

Gestures range from a frown to a simple shrug to a wave of the hand. Pointing is one of the most common gestures used in daily interaction. We point to identify objects of interest and to direct attention to a particular direction. Thus, using computer vision to recognize pointing is a step towards providing a more natural interaction with the computer.

1.1 The Problem

General pointing involves using multiple cues and is largely context-dependent. However, sometimes pointing alone is not enough to identify objects. For example, if we want to identify a man among a crowd, pointing in the direction of the man may be ambiguous. In these circumstances, we use other cues such as descriptive words to give the observer additional information. In addition to pointing in the direction of the man, we can also describe him as, 'the man wearing a pink suit.' Since solving the general pointing problem is beyond the scope of a master's thesis, this thesis focuses only on pointing at objects that are on flat surfaces. The problem is as follows: given a sequence of video images taken from a fixed camera mounted on the ceiling looking down at a table, the task is to determine which icon on the table is being pointed at. Figure 1-1 shows a view of the table with a number of icons (square floppy disks) on top. Figure 1-2 shows a snapshot of a pointing event. The problem requires identifying the shape of the hand and accepting only shapes that are in the form of a pointing hand such as the one in figure 1-2. To determine which object is indicated, the system also needs to compute the location of the tip of the index finger in the image.

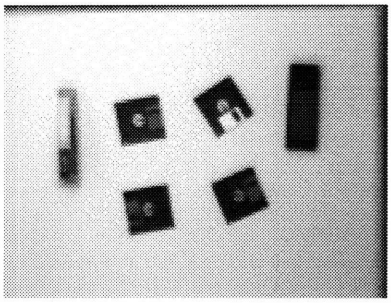

Figure 1-1: A snapshot of the scene

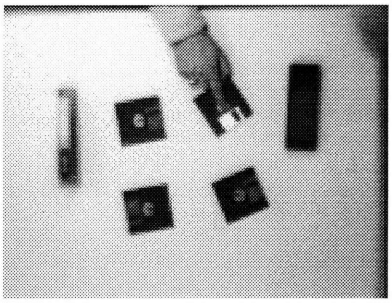

Figure 1-2: Scene where a pointing event is taking place (the hand is in the shape of a pointing hand)

1.2 Challenges

The gesture recognition problem has a number of challenges which make it an interesting problem. First of all, the hand has many degrees of freedom and slight changes in the configuration of the hand can produce a different image in the camera. The hand has approximately twenty-two degrees of freedom. The thumb has three degrees of freedom: one for the outside joint and two for the joint connecting the thumb to the palm. The other four fingers each have four degrees of freedom for a total of nineteen. The wrist has two degrees of freedom: one in the horizontal and another in the lateral direction. The arm has a rotation component that accounts for the twenty second degree of freedom. Without considering the degrees of freedom in the other parts of the body, the hand itself is a complex mechanism. Any change in one of the joints in the hand could change the shape of the hand in the image. Since the optical axis is perpendicular to the table, a rotation in the arm would cause the most drastic change in the shape of the hand in the image plane.

Secondly, part of the hand may be missing due to occlusion or the inability to segment out the whole hand from the image. With missing data, it is difficult to distinguish whether something resembles a hand or not. Finally, not only is it necessary to identify the shape of the hand, it is equally important to recover the corresponding location of a specific model feature point in the sample image.

1.3 Application

This research is part of a project called the Intelligent Room at the M.I.T. Artificial Intelligence Laboratory whose objective is to provide a more natural human computer interaction. The pointing gesture system allows users to specify a location or to direct attention to some particular object on a flat surface in The Room analogous to the way a mouse is used in a window environment. One such example is to use the gesture system along with speech commands to navigate the web.

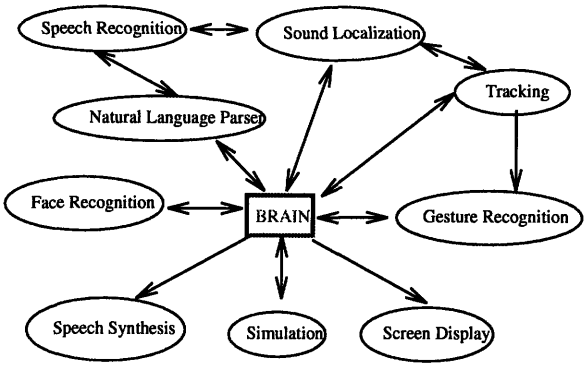

Figure 1-3: Communication between the Brain and the different modules

Figure 1-3 shows the communication pathway between the brain of the office and the various modules in the Intelligent Room. The following are the essential modules that provide the basic interface to the Room: gesture recognition, speech recognition, natural language understanding, face recognition, speech synthesis, and person tracking. The gesture module allows users to interact with the room using their hands rather than conventional input devices such as keyboards and mouse. The speech recognition module [1] enables users to communicate with the Room using spoken speech. The natural language parser enhances the spoken interaction by accepting full English sentences rather than single word interactions. With the face recognition module, the system could identify user or users inside the Room to provide not only security (not implemented) but also to learn to associate certain interactions with particular individuals for more efficient response. Person tracking is also an important module because it allows the Room to monitor people's activities. For example, if two people are standing still next to each other, they are probably having a conversation. If a group of people are sitting around a table, a meeting is probably taking place. If a person is walking up to or standing next to one of the display screens then that person probably is trying to use the screen to display something. Because resources are limited, it is important that the Room can identify what the context of the interaction is so that it can allocate its resources efficiently to provide faster response.

1.4 The Approach

In this thesis, a pointing interaction is simulated by placing a number of icons on top of the table as shown in figure 1-2. Whenever the gesture system detects an object in the shape of a pointing hand, it will keep track of the tip of the index finger. When the tip of the index finger is on top of an icon for more than a few seconds, an event associated with the icon is executed. The events can be anything the Room has

[1] We used the speaker independent speech recognition system developed by the SLS group at the M.I.T. Laboratory for Computer Science.

access to and range from opening a Netscape window to dimming the lights in the Room to executing a program on some computer. The gesture recognition module sends the coordinates of the detected finger tip to the Brain who performs the higher level processing based on the current state of The Room and the inputs from other modules.

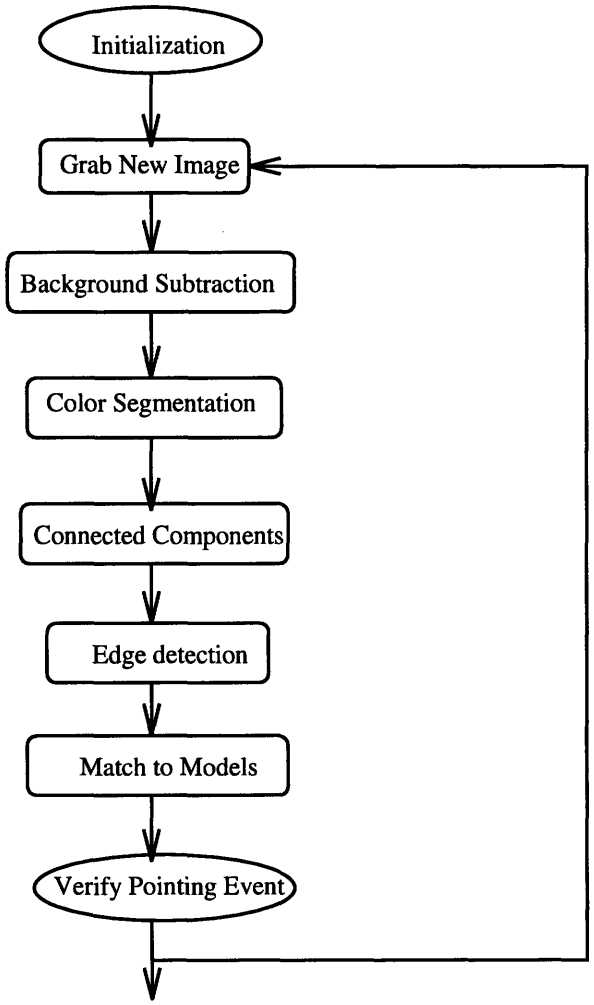

Figure 1-4: Flowchart of the pointing gesture system

Figure 1-4 is a flow chart of the approach used in detecting a pointing event. In the initialization stage, the icons are detected; the background is set; and the library of outlines of hands in the shape of acceptable pointing hands is read in. Then a new image is grabbed from the camera. The background is subtracted from the new image and is updated by allowing objects that have been stationary on the table for a set amount of time to become part of the background. Connected components based on hue and saturation values are run on the background subtracted image to extract different color regions. Boundary points of segmented components with sufficient number of points are then matched against a model data set of 2D hand outlines.

The match process uses an energy minimization technique which involves defining a function that measures the disparity between the sensory data and the model data. How well the sensory data matches the model data is determined by minimizing this disparity using standard gradient based minimization methods. Once a hand in the pointing configuration is found, the system computes the (x,y) location of the tip of the index finger. If the fingertip stays on top of an icon for longer than a few seconds, then the event associated with the icon gets executed.

Chapter 2

Survey of Gesture Systems

This chapter is divided into two sections: the first section reviews existing gesture systems and the second section discusses various matching techniques that could be used in the 'match to model' part of the proposed approach.

2.1 Gesture Systems

Existing gesture recognition systems fall into the following five categories: systems that use a DataGlove, track colored markers, use template matching, do hand encoding, and perform 3D hand modeling.

2.1.1 Gestures made by DataGlove

Many gesture systems use a DataGlove simply because vision processing is not required. The DataGlove is a glove with sensors that can detect the bending of the fingers. In some cases a transmitter is installed on the glove to allow calculation of the 3-D position of the glove. Three receivers, not placed in a straight line, enable the calculation of the exact 3-D position of the DataGlove. [Baudel and Beaudouin-Lafon, 1993] implemented a remote control of computer-aided presentations using a VPL DataGlove. The gesture model incorporates the start position, arm motion, and stop position of a gesture. The command gestures are table of contents, mark page, next page, previous page, next chapter, previous chapter, and highlight area. The system has a recognition rate of 90-98% with trained users. Despite the disadvantage of having cumbersome wires, DataGlove systems are widely used because the hard-wired sensors are reliable and accurate.

2.1.2 Gestures Derived from Tracking Markers

Motion based gesture recognizers are a class of vision based systems that track the transitions of a point or sequence of points to derive a gesture. These gesture systems track the finger-tips [Davis and Shah, 1993], palm [Nowlan and Platt, 1992], open hand [Freeman and Weissman, 1995], colored gloves [Starner and Pentland, 1993], and dynamically changing hand [Darrell and Pentland, 1993]. These motion based

systems can be separated into two categories. One type uses additional help of colored markers to avoid the problem of detecting the hands. The other type tracks the hand from some known state.

[Starner and Pentland, 1993] implemented a system that tracks a yellow and orange glove to recognize American Sign Language (ASL). The user is required to wear a yellow glove on the right hand and an orange glove on the left hand. The (x,y) position and angle of least inertia of the hands are used with a Hidden Markov Model (HMM) to determine a sentence. This approach is based on the HMM technique used by the speech recognition community. Distinct colored gloves makes it easy for computer vision systems to detect not only where the hands are but also whether the hand is the left or right hand. This system, however, does not make any attempt to determine the shape of the hand except for axis.

Along the same line, [Davis and Shah, 1993] proposed a system that tracks markers placed on both sides of the fingertips. All gestures begin in an open hand position with all the fingers in the upright position. Traversing a Finite State Machine (FSM) based on the position of the five finger tip markers during the gesture sequence determines a gesture. The following gestures were recognized: up, down, left, right, rotate, grab, stop. For example, the gesture to move right is defined as an open hand rotating to the right; the fingertip markers in this motion sequence trace out the rotation path which is essentially five vectors pointing to the right. The other six gestures have different and unique fingertip motion vectors. Recognition of a gesture sequence consisting of nine 128x128 pixel frames, each sampled at 4Hz took one tenth of a second on a Sparc-1.

