VISUAL HAND TRACKING AND GESTURE ANALYSIS

BY

JOHN LIN

B.A., University of Illinois, 1998
M.S., University of Illinois, 2000

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

ABSTRACT

The emergence of new display and computing technologies have brought attention to new paradigms of human-computer interactions that are fundamentally different from using traditional mechanical input devices. Communication through visual gesture interface is a potential alternative that could provide multidimensional interactions. Though rich information is embedded in video input, so is a large amount of noise and uncertainties. Extracting correct and useful information from a video is a challenging problem, and this makes capturing the human hand motion an extremely difficult task.

This dissertation presents novel techniques to effectively and efficiently track articulate hand motion in a video sequence. Since tracking articulate hand motion is equivalent to a search in a high-dimensional space, an analysis of the feasible configuration space is given, and semiparametric and nonparametric representations are proposed to model the motion constraint. For each of the representations, a corresponding efficient tracking algorithm is proposed. Experiments have shown the effectiveness of these approaches. To fully automate the tracking system, this dissertation also presents an algorithm for automatic initialization which can be done in real time. Based on this result, several vision-based gesture interfaces are built.

To my parents.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my adviser, Professor Thomas S. Huang, for his guidance, support, and freedom throughout my graduate studies. It has been a privilege to be his student. I would also like to express my gratitude to my mentor and coadviser, Professor Ying Wu, for his guidance and patience all these years. I would like to thank my other committee members, Professor Narendra Ahuja and Professor Seth Hutchinson, for their valuable advice and discussions during my study and their extreme tolerance during the production of this dissertation. I have benefitted enormously from my summer internship experiences and special thanks are extended to my mentors, Dr. Paul Viola, Dr. Arun Hampapur, Dr. Sharat Pankanti, and Dr. Rudolf M. Bolle, for their selfless discussions and guidance. I would also like to thank Professor Robert Fossum for the enlightening discussions in mathematics.

Many colleagues in the Image Formation and Processing group at the Beckman Institute deserve much thanks. In particular, I would like to thank Dr. Vladimir Pavlovic, Greg Berry, Dr. Qiong Liu, Wilfredo Almaguer, Dr. Qi Tian, Dr. Pengyou Hong, Dr. Steven Chu, Dr. Munehiro Nakazato, Dr. Zhen Wen, Nemanja Petrovic, Hanning Zhou, Dennis Lin, Terrance Chen, Aleksandar Ivanovic, Charlie Dagli, and the system administrator Gabriel Lopez-Walle. Special thanks to Howard Huang for his friendship and for helping me to proofread many of my papers.

I wish to thank my family for their support and encouragement throughout my entire study here at UIUC. I would also like to thank Irene Chan for her love, support, and tolerance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 Human-computer interaction

As computers play an increasingly important role in our lives due to the rapid advancement in technologies, more efforts have been devoted into studying and developing natural human-computer interaction (HCI) techniques. Current means of reliably interacting with the computer still rely on mechanical devices. Examples of primary devices include keyboards, mice, and joysticks, which are essential to communicate with the computer. While these devices provide accurate and reliable information flow, it is not how humans naturally communicate in daily life. The devices require direct contact from the user in order to operate them. This limitation might appear minimal when used with a PC. But as the computers become more powerful and affordable, new functions of computers have emerged that do not restrict the user to sit in front of the computer and communicate using traditional I/O devices such as monitors and keyboards. This can be best seen with the examples in virtual reality (VR) environments, wearable computers, and augmented reality (AR). For another instance, during a presentation, it would be desirable if a user could operate the computer from an arbitrary location, rather than having to walk back and forth to update the slide or having to carry a remote control in the hand all the time.

In the case of VR environments, the attempt is to simulate the reality through computer graphics synthesis. It would be desirable to allow the user to carry minimal equipment and at the same time communicate with the environment in a natural manner. Common devices for VR are the wand and magnetic sensors for retrieving motion parameters of the user. Users must carry a wand in the

hand and have the magnetic sensors attached to the body at all times. Though improvements can be made to use wireless transmission to remove the physical connections with the controllers, such devices are still cumbersome to use. Also, these devices are not capable of providing complicated commands due to the number of degrees of freedom (DOF) of the inputs. Furthermore, they cannot achieve immersions.

1.1.2 Vision-based systems

Visual input allows the computer to mimic natural human sensing mechanism using the eye. Natural human interactions do not require the information flow through the use of keyboard and mouse because eyes and ears are capable of collecting rich information from the environment. The computer, in principle, should be able to interpret and understand human motion through video input, which provides a rich information of the environment.