2.1.3 Template Matching Systems

The above systems have avoided the problem of hand detection by requiring the user to wear a colored marker. This requirement is cumbersome and unnatural. Several researchers ([Freeman and Weissman, 1995], [Darrell and Pentland, 1993], [Starner and Pentland, 1993], [Rehg and Kanade, 1993], [Nowlan and Platt, 1992], [Gao and Wang, 1995]) have attempted to provide a glove-free gesture interface. With the glove or markers, it is easy to identify the hands since the markers are chosen to be very distinct from other objects in the scene; without the markers, the vision system has to detect the hands. [Freeman and Weissman, 1995] and [Nowlan and Platt, 1992] identify the hand from a cluttered scene while allowing other parts of the body to move simultaneously. Other researchers assume that the hand is the only object moving in the scene.

Systems that Localize Hands

[Freeman and Weissman, 1995] introduced a system that uses an open and closed hand configuration to control the television. The system detects the hand by performing normalized correlation between a model template of the open and closed hand and patches of the image, over the entire image. An open hand initiates the gesture interaction mode where the hand is tracked and treated as a mouse. A closed

hand terminates the interaction. An interaction sequence is as follows: the user opens his/her hand and a hand icon appears on a separate feedback TV along with some icons (buttons) for changing volume and channel. As the user moves the hand around, the hand icon follows. An event associated with a particular icon is activated if the hand icon is left on top of that icon for a small fraction of time. Closing the interaction loop with feedback is very useful since the user gets a sense of how the system operates and can make adjustments to help the system improve its performance. This system can operate in real time because hand detection is performed only once and the rest of the computation is to track the hand.

This system uses normalized correlation with a fixed template to detect the hand. Thus, a hand in an orientation sufficiently different from the stored model will not be recognized. For controlling the television, recognition of the templates in a different orientation is not necessary. But for pointing, orientation is a major issue since the user could be pointing from many different angles. There are limitations to using the bitmap correlation method. One would have to generate a distinct bitmap for each sufficiently different orientation and the result of comparing a sample bitmap to a model bitmap at one orientation does not give a bias for which of the remaining orientations to choose. The next chapter discusses a method that takes advantage of the results from previous correlations to estimate the orientation of the hand.

Instead of having a fixed template of the open and closed hand, [Nowlan and Platt, 1992] used a neural network to learn the template of the hand. The system operates on a four times subsampled image. A hand locator net attempts to find the hand in the difference image, an image found by subtracting the current image from the previous image, pixel by pixel. Simultaneously, another hand local net operates on the intensity values of the current image and the result from the two nets are used to vote for the best position of the hand. The voting procedure is as follows: the position of the hand from the intensity network and the difference network is spatially ANDed and the best position is used if it is above some threshold. Otherwise, the strongest difference network prediction above the threshold is used. Else, the best intensity network prediction above the threshold is used. Finally, if all the previous steps fail, the prediction from the previous frames is used to estimate the hand location. The estimate from the hand tracker network is then fed into the open-closed recognizer net. Nowlan and Platt reported an accuracy of 91.8% within 10 pixels for test images at a resolution of 320 by 240 pixels acquired at 10Hz. For each of the 18 subjects, 30 frames were collected for training and 19 were used in testing. A neural net is useful in training the computer to recognize various models. The training, however, is tedious and time consuming. In order to add a new model, one has to provide the neural net with many examples whereas in the template based approach one just has to add the new model to the database.

Systems that assume only the hand is moving

Instead of focusing on hand detection, some researchers assume that the hand begins in a known position. [Darrell and Pentland, 1993] start with the hand in some known position and track the changes in its shape during a gesture sequence. Previous

systems track the hand but do not capture the changes in the shape of the hand during a gesture sequence. Tracking the spatio-temporal changes in the hand is called dynamic gesture recognition. Darrell and Pentland studied two gestures: the hello and good-bye gesture. The system starts with a known view of the hand and tracks the disparity between the current state of the hand and the most recently known state using normalized correlation. Once the correlation score falls below a preset threshold, the current view is added to the model set and the most recent known state becomes the current view. A gesture is the set of views observed over time. The detection phase consists of matching stored gestures against most recently observed view models. This system operates in real time but requires special hardware. Although this approach uses the same normalized correlation method as previous systems, orientation is no longer an issue since new models are added as the correlation score falls below a certain threshold. This dynamic approach does not apply to pointing since pointing is a static gesture and the shape of the hand does not change during a pointing event. This system also places an unnatural restriction on the user to start every gesture sequence from a given position.

2.1.4 Hand Encoding Systems

Hand encoding systems explore compact representations to encode snapshots of different hand configurations. A distinct configuration could be used to represent a command. For example, an open hand with the fingers upright could be used as a 'go' command and a closed fist could be a 'stop' command. [Freeman and Weissman, 1995] uses this scheme with the additional tracking of the open hand. Hand encoding deals with gestures in a static way but the output of the encoding could be used by an FSM or HMM to recognize dynamic gestures as in [Davis and Shah, 1993] and [Starner and Pentland, 1993]. The predominant issue in any encoding technique is information retention. Hand encoding methods tend to be global methods which do not retain the local information necessary for the pointing application. There are two types of hand encoding systems: one which uses orientation histograms ([Freeman and Roth, 1994]) and the other which uses difference Freeman Chain codes ([Gao and Wang, 1995]). Orientation histograms do not capture the shape of the object and chain codes do not provide a way to extract the location of feature points such as the tip of the index finger.

Orientation Histograms

[Freeman and Roth, 1994] used local orientation histograms to encode the static hand configurations. This is based on a pattern recognition technique developed by R.K. McConnell. Steerable filters are used to compute orientation maps of the image of the hand. A polar plot of the orientation angle and frequency is then used as the feature vector for identifying different hand configurations. As the name implies, this method is useful for discriminating orientation but not useful in differentiating shape.

Chain Codes

[Gao and Wang, 1995] used difference Freeman chain codes to encode the different static hand poses. Their system traversed the outline of the hand to produce a difference chain code. The difference code is plotted on an x,y axis where the x axis is the order of traversal along the contour and the y axis is the modified Freeman code. The plot is then low-pass filtered to produce a plot that has a positive lump for a concave edge and a negative lump for a corresponding convex edge. For example, an open hand with the five fingers upright and separated produces a difference chain code which, after smoothing, corresponds to five positive lumps interspersed by negative lumps. The smoothed difference chain code is fed into a back-propagation neural network for automatic recognition. This system relies on a clear segmentation of the hand with no discontinuities. Shadows, trailing edges, and noise in the segmentation could severely alter the chain code of the hand.

2.1.5 Hand Modeling

Instead of using multiple templates or models, [Rehg and Kanade, 1993] proposed a system called *DigitEyes* which used a single 3D model of the hand to explain the appearance of the hand in the image. *DigitEyes* models the hand kinematically and geometrically. *DigitEyes* can recover the 27 degree of freedom configuration of the hand at 10Hz. The hand is modeled as a collection of 16 rigid bodies: 3 finger links for each of the five digits and a palm. Each of the four fingers are assumed to be planar with four degrees of freedom, 3 at each link point and an additional one at the connection between the finger and the palm. The fingers are modeled geometrically by cylinders and the fingertip is represented by a hemisphere. Exact measurements of the links are manually entered into the system. The system relies on these measurements and the intensity changes at the link joints to identify the links and to recover the hand configuration. The system uses specialized hardware and stereo to perform 3D recovery. One of the main issues in stereo is finding the correspondence between the set of pixels that represent the hand in one view and the set of pixels that represent the same hand from a different view. Solving for the correspondence is computationally demanding. The next chapter describes a template based approach that is more efficient in solving the simpler 2D pointing problem.

2.1.6 Summary of Gesture Systems

DataGlove and colored marker trackers are not desirable because wearing gloves or markers is cumbersome and unnatural to users. Less intrusive computer vision based systems are more favorable. For pointing, global hand encoding methods such as [Freeman and Roth, 1994], [Gao and Wang, 1995] are not useful because local information is also needed to restore the location of the index finger tip. Pointing is a static gesture and is constrained to a relatively fixed depth. Thus, it makes sense to use a template based approach since we are only interested in a few templates. The main bottleneck in any template based approach is the way in which the match

between the sensory data and the model database is performed. Instead of using the traditional match methods in which a bitmap of the sensory data is correlated with a model database of bitmaps as used in [Freeman and Weissman, 1995], [Nowlan and Platt, 1992], [Darrell and Pentland, 1993], this thesis introduces two extensions. First, since the hand is uniform in color, the outlines of the hand configurations are used instead of the bitmap so there are fewer points to process. Second, the match is not by correlation with a bitmap but by a match technique using energy minimization which can handle rotation. This method is based on the minimization approach in [Grimson *et al.*, 1993].

2.2 Matching Techniques

Many computer vision tasks require some form of matching of a model patch to some part of the image. Matching across two images is used in surface reconstruction to derive topographical maps from satellite images, in assembly lines to spot for defective parts, and in registration for enhanced reality. Pattern matching can be viewed as a method of solving for the transformation from one image patch to another.

There are roughly three classes of match methods to choose from: feature matching by searching the correspondence space, matching by searching the pose space, and matching by using alignment techniques.

2.2.1 Feature Matching by Searching the Correspondence Space

Local features such as edges, lines, distinct curves, or certain image patches are extracted from the sensory data and used to match against model features. This is essentially a tree of all possible pairings between model features and sensory features. The task is to find a leaf such that the path from the root to the leaf maintains a consistent interpretation of the data. An interpretation consists of the set of parameters that represent the transformation between model-data features at each node. For m model features and s sensory image features, there is an interpretation tree that has m^s nodes (not counting cases where the sensory feature do not exist among the model features). Since the search method can be exponential, geometric constraints are sometimes used to prune the interpretation tree [Grimson and Lozano-Pérez, 1987]. Once the leaf of the tree is reached, a set of parameters is obtained to compute the location of any feature point.

2.2.2 Matching by Searching the Pose Space

In the simplest form, *matching by searching the pose space* involves correlating the model object with the sensory data over all possible poses to find the best correlation. Methods such as generalized Hough transforms and geometric hashing can be used to avoid searching through the entire solution space.

Generalized Hough Transforms, also known as parameter hashing, performs a search in the pose space to find the best model. The Hough space, H, is a hash table for the pose parameters. For each pair of model and sensory image feature, the range of possible transformations is computed and entered into the hash bucket in H. Since model-sensory image feature pairings which hash into the same bucket in H define similar transformations, matching simply entails searching the Hough Space, H, for the bucket with the greatest number of entries.

Geometric Hashing [Lamdan and Wolfson, 1988] is an alternative approach to generalized Hough transforms. The idea is to compute all possible bases using the model points and to store these new bases in the hash table at all bucket entries which correspond to the position of the remaining model points relative to the new basis. In the 2D case, geometric hashing consists of building a hash table of bases as follows ([Grimson, 1990]):

- choose 3 non-linear model points m_1, m_2, m_3 as a basis formed by an origin o and a pair of axes u, v

$$\begin{aligned} o &= m_1 \\ u &= m_2 - m_1 \\ v &= m_3 - m_1 \end{aligned}$$

- rewrite the coordinates of each additional point m_i in terms of the basis u, v. That is, find α, β such that:

$$m_i - o = \alpha u + \beta v$$

- hash index (α, β) and store in the hashed bucket the basis triple (0, u, v)
- repeat the process for all possible model triples.

The lookup phase is performed in a similar manner. Three sample points in the sensory image are chosen to form a basis. The remaining points are transformed into this new basis and the transformed position is used as the key to lookup the basis (0, u, v) in the hash table. A tally of these bases are kept and the (0,u,v) basis with the greatest count (above a certain threshold) is chosen to be the correct transformation, after all possible sample bases have been explored.

2.2.3 Alignment Techniques (Hypothesize and Test)

Hypothesize-and-test methods [Ullman, 1987], [Huttenlocher and Ullman, 1987] consider just a sufficient amount of data to recover the pose for verifying the correctness of the hypothesis on the rest of the data using this estimated pose. For example, in 2D a pairing of two points in the model data with two other points in the sensory data is sufficient to compute the transformation that maps the sensory data into the model data. The transformation is applied to the rest of the points to verify the interpretation. All possible pairings are considered to find the best interpretation. In 3D, three nonlinear point pairings are sufficient to recover the pose.

2.2.4 Match by Using Minimization

Previous systems exhaustively consider either all model-sensory data feature pairings or model-sensory bases pairings. An alternate approach is to define a distance function that relates the model and sensory data set and then minimize this function to obtain their relationship. This method has been used in the medical field ([Grimson *et al.*, 1993]) to register MRI or CT scan data to laser scan data of the patient's head. The task is to align the laser scan data of the head surface to the 3D MRI points. Once the set of transformation parameters from one data set to another is known, additional video sources can be inserted into these data sets for enhanced visualization to aid in neuro-surgery. This method involves providing the initial alignment of the 3D MRI or CT scan data points with the laser data using a graphical user interface. The initial alignment gives a estimate of the view direction which is then used to project 3D MRI/CT points onto a plane orthogonal to the estimated view direction (the 3D points closest to the plane are kept). The user then selects the laser data points that correspond to the head by drawing a box around it; this is done to avoid capturing outliers. Next, all possible hypotheses between three laser data points and three corresponding 3D points are formed and validated using geometric constraints. The set of hypotheses is further pruned by using the Alignment Method ([Huttenlocher and Ullman, 1990]) where the transformation defined by the three pairs of corresponding 3D/laser data points are used to verify the validity of the rest of the data points for the given transformation. The resulting hypotheses are verified by using Powell's method to minimize the sum of Gaussian weighted distances to obtain an approximate fit. The sum of the closest distance squared between the laser points and the MRI/CT data is minimized to get a finer alignment. Random restarts are used to kick the system out of local minima. [Grimson *et al.*, 1993] Since then they have used local gradient based methods to perform the minimization. In applying the minimization technique to hand gesture recognition, this thesis introduces a number of optimizations to enable real time performance. An automatic initial alignment based on the centroid and axis of least inertia is introduced. In addition, the closest distance is precalculated and stored so that each closest distance computation is reduced to a table lookup. The next chapter will describe the match method in greater detail.

Chapter 3

Pointing Gesture System

Figure 1-4 is a flowchart of the approach to solving the pointing problem. There are basically two things that need to be done. The gesture system needs to determine the shape of the hand and identify the location of the tip of the index finger. In this problem, pointing is constrained to the table top at a fixed depth from the camera and the hand will be mostly planar. Since pointing is a static gesture and there are only a few ways to point, a template based approach is appropriate. Traditional correlation based methods of convolving or finding the sum of squared differences between a sample bitmap and a model bitmap can not handle occlusions well and do not take into account orientation which is crucial in pointing. It is often the case that a separate template is generated for each desired orientation. This work proposes a different approach which uses only the silhouettes of the hand and the match is based on energy minimization. Energy minimization involves defining a match metric which measures the disparity between the sample and the model and uses standard gradient based minimization techniques to minimize the disparity. The match metric is the root mean squared (RMS) of the thresholded distance of each point in the model data set to the nearest point in the sample data set. For each model, this method can handle a range of rotation in the plane parallel to the table and can also handle occlusions.

Another problem to gesture recognition is segmentation. How does one extract the object of interest from the scene to compare it with the model templates? One option is to use motion and color to segment out the hand. This method is not feasible because the whole hand may not be moving and the color of the skin is unknown. Since the color of the hand is uniform, background subtraction can be used to extract pixels that are different from the background. These pixels are separated into different groups based on color. Then, the outlines of these clusters are matched to a model database of hand outlines. To isolate the hand from the background, this method assumes that the color of the hand is distinct from the background. In summary, there are four stages in the pointing system: an initialization stage, a segmentation stage, a matching stage, and then a verification stage.

3.1 Initialization Stage

The initialization stage entails initializing the background, reading in the model database of hand configurations, detecting the icons, and assigning some action or set of actions to each icon. The action associated with each icon can be anything which the Brain can access, ranging from executing Netscape on a particular machine, to turning the lights off, to activating tracking. The user can change the actions associated with the icon at any time. The entire set of actions can be changed on the fly to a different set of actions based on the context of the room. For convenience, any circular or square object the size of a post-it note is used as an icon. The icons are detected from the background image by first performing color segmentation based on HSV values. The HSV color space used because it is less sensitive to lighting variations in the image. Connected components are run on the image to label the different groups in the image. For each group, if the ratio of the central (2,0) moment and the central (0,2) moment is close to 1 and the number of points in the particular group is between 1% and 3% of the total number of pixels in the image, that group of points is an icon. The resolution of images processed by the gesture system is 160x120 pixels (a quarter of the full resolution).

$$\mu_{jk} = \sum (x - \bar{x})^j (y - \bar{y})^k \tag{3.1}$$

Equation 3.1 defines the central (j,k)th moment. For a circle, the ratio of the second central moment is 1; the ratio for a square is approximately 1. Square and circular icons in arbitrary rotation on the table can be detected using this method. Using straight forward correlation, one would have to rotate the template and perform the match for each desired rotation. This method would not be very efficient.

Figure 3-3 shows the boxes extracted from the edges (figure 3-2) of the background image (figure 3-1). The edges of the diskette in the upper right corner is not square because the metallic part blends in with the background. The object is not detected as an icon.

3.2 Segmentation Stage

This section is concerned with segmentation. The purpose of segmentation is to select the outline of objects in the scene to be used for comparison with the model data set. The quickest way is to take the difference of the previous image from the current image which will give an indication of the areas that have changed. The outline of the hand will show up in the difference image if the whole hand moves and if it moves slowly. However, this is often not the case. Sometimes only part of the hand is moving. For example, people sometimes point by wiggling their index finger without moving the rest of their hand. In such cases, only the index finger appears in the difference image. At other times, the hand moves too quickly from one frame to the next leaving little overlap between two consecutive frames. Figures 3-4 and 3-5 is an example of two consecutive frames at time t_i and t_{i+1}. Figure 3-6 shows the difference image between these two frames. Since the hand is moving too fast, two

Figure 3-1: The background image

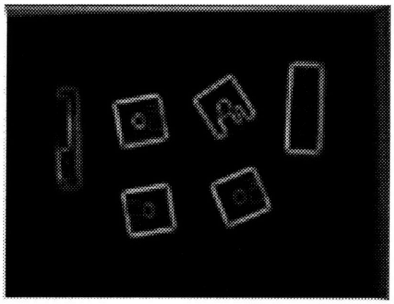

Figure 3-2: The edges of the background image

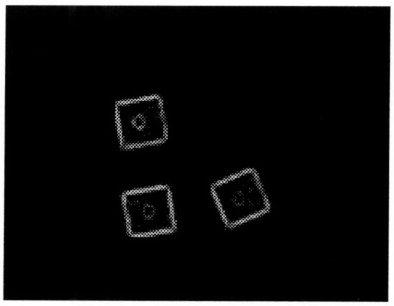

Figure 3-3: The boxes detected from the edge image

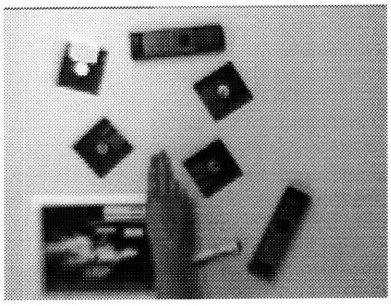

Figure 3-4: segmentation example: frame t_i

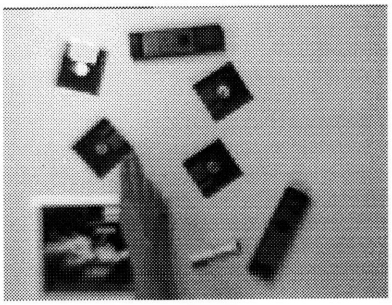

Figure 3-5: segmentation example: frame t_{i+1}

Figure 3-6: difference image of frame $t_{i+1} - t_i$. Notice that the frame rate is too slow to capture the motion so two hands show up in the difference image between the two consecutive frames.

hands show up in the difference image. This situation occurs when the frame rate is too slow to capture the changes in an object that moves too quickly.

An alternative approach is to use an adaptive background and subtract this background from the current image to extract the foreign objects. The restriction that the background does not change during pointing is imposed otherwise everything will show up in the difference image between the current frame and the adaptive background. Regardless of how rapidly the hand moves, only one hand will show up when the current frame is subtracted from the adaptive background. The pixels of foreign objects in the scene are tracked. If the object remains in the scene for more than n frames (typically n=100), it becomes part of the adaptive background.

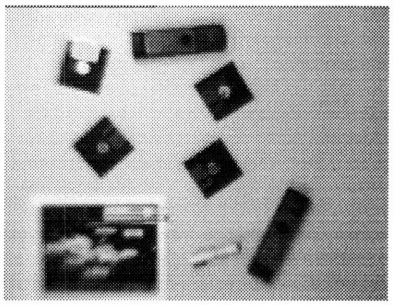

Figure 3-7: The background image

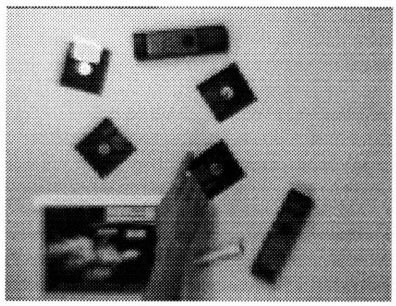

Figure 3-8: snapshot of a live video image

Figure 3-9: The background subtracted image. Note the shadow cast by the hand.

Figure 3-10: The segmented hand.

Since the color of the hand is uniform, one can exploit this property to segment out the hand. Color segmentation is performed on the background subtracted image to obtain separate components by running connected components based on HSV values. Color segmentation is important because the hand casts a shadow on the table which

will show up in the difference image. Color segmentation enables the separation of the shadow of the hand on the table from the actual hand. Components with few points are discarded and the remaining components are fed into the match routine.

3.3 Match to Model Stage

The match routine selects a model that best resembles the data. This routine also returns a set of parameters that enables the calculation of the position of feature points such as the tip of the index finger. It is important that the system can handle rotation of the object in the plane. Any model based match method can have the risk of having too many models which will stress the computational capability of the system. Thus, it is essential to limit the size of the pointing database without sacrificing functionality. Traditional correlation methods can not handle rotated objects that are sufficiently different from the model. Figure 3-12 is an example object and figure 3-13 shows the object rotated 15 degrees clockwise. Any straight forward correlation method which convolves a model object with a similar object that has been rotated will not give the desired result. To cope with rotation, sufficiently different instantiations of the model at the desired orientations must be stored. Adding more models will make the system slower. Even if methods that do not account for rotation can select a proper model, this only gives an index into the model database and does not provide a means of determining the position of a particular feature point.

A score function which is based on the root mean squared (RMS) of the thresholded distance to the closest sample point is used to calculate an average measure of the disparity between the two data sets. The closest distance is thresholded to avoid capturing outliers. If the data set approaches its best fit with the model, the function score becomes smaller because the points in the model is closer to the sample points. This sets up a nice framework for energy minimization.

Correspondence between the sample data set and the model data set is no longer an issue when using this scoring function since the closest point is selected. Finding a correspondence between the sample and model is the main bottleneck in many match systems. If the sample and model were outlines of the identical object, the score function would be zero at the global best fit because each point in the sample would correspond to a point in the model in exactly the same location. All minimization problems, however, are plagued with the problem of local minima. Thus, minimization only takes the function to a local minimum but does not guarantee that point to be the global best fit. This local minimum problem will be addressed later in the chapter.

In general, the match process involves defining a score function which quantifies the disparity between two data sets. A standard gradient based minimization technique is then used to find the minimum disparity. Since the extra information in the gradient will accelerate the minimization process, it is necessary to derive the gradient of the score function. In addition, implementation issues used to increase efficiency will also be discussed later in the chapter.

3.3.1 The Scoring Function

The energy minimization scheme requires a score function that is 1) based on some adjustable parameters and 2) quantifies the disparity between the model data and the sample data given those parameters. For the match routine, the parameters are the (x,y,θ) vector used to transform the sample data. The score function quantifies the similarity between the transformed model data points and the sample data points. A reasonable metric to use for the similarity measure is the Euclidean distance (eq. 3.4) between points in the model data to the closest point in the sample. The model data is first rotated by θ and then translated by (x,y). Then it is overlaid on top of the sample data for comparison. The comparison is performed as follows: for every transformed point in the model data set, sum the square of the thresholded distance between the transformed data point and the sample point closest to it; the square root of the average is returned as the output of the scoring function. To avoid capturing outliers, the search is limited to a small search area and the closest distance is thresholded. This score is referred as the root mean squared (RMS) on the closest distance.