Some factors must be considered when designing and implementing vision-based algorithms:

- *Robustness and accuracy.* Video inputs are often noisy due to sensor noise and changes in environment variables such as lighting, background composition, and occlusions. The algorithms should be able to robustly extract the relevant information and infer the correct model state.

- *Speed.* To have a working system, it is necessary to have a real-time processing rate or at least close to the video-frame rate.

- *Implementation.* The algorithms should be conceptually straightforward, and the implementation should be easy to adapt to other scenarios.

Many earlier algorithms proposed using markers to aid in processing. While this offers a strong cue for visual processing, it could be prohibitive and cumbersome for users to attach markers to their hand, face, or body every time they use a computer. Recent research focuses more on markerless algorithm and use markers only to help collecting training data.

There are several advantages associated with vision-based algorithms:

- *Nonintrusiveness*: The main advantage of using vision-based techniques is its nonintrusiveness. The user can operate the computer at a distance without requiring the physical contact

with any device, sensors, or markers. This provides a more natural interaction with the computer and a more immersive experience in the virtual environment.

- *Rich information*: Compared to the data input through keyboard and mouse, a camera is capable of delivering much richer information from the two-dimensional (2D) arrays of color signals. Furthermore, in addition to receiving the discrete digital data directly, higher-level semantics can be inferred from the video sequences. This offers a much more versatile input compare to traditional mechanical devices which are limited for their specific functions. Vision-based algorithms also allow for a wide varieties of applications such as teleconferencing and biometrics.

- *Low cost*: Digital cameras and camcorders are becoming more affordable and the video qualities are becoming better. Many PC vendors now include an option for the customers to choose a digital camera when ordering the PC. A USB webcam can now be bought for about $100 whith video resolutions up to 640×480 pixels. A rapidly increasing number of cellphones nowadays are also equpped with camera as a standard option. It is not difficult to foresee that cameras will be the standard equipment for many electronics such as PCs, TVs, projectors, and cellphones.

1.1.3 Visual gesture interfaces

A significant amount of research has been devoted in making a real-time and robust gesture interface realizable [1, 2, 3, 4]. Among the different modalities humans utilize in natural communication, the use of hand gestures plays a major role, largely due to the dexterity of the human hand. The dexterity of the hand is reflected among the numerous mechanical devices that exploits accurate hand control such as the mouse, keyboard, wand, trackball, touch pad, touch screen, joystick, and pen-enabled devices. Using hand gestures to communicate not only is intuitive but also provides a multidimensional input [5].

To reliably capture the hand motion, many current systems employed glove-based devices. Joint angles and hand shape is measured by mechanical or optical sensors. The position of the hand can be determined by using an additional magnetic sensors attached to the wrist of the glove. This approach can determine hand postures with high speed and accuracy, and many gesture recognition

algorithm has been reported to work with it. However, the use of such a device is cumbersome and does not allow an immersive feeling for HCI. The cable connecting the glove and controller limits the user motion and using wireless transmitting devices increases the already high costs that prohibits widespread use. Vision-based algorithms, on the other hand, provide a promising alternative for capturing hand motion.

1.2 Motivation

Among different modalities involved in natural HCI, the human hand provides an extremely expressive tool that is capable of transmitting complex information. However, this powerful feature in expressive signing also makes it very difficult for computers to infer the true hand state from a sequence of 2D images. To accurately capture the hand motion involves interdisciplinary research including computer vision, computer graphics, statistical learning, and pattern recognition. A complete visual hand motion capturing system would consist of several building blocks. Here we explore a few of the issues that motivate our research.

1.2.1 Tracking in high-dimensional space

A hand is a highly articulate object having roughly 27 DOF. Recovering the shape of such a complex 3D structure from a 2D image is a difficult problem. To accurately estimate the hand state at any time instant requires a search in the high dimensional space. Current computing technology is still prohibitive to allow a full range exhaustive search in $\mathcal{R}^{20}$. Although performing tracking rather than frame-by-frame recognition helps to greatly reduce the search space, it is still a formidable task without the use of any prior knowledge.

Fortunately, the hand motion is highly constrained. By incorporating the constraints in the tracking, we could greatly reduce the search complexity. In particular, we must address two key issues:

- *Modeling the feasible space.* We would like to identify a compact motion constraint representation to effectively model the feasible configuration space. Clearly, feasible hand shapes do not occupy a filled region in $\mathcal{R}^{20}$. There are holes due to the impossible hand shapes and

there are sets of configurations that are more naturally and frequently performed in everyday tasks.