The RMS on the closest point provides an approximation of the distance between the points in the model data set to the sample data but does not capture the local properties of each point. For each model data point, the scoring function only reflects the average proximity of this point to its nearest neighbor in the sample data set but does not reflect its relation to other neighboring points in the sample data set. This may result in a score function that is not smooth. A small change in (x,y,θ) may cause a discontinuous change in the scoring function.

$$GaussianWeightedDist(\vec{m'}_i, \sigma) = \sum_{i=1}^{n} e^{-\frac{EuclideanDist^2(\vec{m'}_i, \vec{s_j})}{2\sigma^2}} \tag{3.2}$$

An alternative to the distance-to-closest point metric is to use a Gaussian weighted distance (equation 3.2). In equation 3.2, $\vec{m'}_i$ is the transformed model point according to (x,y,θ) and σ is the variance of the Gaussian distribution. $\vec{s_j}$ is the sample point that is closest to a particular $\vec{m'}_i$. The computation of $\vec{m'}_i$ will be discussed later. For each transformed model point $\vec{m'}_i$, all points in the sample data set within the search distance to $\vec{m'}_i$ will be considered and weighted according to a Gaussian distribution. A small change in the pose parameters creates smoother changes in the score function. Although the Gaussian weighted distance gives a smoother scoring function, the Euclidean distance is used because it is less computationally intensive.

The score function is defined more formally below. Given a particular pose vector $\mathcal{P}$ consisting of (x,y,θ), the RMS score function on the closest point is defined as follows:

$$C(s, m, x, y, \theta) = \sqrt{\frac{\sum_{i=1}^{n} EuclideanDist^2(\vec{m'}_i, \vec{s}_j)}{n}} \tag{3.3}$$

$$EuclideanDist(\vec{a}, \vec{b}) = \begin{cases} \| (\vec{a} - \vec{b}) \| & \text{if } \| (\vec{a} - \vec{b}) \| < search\ distance \\ search\ distance & \text{otherwise} \end{cases} \tag{3.4}$$

The score function $\mathcal{C}$ takes in a sample s, a model m, an (x, y) translation, and a θ rotation. n is the number of points in the model data. $\vec{s}_j$ is the sample point closest to the transformed model data point $\vec{m'}_i$. $\vec{m'}_i$ is the vector that represents the model data point i at pose $\mathcal{P}$. The pose vector $\mathcal{P}$ consists of three parameters: x, y translation, and a θ rotation. $\vec{m'}_i$ is found by rotating $\vec{m}_i$ by an angle θ about the sample data's centroid and translating by x, y. In other words, the pose vector $\mathcal{P}$ defines the following transformation matrix $\mathcal{T}$:

$$\mathcal{T} = \begin{bmatrix} cos\theta & -sin\theta & x + s_{cx} \\ sin\theta & cos\theta & y + s_{cy} \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$\vec{m'}_i = \mathcal{T}(\vec{m}_i - \vec{m}_c)^t \tag{3.6}$$

$\vec{m}_c$ is the centroid of the model data set. $\vec{s}_{cx}$ and $\vec{s}_{cy}$ are the x, y value of the centroid. In Equation 3.6, $(\vec{m}_i - \vec{m}_c)^t$ is a column vector that is the difference of vector $\vec{m}_i$ and vector $\vec{m}_c$ transposed. Take for example, a model data point $\vec{m}_i$ at (2,1,1) in homogeneous coordinates and a model centroid $\vec{m}_c$ at (1,0,1). The transformed model point, $\vec{m'}_i$, relative to the pose vector $\mathcal{P}$ of (2,3,90deg) is (2,4,1).

$$\begin{aligned} \vec{m}_i - \vec{m}_c &= [2,1,1] - [1,0,1] = [1,1,1] \\ (\vec{m}_i - \vec{m}_c)^t &= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ \mathcal{T} &= \begin{bmatrix} 0 & -1 & 2+1 \\ 1 & 0 & 3+0 \\ 0 & 0 & 1 \end{bmatrix} \\ \vec{m'}_i &= \mathcal{T}(\vec{m}_i - \vec{m}_c)^t \\ &= \begin{bmatrix} 0 & -1 & 3 \\ 1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix} \end{aligned}$$

Figure 3-11 illustrates the transformation example shown above. The data point $(\vec{m}_i)$ is rotated 90 deg about the data centroid $(\vec{m}_c)$, and translated 2 units right and 3 units up to yield the transformed point $(\vec{m'}_i)$. The transformed point $\vec{m'}_i$ is used to select the point in the sample that has the closest Euclidean distance to it.

Figure 3-12 shows an example of a typical object; it is an outline of the hand in the configuration that is similar to a horn. Figure 3-13 is the horn rotated by 15 degrees clockwise about its centroid. In figure 3-14, the rotated horn is translated by 2 pixels right and 3 pixels up. Figure 3-15 shows the rotated and translated horn overlaid on top of the original image of the horn. The score function computed on the

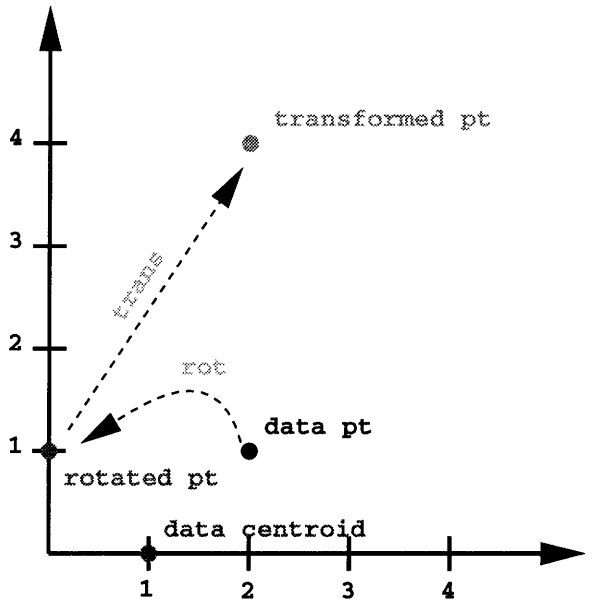

Figure 3-11: Transformation example (pose vector $\mathcal{P} = [2, 3, 90\,\text{deg}]$). The data point ($\vec{m}_i$) is rotated by 90 deg about the data centroid ($\vec{m}_c$) and translated 2 units right and 3 units up to yield the transformed point ($\vec{m'}_i$). The distance squared between $\vec{m'}_i$ and the closest sample point is used in the score function.

original horn and the transformed horn at pose (2,3,15 deg) is 4.22823, a measure of how *close* the transformed horn is to the actual horn. At pose (2,3,10 deg) the score is 4.533140.

Figure 3-16, 3-17, 3-18 show ideal one dimensional plots of the score function between the transformed horn and the original horn. The x axis is the perturbation in each dimension, and the y axis is the value found by computing the RMS score between the horn configuration and the same horn configuration displaced by the x-value along the particular dimension.

Matching displaced data with itself gives insight to how the function behaves in the ideal case. The RMS score is zero when there are no displacements in the x,y, and θ direction. The function displays a quadratic behavior as it moves away from the pose [0,0,0]. The function plot of the theta rotation (fig. 3-18) portrays oscillatory behavior because as the model rotates about the centroid, more points are displaced from alignment. As the model further rotates, the points fall back into alignment. The match routine assumes that the system will converge to a state that is close to a one to one correspondence when the minimization is completed. Since there are several basins of attraction, the system must start with an initial guess sufficiently close to the solution otherwise the function may be trapped in a local minimum. Initial alignment will be discussed in a later section.

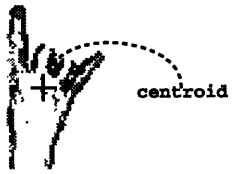

Figure 3-12: an example hand configuration. The horn and its centroid is shown.

Figure 3-13: the horn rotated 15 deg clockwise about its centroid

Figure 3-14: the horn rotated 15 deg clockwise about its centroid and translated 2 pixels right and 3 pixels up

Figure 3-15: the transformed horn overlaid on top of the original horn (RMS function score=4.22823)

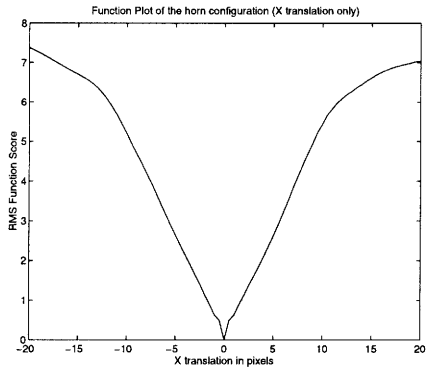

Figure 3-16: plot of the score function between the horn and the horn translated along the x axis

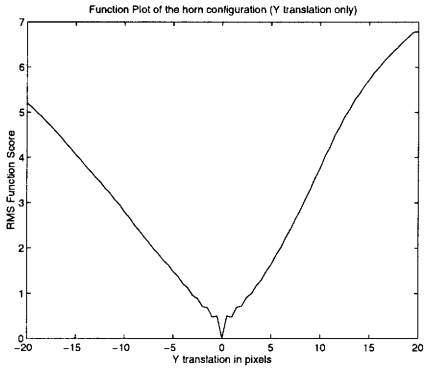

Figure 3-17: plot of the score function between the horn and the horn translated along the y axis

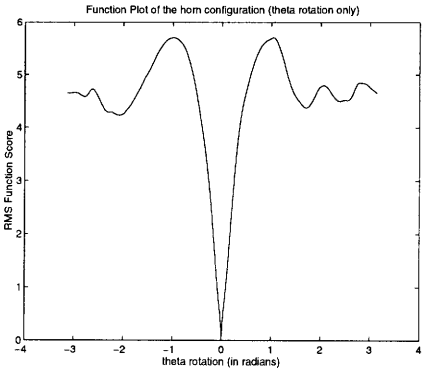

Figure 3-18: plot of the score function between the horn and the horn rotated about its centroid

3.3.2 Minimization

For each pose $\mathcal{P}$ the score function $\mathcal{C}$ represents the proximity of the model data set to the sample data. In the ideal case, the model data set is identical to the sample data set; $\mathcal{C}$ is equal to zero. This suggests that given some initial pose, the match routine must find a pose such that the score function is close to zero. In mathematical terms, the match problem reduces to minimizing $\mathcal{C}$.

Minimization is a well-studied field in numerical analysis. There are many standard minimization methods available. When no information about the gradient is available, Powell's method ([Press *et al.*, 1992] Chapter 10, Section 5) is a possible approach to performing the minimization. Powell's algorithm performs N independent line minimizations along N dimensions and replaces the direction with the greatest changes in the previous iteration with the average direction moved[1] at the end of each iteration.

Figures 3-16, 3-17, and 3-18 show an example of what the function looks like in each of the pose dimension. An interesting question, therefore, is how well does the minimization method recover the pose given these displacements. Figures 3-19, 3-20, and 3-21 show the function recovery of the horn using Powell's method after the horn has been translated along the x, y axis and rotated about its centroid. The x-axis indicates the displacement along the particular dimension and the y-axis is the final RMS score at the end of the minimization. Note that the search distance is only 10 pixels. In the translation case, the function recovery is excellent. In the rotation case, the system performs very well for rotations that are within $\pm$ 89 deg which is exactly to be expected from 3-18 since the origin is the basin of attraction for rotations between $\pm$ 1 radian. It is acceptable for the routine to fail when objects are rotated more than 90 deg. To recognize the entire range of rotations, a mirror image of the sample about the x-axis can be generated for the match in addition to performing the match on the original sample. A mirror image about the y-axis can also be used to detect the opposite hand (ie. the right hand if the model was the left hand). Note that the function recovery plots above is for perfect data; the match routine will not perform as well for sample data that is not identical to the model. Function recovery for non-perfect data will be discussed in the results chapter (chapter 4).

Powell's method, however, does not take advantage of the wealth of information available in the gradient. The gradient of the score function in equation 3.3 is as follows:

$$\begin{aligned} \mathcal{C} &= \sqrt{\frac{\sum_{i=1}^{n} dist^2(\vec{m'}_i, \vec{s}_j)}{n}} \\ \frac{\delta \mathcal{C}}{\delta \mathcal{P}} &= \frac{\delta}{\delta \mathcal{P}} \sqrt{\frac{\sum_{i=1}^{n} dist^2(\vec{m'}_i, \vec{s}_j)}{n}} \end{aligned}$$

[1] the direction from the old point to the new point after the N line minimizations along the N dimensions.

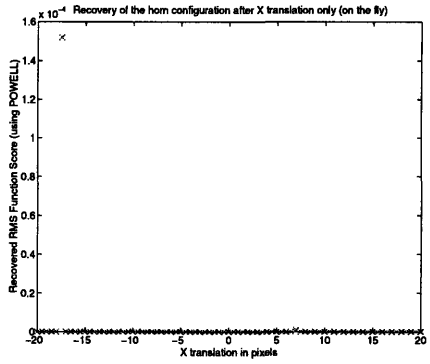

Figure 3-19: Plot of RMS recovery after x translation using Powell's method

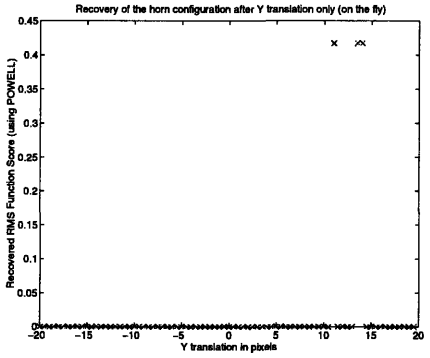

Figure 3-20: Plot of RMS recovery after y translation using Powell's method

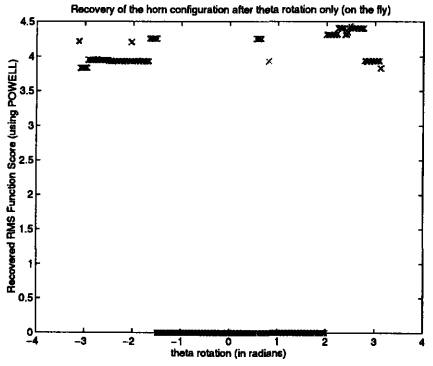

Figure 3-21: Plot of RMS recovery after theta rotation using Powell's method

$$
\begin{aligned}
&= \frac{1}{2n\mathcal{C}} \frac{\delta}{\delta \mathcal{P}} \left[\sum_{i=1}^{n} dist^2(\vec{m'}_i, \vec{s}_j) \right] \\
&= \frac{1}{2n\mathcal{C}} \sum_{i=1}^{n} \frac{\delta}{\delta \mathcal{P}} \left[(\vec{m'}_i - \vec{s}_j)(\vec{m'}_i - \vec{s}_j) \right] \\
&= \frac{1}{2nC} \sum_{i=1}^{n} 2(\vec{m'}_i - \vec{s}_j) \frac{\delta}{\delta \mathcal{P}} (\vec{m'}_i) \\
&= \frac{1}{n\mathcal{C}} \sum_{i=1}^{n} (\vec{m'}_i - \vec{s}_j) \frac{\delta}{\delta \mathcal{P}} (\vec{m'}_i)
\end{aligned}
$$

To solve for $\frac{\delta}{\delta \mathcal{P}}(\vec{m'}_i)$, take the derivative of equation 3.6 with respect to the parameters of the pose vector $\mathcal{P}$. $\frac{\delta \vec{m'}_i}{\delta x} = 1$ and $\frac{\delta \vec{m'}_i}{\delta y} = 1$. $\frac{\delta \vec{m'}_i}{\delta \theta}$ is as follows:

$$
\begin{aligned}
\vec{m'}_i &= \mathcal{T}(\vec{m}_i - \vec{m}_c) \\
\frac{\delta}{\delta \theta} \vec{m'}_i &= \frac{\delta}{\delta \theta} \mathcal{T} (\vec{m}_i - \vec{m}_c) + \mathcal{T} \frac{\delta}{\delta \theta} (\vec{m}_i - \vec{m}_c) \\
&= \frac{\delta}{\delta \theta} \mathcal{T} (\vec{m}_i - \vec{m}_c) \\
&= \begin{bmatrix} -sin\theta & -cos\theta & 0 \\ cos\theta & -sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} (\vec{m}_i - \vec{m}_c)
\end{aligned}
$$

Since the gradient of the score function can be computed, quasi-Newton or conjugate gradient methods can be used to take advantage of this extra information to get an $\mathcal{O}(n)$ increase in efficiency. Powell's method performs N separate line minimizations while the gradient methods perform only one line minimization along the gradient at each step. The quasi-Newton methods, also known as variable metric methods, come in two variations: *Davidon-Fletcher-Powell* (DFP) and *Broyden-Fletcher-Goldfarb-Shanno* (BFGS). The variable metric method computes a good approximation of the inverse Hessian matrix. The conjugate gradient method takes steps that are conjugate to the gradient and also conjugate to the previous steps to avoid spoiling the work done in the previous minimization steps. Although, the conjugate gradient method gives relatively the same level of performance as that of variable metric methods, it uses less storage because it does not approximate the Hessian. See [Press *et al.*, 1992] chapter 10, section 6 for a more in depth description of the conjugate gradient method and section 7 for a discussion of variable metric methods.

Figures 3-22, 3-23, and 3-24 show the function recovery of the horn using variable metric methods after the horn has been transformed independently along the pose axis. The x-axis indicates the perturbation along a particular dimension and the y-axis is the recovered RMS score. With a search distance of 10 pixels, the system is capable of recovering the pose for perturbations that are within a $\pm$ 10 range. The recovery after displacements in the y dimension is better than the recovery in the x dimension because there are more points along the y dimension. If the search

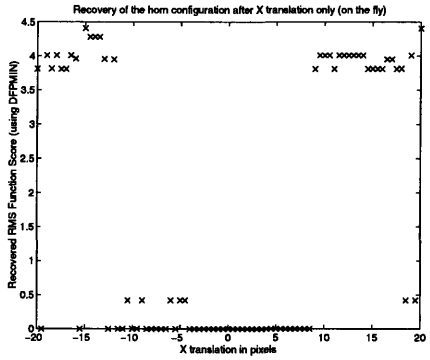

Figure 3-22: Plot of RMS recovery after x translation using Dfpmin method. Search distance=10 pixels.

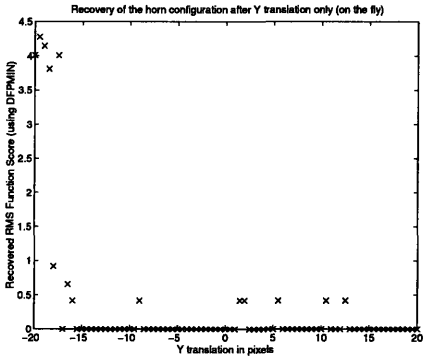

Figure 3-23: Plot of RMS recovery after y translation using Dfpmin method. Search distance=10 pixels.

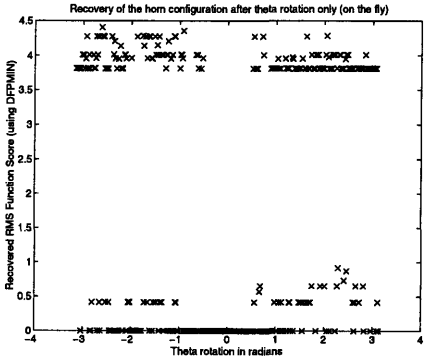

Figure 3-24: Plot of RMS recovery after theta rotation using Dfpmin method. Search distance=10 pixels.

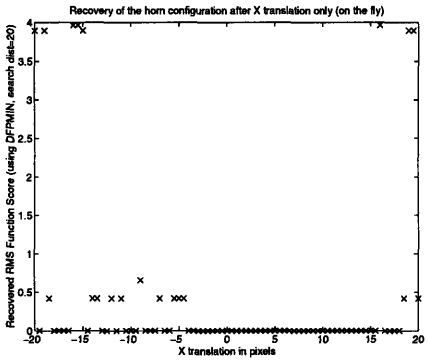

Figure 3-25: Plot of RMS recovery after x translation using Dfpmin method. Search distance=20 pixels.

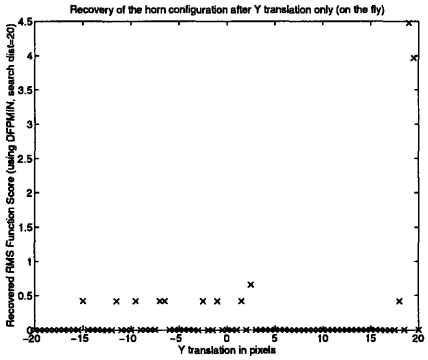

Figure 3-26: Plot of RMS recovery after y translation using Dfpmin method. Search distance=20 pixels.

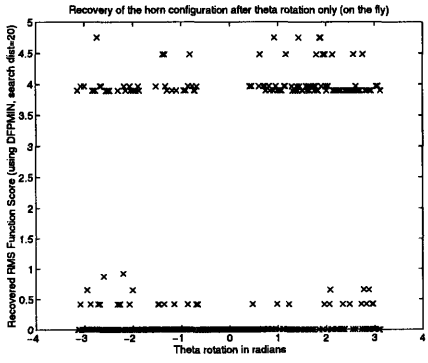

Figure 3-27: Plot of RMS recovery after theta rotation using Dfpmin method. Search distance=20 pixels.

area increases, the function recovery will improve proportionally with the search area. Figures 3-25, 3-26, and 3-27 show the function recovery when the search area has been increased to 20 pixels. The few points that have an RMS score of approximately 0.5 in figures 3-22 through 3-27 are insignificant because they are most likely caused by the discontinuity in the gradient which is an artifact of using the closest Euclidean distance in the scoring function. As discussed earlier, using the Euclidean distance causes the gradient of the function to be discontinuous.

Figures 3-28, 3-29, and 3-30 shows the relative number of function calls using Powell's method versus the variable metric method (DFP). The graph shows an order N decrease in function calls when using the variable metric approach. This is as expected since only one line minimization along the gradient is performed instead of N separate line minimizations along each dimension.

3.3.3 Initial Alignment

The initial estimate plays a crucial role in the match process. To prevent the function from converging to a local minimum, the match routine must begin with a pose that is close to the solution. A simple heuristic is used to automatically compute the initial pose. The (x,y) translation components of the initial pose is determined by aligning the centroids of the sample and model data. The rotation component is found by aligning the axis of least inertia. This simple heuristic works well when there is a good segmentation but if part of the hand was occluded, the centroid and axis of least inertia will not provide very accurate estimates. Although not implemented, a more elaborate method of fitting lines to collinear segments in the image with a Hough transform could be used. These lines in the sample can be aligned to lines in the model to obtain an initial pose. In 2D, only two non-parallel lines are needed.

3.3.4 Optimization

The main limitation of the system is the function evaluations since the score function has to be evaluated many times in order to compute the minimum. More specifically, the bottleneck is in computing the distance from the transformed sample point and the closest point in the sample. Hence, a hash table is computed for each sample to increase efficiency in evaluating the score function. Each hash bucket contains a list of points that are within a search distance to the the (x,y) value corresponding to the hash bucket. Computing the closest sample point to a particular transformed model point reduces to processing the points in the hash bucket corresponding to the transformed model point. The models are also pre-hashed for use in the verification stage.

A point (x,y) is hashed into bucket (i,j) of the hash table as follows:

$$i = (x - x_{min} + search\ distance)\frac{hashsize}{x_{max} - x_{min} + 2 * searchdistance} \quad (3.7)$$

$$j = (y - y_{min} + searchdistance)\frac{hashsize}{y_{max} - y_{min} + 2 * searchdistance} \quad (3.8)$$

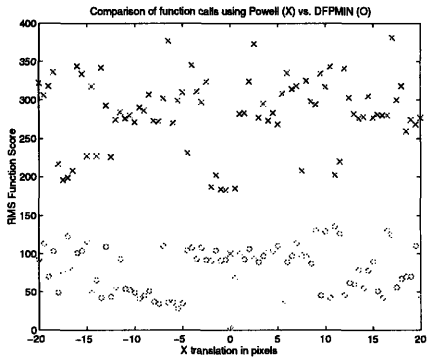

Figure 3-28: Comparison of the number of function calls using Powell's method vs. DFPMIN for X translation only

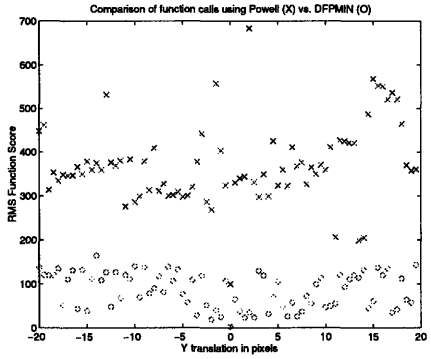

Figure 3-29: Comparison of the number of function calls using Powell's method vs. DFPMIN for Y translation only

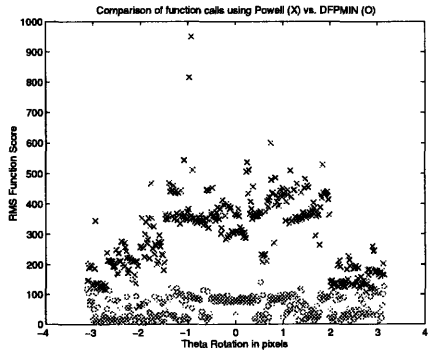

Figure 3-30: Comparison of the number of function calls using Powell's method vs. DFPMIN for theta rotation only

where x_{min} and x_{max} are the minimum and the maximum of the object. The hash table is of size *hash size* x *hash size.* The range of the object data set, $(x_{max} - x_{min})$, is extended by $2 * searchdistance$ to leave some room around the outside edge of the hash table to account for the object points on the outside edge. Solving for x and y in equations 3.7 and 3.8, one obtains two equations (3.9, 3.10) that describe how to convert from hash index (i, j) to object point (x, y).

$$x = \frac{i * (x_{max} - x_{min} + 2 * searchdistance)}{hashsize} + x_{min} - searchdistance \quad (3.9)$$

$$y = \frac{j * (y_{max} - y_{min} + 2 * searchdistance)}{hashsize} + y_{min} - searchdistance \quad (3.10)$$

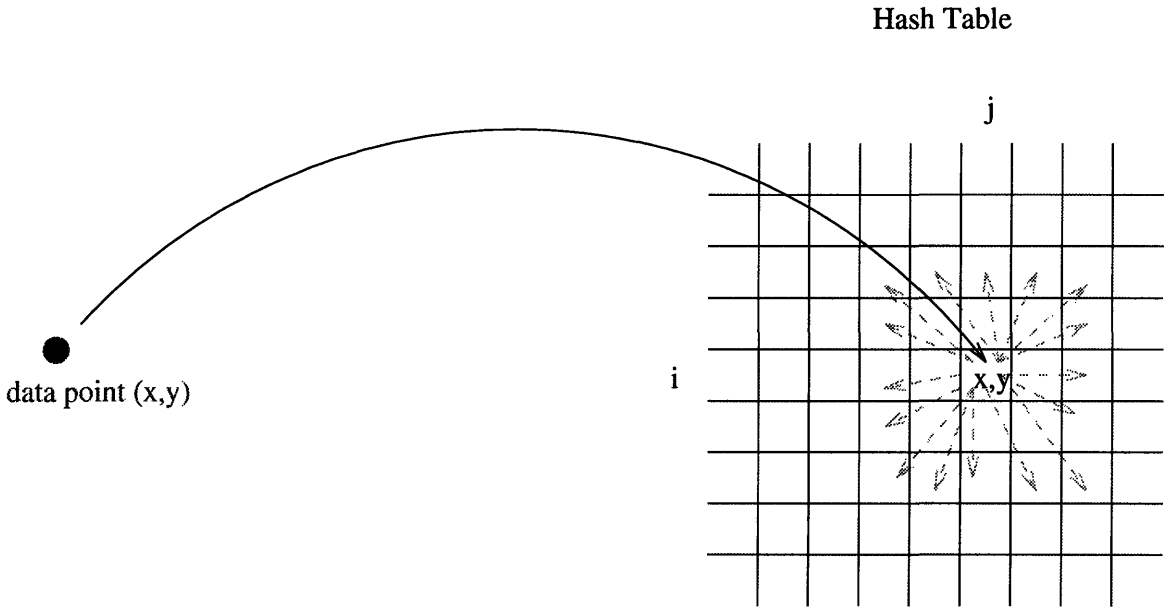

Figure 3-31: A point (x_i, y_i) is hashed into bucket (i, j). (x_i, y_i) is inserted into neighboring buckets that are within the *search distance.* The distance is found by converting (i,j) into (x_b, y_b) and neighboring bucket's (k,l) index into (x,y) value and then computing the Euclidean distance between (x_b, y_b) and (x,y).

Every sample data point is hashed into hash bucket (i,j) using equations 3.7 and 3.8. Equations 3.9, 3.10 are used to find the (x_h, y_h) value corresponding to bucket (i,j). The data point is added to neighboring buckets if and only if the distance between (x_h, y_h) and the (x,y) values of buckets that are close to bucket (i,j) is less than the search distance (see figure 3-31).

Since the model objects are on the average 47 pixels wide by 47 pixels high, the size of the hash table is set to 47x47 to capture a one-to-one mapping from the image pixel to the hash bucket entry. The one-to-one mapping is significant when using pre-computed closest distance.

Computation of the closest sample point to the transformed model data point involves hashing the transformed data point, traversing the list of points in the hashed bin, and returning the closest sample point to the transformed model point. Since traversing the list of points within a search distance to a particular model point is still computationally expensive, the closest distance is pre-computed and stored in the the hash table. The closest distance is pre-computed as follows: for each bucket (i,j) in the hash table, convert the bucket location to a point (x_h, y_h) in the sample image; find the point in the hashed bin that is closest to (x_h, y_h) and less than the *search distance* and store the square of this distance in the hash table. If the distance is greater than or equal to the *search distance*, store the *search distance* squared in the hash table. This procedure introduces aliasing into the score function since the closest distance for all points in a particular bucket is quantized to a single value. Aliasing is acceptable in this case since the mapping is almost one to one. Thus, the quantized value gives a close estimate of the actual score function. Figures 3-32, 3-33, and 3-34 plot the comparison between between pre-computed closest distance (O) and on the fly computation of the closest distance (X). The pre-computed distance gives a result that is similar to the on-the-fly computation because the size of the hash table is chosen to be close the size of the actual model.

3.4 Verification Stage

The verification stage performs some fundamental checks to ascertain the consistency between the selected model and the sample data. The lowest model $\rightarrow$ sample match score is chosen with other models whose score are similar. If there are more than one model selected, the final arbitration is done by evaluating the sample $\rightarrow$ model match score at the best pose and selecting the model with the lowest score. The match process also returns the final pose as a by-product of the minimization. This final pose gives the transformation from the model data set to the sample data set. For pointing, it is necessary to find the coordinates in the sample data set that correspond to the tip of the index finger. Equation 3.6 describes the transformation from the model to the sample. Thus, the location of any feature point in the sample data can be extracted by applying the given transformation to known model feature points using the parameters returned by the match routine.

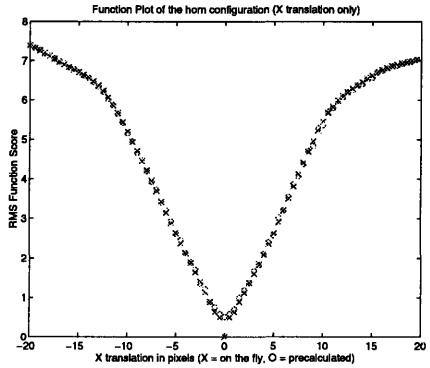

Figure 3-32: function plot of x translation - precalculated closest distance (O) vs. on the fly computation (X)

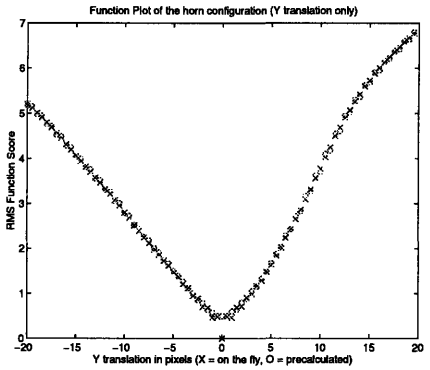

Figure 3-33: function plot of y translation - precalculated closest distance (O) vs. on the fly computation (X)

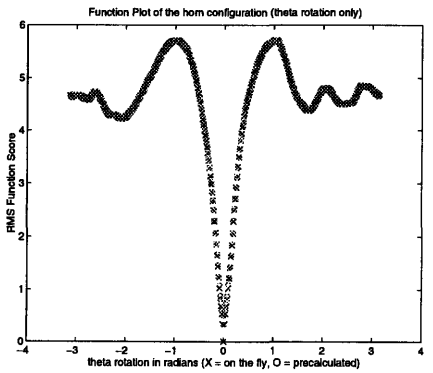

Figure 3-34: function plot of theta rotation - precalculated closest distance (O) vs. on the fly computation (X)

Chapter 4

Results

This chapter addresses the following issues: accuracy, sensitivity, selectivity, robustness, and problems encountered. For pointing, it is crucial to know how accurate the system is in recovering the tip of the index finger. In addition, we also want to know how selective the system is in choosing a model from the database that best resembles the sampled data. Since the system is sensitive to local minima, the match routine must start out close to the solution. Results on the reliability of the initial automatic alignment will be presented. A related question is sensitivity, in other words, how close must the initial alignment be in order to achieve a good match. Finally, results on the robustness of the system in handling changes in scale and variations in lighting is presented.

4.1 Model Database and Test Data

The model database consists of sets of (x,y) data points that represent the edge outlines of various hand configurations. The model images are obtained by using background differencing on an Indy Silicon Graphics workstation with a frame rate of 30Hz. Figures 4-1 through 4-13 are hand configurations that make up the model database used in presenting the results in this chapter.

Figures 4-14 to 4-16 represent typical experimental examples of hand configurations encountered by the system. Note that figures 4-17 and 4-18 have a number of points missing. In such cases, the proposed segmentation routine will not be able to cluster these sparse data points into one group. A possible alternative is to match combinations of clusters identified by the segmentation routine to the model database. This option was not explored due to real time constraints but the sparse data is shown to illustrate the match routine's ability to handle sparse as well as occluded data.

4.2 Accuracy

For the purpose of pointing, we would like to determine how accurate the system is in finding the location of the tip of the index finger. Figures 4-19, 4-20, 4-21 show the model, the sampled data, and the final alignment after matching the model to the

Figure 4-1: a configuration (model)

Figure 4-2: b configuration (model)

Figure 4-3: e configuration (model)

Figure 4-4: f configuration (model)

Figure 4-5: hook configuration (model)

Figure 4-6: horn configuration (model)

Figure 4-7: i configuration (model)

Figure 4-8: l configuration (model)

Figure 4-9: open hand configuration (model)

Figure 4-10: open palm configuration (model)

Figure 4-11: point configuration (model)

Figure 4-12: g configuration (model)

Figure 4-13: y configuration (model)

Figure 4-14: sample #1

Figure 4-15: sample #2

Figure 4-16: sample #3

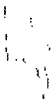

Figure 4-17: sample #4

Figure 4-18: sample #5

sampled data. The caption of the 3rd figure illustrates the error and the RMS score at the end of the match. Each category has two numbers; the first is found using pre-calculated closest distance while the second number in parentheses is the result obtained by using a method that searches all neighboring points within a certain distance to find the closest point. The RMS score for the two method is roughly the same but a significant amount of computation is saved by using the pre-calculated closest distance.

The error is the distance, in pixels, between the recovered index finger tip and the actual index finger tip. The tip of the index finger is manually selected in both the model and the sample data . The match routine outputs a set of transformation parameters that tells how to translate the model data points into the sample data points. This transformation is used to map the index finger tip of the model to a corresponding point in the sample data set. The Euclidean distance between this recovered point and the actual finger tip in the sample is the accuracy. Figures 4-21, 4-24, 4-27, 4-30, and 4-33 show that the accuracy is roughly the same as the RMS score. This is true only when there are no large errors or in general all errors are about the same size. Given an image size of 160x120 pixels with the hand occupying approximately 50x50 pixels, an average accuracy of two to three pixels away from the actual finger tip is useful in pointing. The system also does well in the last two cases where there are missing data. The error for the last case is high because the actual sample finger tip was chosen according to the segmented data; the real index tip is probably closer to the recovered index finger tip. Also notice that the RMS score for the last two cases (figures 4-30 and 4-33) is a high 3.7, 3.8. This is attributed to

Figure 4-19: model: point

Figure 4-20: sample #1

Figure 4-21: error: 1.2 (1.8) RMS: 1.2 (1.1)

Figure 4-22: model: point

Figure 4-23: sample #2

Figure 4-24: error: 2.4 (2.6) RMS: 2.4 (2.5)

Figure 4-25: model: point-side

Figure 4-26: sample #3

Figure 4-27: error: 3.8 (2.9) RMS: 1.9 (1.8)

Figure 4-28: model: point

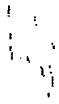

Figure 4-29: sample #4

Figure 4-30: error: 2.1 (2.4) RMS: 3.7 (3.7)

Figure 4-31: model: point

Figure 4-32: sample #5

Figure 4-33: error: 4.4 (4.8) RMS: 3.8 (3.8)

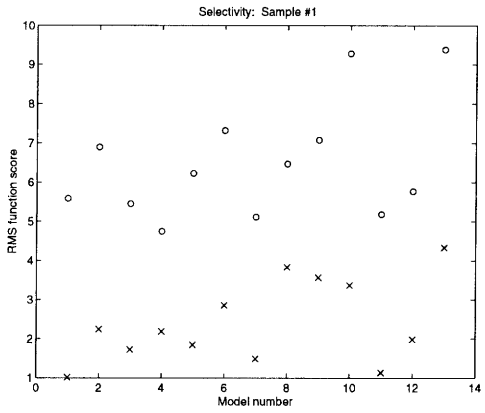

Figure 4-34: Plot of the RMS score between sample #1 and each model (x = RMS score of model to sample at best pose, o = RMS score of sample to model at the best pose)

the fact that the model, which has many points, is being matched to a sparse sample. Because there is not a one-to-one correspondence, the remaining points in the model are matched to the closest point in the sample, which is on the average 3.7 pixels away. To put the RMS score into perspective, an RMS for the best match is 0 and the RMS of a model hand configuration to a single point is approximately 9.5. There is not a clear-cut triage for the RMS score; it all depends on the density of the data and how stringent the match requirement is for the particular application.

4.3 Selectivity

Although the match system has an acceptable accuracy, an important question is how well does the system discriminate between different hand configurations?

Figure 4-34 plots the final RMS score of aligning each of the model in the model database (Figures 4-1 - 4-13) to sample #1 (Fig. 4-14). The (x) indicates the final RMS score of matching the model to the sample and the (o) is the RMS score evaluated in the reverse direction at the final pose. Figures 4-35 through 4-47 show the actual alignment at the end of the match between each of the model and sample #1. Although model #1 is very similar to the sample (RMS=1.0) in figure 4-34, model #11 (RMS=1.1) is the correct match. The reason model #1 has the best score is because we are only matching in one direction; model $\rightarrow$ sample. Hence, it is perfectly legal for model #1 to have the lowest RMS score since the sample overlaps model #1 and all the points in model #1 are accounted for. Therefore, a final verification step is required. That is, match each model to the sample, select the models with the lowest RMS scores, and arbitrate by performing one function call in the reverse direction by computing the function score of matching the sample $\rightarrow$ model. This step will give a sense of how the points in the sample are being matched to points in the model. In the example, matching sample #1 $\rightarrow$ model #1 gives

an RMS score of 4.6 while matching sample #1 → model #11 gives an RMS score of 4.0. Rather than computing the RMS score as a function of how well the model data points match to the sample data points, an alternative is to perform the match both ways and use the sum of the RMS as the scoring function. The former method works just as well in selecting the correct model and is more efficient since we are only computing the score function in one direction with one final function call in the reverse direction for a select number of models that have low RMS scores at the end.

Note the unusual alignment in figure 4-47. This is not so strange given the definition of the scoring function to be minimized. The match routine must find an alignment in which all the points in the model are matched to points in the sample with the minimal Euclidean distance between the two data set. The alignment in figure 4-47 is indeed the best possible arrangement such that all the points in the model are closest to the points in the sample.

To get an idea of the behavior of the system, each of the model is matched against every other model in the model database. The result is shown in the figures 4-48 and 4-49. Each row in the table shows a particular model matched against the models in the database. The first number is the RMS score of matching the model configuration specified by the row label to the model configuration specified by the column label. The second number is the RMS score evaluated at the best pose in the reverse direction. The diagonals are zero since the sample and model are identical. Note that the word match here means matching in one direction only; *a* match to *b* means measuring the distance, for all the points in model *a*, to the corresponding closest point in sample *b*. The reverse case, however, is not the same. Since the match is only in one direction, the table is not symmetric. Matching a set of points to its superset yields a low RMS. When matching in the reverse direction, points which are not in the intersection cause the RMS score to increase.

Model *a* shows a fist with an extended thumb. By inspection, there should be a low RMS score when matching model *a* → model *horn* and *a* → *point* because the points in model *a* are approximately a subset of these models. Though not as low as the *horn* and *point*, there should also be relatively low RMS scores when matching model *a* with the rest of the models. When matching in the reverse direction (ie. match each of the models to model *a*), many extra fingers in the rest of the models appear and cause a high RMS score. The table shows an average score of 5.0 for models not resembling model *a*. Similarly, model *b* has a low RMS when matched to model *f*, *open hand*, and *open palm*. Model *e* is close to *f* and model *f* to *open hand* and *open palm*. Model *hook* is close to *g* because they both have the same orientation although *g* has an extended forefinger.

At a glance, model *e* is expected to have a low RMS score when matched with model *b*, but this is not the case because the thumb is not present in *b*. Matching *y* → *hook* does not yield a low RMS score because there is a high concentration of points in the extended forefinger. The table illustrates that it is plausible to discriminate the different hand configurations by first matching in one direction, selecting models with low RMS scores, and then choosing the model with the lowest sample → model RMS score as the best match.

Figure 4-35: a conf to sample #1

Figure 4-36: b conf to sample #1

Figure 4-37: e conf to sample #1

Figure 4-38: f conf to sample #1

Figure 4-39: hook conf to sample #1

Figure 4-40: horn conf to sample #1

Figure 4-41: i conf to sample #1

Figure 4-42: l conf to sample #1

Figure 4-43: open hand conf to sample #1

Figure 4-44: open palm conf to sample #1

Figure 4-45: point conf to sample #1

Figure 4-46: g conf to sample #1

Figure 4-47: y conf to sample #1

	a	*b*	*e*	*f*	*hook*	*horn*
a	0.00	3.11	1.65	2.43	2.22	1.04
	0.00	4.44	5.13	4.28	3.76	3.77
b	4.15	0.00	1.78	1.03	2.14	2.20
	3.78	0.00	3.65	2.14	2.54	5.10
e	4.03	3.67	0.00	1.24	2.54	3.33
	3.53	2.84	0.00	4.38	2.41	4.65
f	4.11	1.99	2.02	0.00	2.84	3.03
	3.24	1.21	2.33	0.00	1.88	5.00
hook	3.53	2.45	1.77	1.86	0.00	1.74
	2.41	2.32	2.99	2.85	0.00	4.61
horn	3.48	4.38	3.10	3.06	4.21	0.00
	1.12	4.83	4.47	4.53	3.45	0.00
i	3.29	3.30	2.14	2.91	2.84	1.58
	1.43	2.46	2.44	3.14	1.85	3.24
l	5.40	5.59	4.47	3.68	5.07	3.77
	3.60	3.44	4.83	2.57	2.97	3.91
open hand	5.38	4.65	2.94	4.54	4.19	3.90
	2.72	2.97	2.03	3.38	2.20	3.50
open palm	4.65	3.16	3.32	2.90	3.53	3.38
	3.04	1.62	2.65	2.70	2.80	4.65
point	2.65	2.97	2.24	2.87	2.69	1.21
	1.33	5.63	3.52	3.82	3.12	3.53
g	3.91	2.84	2.83	1.98	2.39	2.89
	4.97	2.66	3.23	2.30	1.94	5.05
y	5.50	5.37	4.25	4.54	5.08	3.31
	3.47	3.89	4.57	4.30	3.58	4.59

Figure 4-48: *Table 1a:* the first number of each cell is the RMS score at the best pose, P, of matching the data set specified by the row label to the data set specified by the column label. The second number of each cell is the RMS score evaluated at the P in the reverse direction (ie. column label $\rightarrow$ row label)

	i	*l*	*open hand*	*open palm*	*point*	*g*	*y*
a	1.72	2.08	2.56	2.81	0.87	2.00	2.89
	5.98	6.00	5.37	5.28	3.13	5.67	7.40
b	1.70	3.01	1.03	1.33	2.43	1.66	2.57
	3.91	6.70	3.75	3.39	4.68	3.50	6.99
e	1.79	1.89	1.62	2.16	2.75	2.34	3.57
	2.51	5.01	4.43	3.97	2.96	3.61	4.80
f	2.63	1.88	1.17	1.52	4.30	1.90	2.46
	4.53	3.97	3.35	3.16	4.18	2.27	5.55
hook	1.40	2.52	2.20	2.03	4.20	1.37	1.98
	3.57	5.55	4.26	4.18	4.81	2.77	5.45
horn	2.82	3.52	1.86	3.01	4.98	4.31	3.91
	2.05	4.63	4.76	5.56	3.61	3.26	3.75
i	0.00	2.77	2.26	3.05	3.62	3.24	3.93
	0.00	4.66	3.73	3.70	4.96	3.20	4.75
l	4.54	0.00	1.34	3.10	4.48	3.93	4.31
	2.87	0.00	3.17	4.86	2.36	2.43	4.60
open hand	3.49	2.40	0.00	3.03	4.40	3.95	4.46
	2.56	3.37	0.00	2.45	2.68	2.71	4.05
open palm	3.33	3.16	2.36	0.00	3.73	3.15	3.53
	3.37	5.25	3.25	0.00	2.74	3.19	5.41
point	2.96	3.13	3.17	3.44	0.00	2.50	3.28
	5.45	5.95	5.42	4.93	0.00	5.49	6.72
g	2.19	2.35	2.15	2.65	5.15	0.00	3.39
	4.97	4.03	4.22	3.51	5.34	0.00	4.97
y	4.04	3.57	2.81	4.92	5.23	4.96	0.00
	4.43	5.08	3.96	4.50	4.45	3.97	0.00

Figure 4-49: *Table 1b:* the first number of each cell is the RMS score at the best pose, P, of matching the data set specified by the row label to the data set specified by the column label. The second number of each cell is the RMS score evaluated at the P in the reverse direction (ie. column label $\rightarrow$ row label)

4.4 Initial Alignment, Sensitivity Analysis

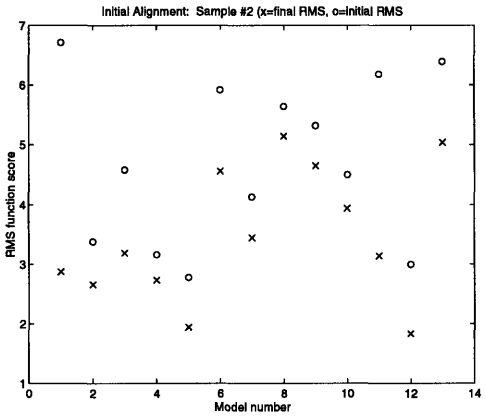

Figure 4-50: Automatic initial alignment. (x = final RMS score, o = RMS at initial guess).

Figure 4-50 plots the initial alignment of each model to sample #3 (4-16. The x-axis is the model number and the y-axis is the RMS function score. The (o) is the function score at the initial alignment and the (x) is the function score at the end of the minimization process. For the relevant models, the initial alignment provides a pose estimate which is close to the final solution. Since the match routine begins near the final solution, it will not take many iterations to arrive at the final solution.

A more interesting question is how good must the initial alignment be in order for the system to achieve a good match. Figures 4-51, 4-52, 4-53 are plots of the recovery of model 'point-side' matched against sample #3 with a perturbation from the best pose along each of the x, y, and θ dimensions. Note that the RMS score at the best pose is 1.8. The system is able to recover the same RMS score within a range of $\pm$ 8 pixels in the x dimension, $\pm$ 20 pixels in the y dimension, and $\pm$ 1.3 radians (74 deg) in the theta dimension. For a search range of 10 pixels in the x and y dimensions, the plots show that the recovery is not extremely sensitive to a good initial alignment. Perhaps the reason why the y dimension has a lower sensitivity range is because there are more points in the y dimension; in sample #3 the x range is 30 pixels while the y range is 70 pixels. Figures 4-51 through 4-53 only portray perturbations in one dimension. What happens when all three dimensions are perturbed? Figure 4-54 shows the RMS recovery for random perturbations of $\pm$ 10 pixels in the x dimension, $\pm$10 pixels in the y dimension, and $\pm$ .9 radians (52 degrees) in the theta dimension. There are 7 cases that fail; probably some of these move the model more than 8 pixels in the x dimension. Some of these failed cases may be due to a local minimum. Although the system is not very sensitive to the initial alignment, it is worthwhile to initialize the match as close to the final solution as possible so that the match routine will converge faster.

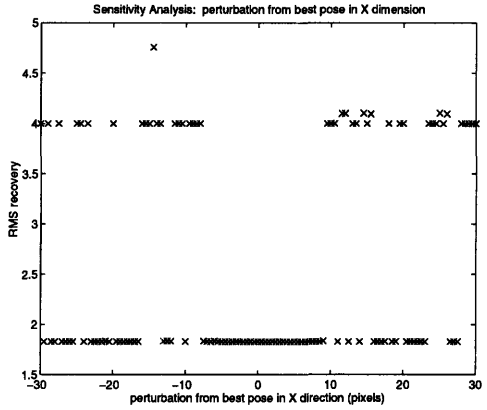

Figure 4-51: RMS recovery of sample #3 after perturbation from the best pose in x dimension. RMS score at best pose = 1.8

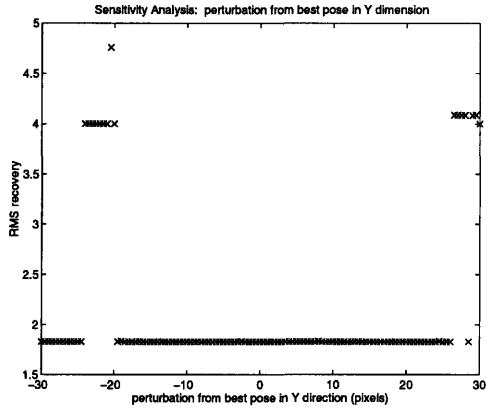

Figure 4-52: RMS recovery of sample #3 after perturbation from the best pose in y dimension

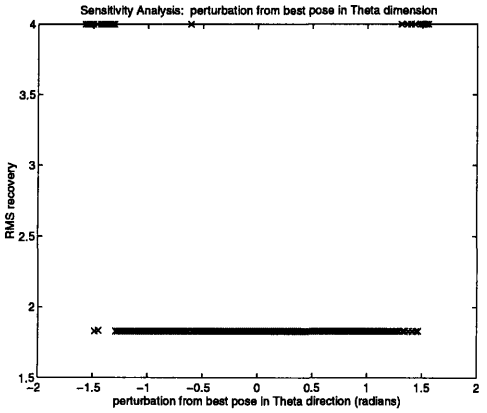

Figure 4-53: RMS recovery of sample #3 after perturbation from the best pose in theta dimension

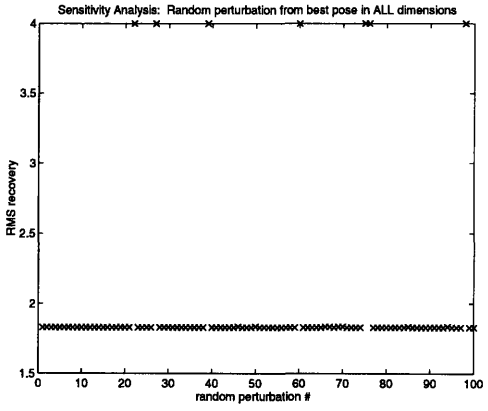

Figure 4-54: RMS recovery of sample #3 after random perturbation from the best pose in all dimensions. Range of random perturbation: $x = \pm$ 10 pixels, $y = \pm$ 10 pixels, $\theta = \pm$ .9 radians (52 deg)

4.5 Robustness

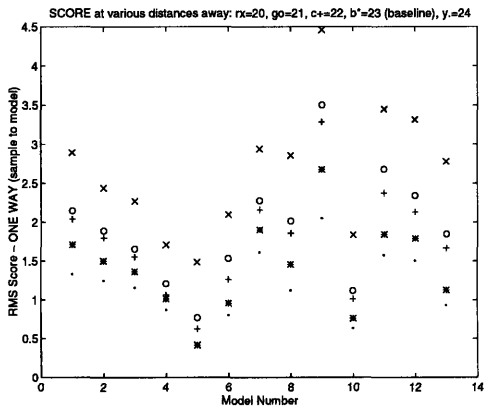

Figure 4-55: Plot showing final match score at various scales

Besides being able to handle occlusions, the system can handle small changes in scale, variable lighting, and a range of rotations within the plane. Since the match routine is based on hand outlines, the segmentation routine is the key in variable lighting. The adaptive background varies with changes in lighting as long as the background does not change too frequently. If the color of the hand is sufficiently different from the color of the background, the outline of the hand will be extracted. As previously shown, the match routine can handle a range of rotations within the plane. Figure 4-55 shows the final RMS score of the hand in a hook configuration at variable scale matched against the model database. The (*) plots the comparison between the model database and the hook taken at the baseline scale where all the other models are taken. The (.) is the hook at 1 inch below the baseline; the hand is farther away from the camera and the image is smaller than the image at the baseline. The (+) is 1 inch above the baseline, (o) 2 inches, and (x) 3 inches. At each scale, the system correctly identifies the *hook* model as the match with the lowest RMS score. The score increases when the scale is different from the baseline because all the points are not directly overlaid on top of the *hook* model. Samples taken further away from the baseline are enclosed inside of the *hook*. Samples taken closer than the baseline form a boundary outside the *hook* similar to the way in which the model 'point-side' forms a boundary around sample #3 in figure 4-27.

4.6 Problems

There are a number of weaknesses with this approach: local minimum, poor initial alignment, and rotation out of the plane. The minimization routine could return a local minima that is not the global best fit. The plot in figure 4-54 displays the times when a local minima appears in the match routine. This, however, only occurs with a poor initial alignment. The problem can be rectified with better initial alignment techniques. Perhaps a Gaussian weighted distance function can be used to pull the

solution in closer to the actual result. Another solution is to use simulated annealing to kick the system out of the local minima at the expense of efficiency.

The minimization method requires the estimate of the starting pose to be close to the final solution. Missing data can drastically change the centroid and axis of least inertia and will result in a poor initial alignment. An alternative is to use the Hough transform to identify long lines in the sample and model and match these lines.

Since this technique is designed to handle 2D pointing, it does not handle rotations out of the plane. New models must be added to the model database to recognize out of plane rotations.

Chapter 5

Future Work

Since this work suggests a method for solving the 2D pointing problem on a flat surface, the next logical step would be to tackle the 3D pointing problem and to apply it to directing the Intelligent Room's attention to some part of the room or to some object in it. It is difficult, even for humans, to recognize what a person is pointing at in 3D. In some cases, following the ray of the person's finger or arm may not be sufficient to distinguish the object being identified. The observer may have to resort to clues from the context of the conversation or the focus of the speaker's eyes. Another crucial aspect is determining which camera(s) from a set of cameras provide a clear view of the person's arm and hands; a view of the person from the back would be useless. Relating the information from the multiple views to extract 3D information is also a very difficult problem. Detecting 3D pointing requires resolving many issues and is a challenging problem to solve.

A related problem to model-based recognition is the size of the model database. As the model database expands, it is necessary to consider ways to minimize the search. One scheme is to use the current scoring function as a means of clustering the models in the database into different groups. A member with the lowest correlation score to all of the other members of the same group is chosen as the representative member for each group. The groups can be subdivided into subgroups to form a search tree based on the representative members of each group. For a quick search, greedy decisions can be made at each branch of the tree. With a fast search engine in place, it will be interesting to add a Hidden Markov Model or finite state machine on top of this framework and explore the limitations of this approach in performing real time dynamic gesture recognition.

Bibliography

[Baudel and Beaudouin-Lafon, 1993] T. Baudel and Charade M. Beaudouin-Lafon. Remote control of objects using free-hand gestures. *CACM*, pages 28–35, July 1993.

[Darrell and Pentland, 1993] Trevor J. Darrell and Alex P. Pentland. Recognition of space-time gestures using a distributed representation. Technical Report No. 197, MIT Media Laboratory, 1993.

[Davis and Shah, 1993] James Davis and Mubarak Shah. Gesture recognition. Technical Report CS-TR-93-11, University of Central Florida, Orlando, FL 32816, 1993.

[Freeman and Roth, 1994] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. Technical Report Technical Report 94-03, Mitsubishi Electronic Research Labs, 201 Broadway, Cambridge, MA 02139, 1994.

[Freeman and Weissman, 1995] W. T. Freeman and Craig D. Weissman. Television control by hand gestures. In *Workshop on Automatic Face and Gesture Recognition, Zurich, Switzerland*, June 1995.

[Gao and Wang, 1995] Wen Gao and Shuanglin Wang. Independent hand gesture recognition in random still background. Technical report, Harbin Institute of Technology, Harbin 150001, China, 1995.

[Grimson and Lozano-Pérez, 1987] W.E.L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 9(4):469–482, 1987.

[Grimson *et al.*, 1993] W.E.L. Grimson, T. Lozano-Pérez, W.M. Wells III, G.J. Ettinger, S.J. White, and R. Kikinis. An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization. *CVPR*, pages 430–436, 1993.

[Grimson, 1990] W. Eric L. Grimson. *Object Recognition by Computer: the role of geometric constraints.* MIT Press, 1990.

[Huttenlocher and Ullman, 1987] D. P. Huttenlocher and Shimon Ullman. Object recognition using alignment. *First International Conference on Computer Vision*, pages 102–111, 1987.

[Huttenlocher and Ullman, 1990] D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. Journal Computer Vision*, 5(2):195–212, 1990.

[Lamdan and Wolfson, 1988] Y. Lamdan and H.J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. *Second Int. Conf. Comp. Vision*, pages 238–249, 1988.

[Nowlan and Platt, 1992] Steven J. Nowlan and John C. Platt. A convolutional neural network hand tracker. Technical report, Synaptics, 1992.

[Press *et al.*, 1992] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992. New York, NY.

[Rehg and Kanade, 1993] James M. Rehg and Takeo Kanade. Digiteyes: Vision-based human hand tracking. Technical Report CMU-CS-93-220, Carnegie Mellon University, December 1993.

[Starner and Pentland, 1993] Thad Starner and Alex Pentland. Visual recognition of american sign language using hidden markov models. Technical report, Perceptual Computing Section, MIT Media Laboratory, 1993.

[Ullman, 1987] Shimon Ullman. An approach to object recognition: Aligning pictorial descriptions. Technical Report No. 931, MIT AI Lab Memo, 1987.

[William M. Wells, 1993] III William M. Wells. *Statistical Object Recognition.* PhD thesis, M.I.T, 1993.