- *Design a corresponding search algorithm.* Given a motion constraint model, we would like to design an appropriate tracking algorithm that can track efficiently using this structure. Though many existing tracking algorithms can handle generic tracking tasks, improvements can be made for different class of scenarios if prior knowledge can be incorporated.

1.2.2 Automatic initialization

To complete a fully automated hand tracking system, the computer must be able to automatically initialize by acquiring the hand geometry and initial motion parameter. With the complexities involved in hand tracking alone, the initialization has often been treated as a sperate problem and has been done manually. This is partly because most of the tracking algorithms still cannot be done in real time and are tested only on recorded sequences. Consequently, the initialization can be made as accurately as possible to ensure the quality of subsequent tracking phase. However, as the computers become more powerful and better algorithms are designed and proposed, it will soon be common to have high-speed articulation tracking algorithm, and manual initialization will be the a cumbersome bottleneck whenever a tracking must be performed.

1.3 Applications

The hand motion recovered from video sequences can be used to interact with computers in a more natural way. There are several scenarios in which hand tracking can provides a promising new alternatives. Replacement of the wand by bare hand for communications in VR environments provides the users with a more immersive experience. Furthermore, the complexity of hand motion allows for a large variety of controls while the mouse can only provide two DOF. Similarly, the user can interact with virtual objects in an AR environment in which the perception of the real world is enhanced with additional virtual objects.

The tracked results can also be used for synthesizing realistic hand motion animation. Besides glove-based techniques, commercial products are also available that make use of vision-based

techniques to capture human motion such as Vicon Motion Systems, Motion Lab Systems, Motion Analysis Corporation, etc. Though reliable tracking results can be obtained, these systems requires the use of markers with careful design in the their positions on human body every time is collected. A markerless algorithm is certainly a more attractive alternative that can provide a less cumbersome method.

From the recovered motion parameters, gesture recognition can be applied and the results could be used directly in controlling a remote device, teleconferencing, and American Sign Language (ASL) encoding.

Last but not the least, with a robust detection and tracking algorithm, the input source can rely on camera rather than keyboards and mice. This results in less peripheral units that must be equipped with a computer. Consequently, the computer no longer needs to be in the vicinity of the user in order to be operated. It could be integrated with any equipment. Also, a camera is much smaller than a keyboard and a mouse.

1.4 Organization

This dissertation will present novel work in several aspects that will help to make model-based hand tracking possible by attempting to reduce the computational complexity and proposing more efficient algorithm for tracking. To complete the study of the system, a real-time automatic model initialization algorithm will be presented. Based on the results of the hand detection algorithm, several gesture interface systems will be described. The remainder of this dissertation is organized as follows:

- Chapter 2 provides an overview of the vision-based hand tracking system and the state-of-the-art research in this area. Techniques for hand localization and segmentation is presented in Section 2.2. The model-based and appearance-based approaches are discussed in Section 2.5.

- Chapter 3 presents the analysis of the hand motion constraints and how they can be used to help reduce the computation complexities in tracking. An overview of the constraints is introduced in Section 3.2. Section 3.3 describes the approach to learn the constraints from

real hand motion data and two representations of the learned constraint model are given in Section 3.4 and 3.5.

- Chapter 4 focuses on the tracking algorithm. The sequential Monte Carlo (SMC) method is introduced in Section 4.2. Section 4.3 described how semiparametric representation can be incorporated in SMC tracking for capturing finger articulation. The local and global parameters are combined in an iterative procedure described in Section 4.4. Tracking results and analysis are shown in Section 4.6.

- Chapter 5 extends the original formulation of the SMC tracking by incorporating the Nelder-Mead (NM) simplex search. Details of the NM algorithm is presented in Section 5.3. In Section 5.4, an adaptation of the NM is given to incorporate the nonparametric representation of hand motion model for efficient hand motion tracking. The new algorithm, called stochastic Nelder-Mead (SNM) search, is given in Section 5.5. Results of using SNM for tracking is shown in Section 5.6.

- Chapter 6 describes an automatic initialization algorithm that can be used to automate the model-based tracking system. The steps for identifying the palm and fingertips and the intermediate detection results are shown in Section 6.4 and 6.5. In Section 6.6, an algorithm is presented for identifying the Thumb. Section 6.7 combines all detection results to automatically construct a hand model.

- Chapter 7 presents several applications using the vision-based gestural interface for natural HCI. In particular, the following systems are described in Sections 7.2 - 7.5: virtual object manipulation, virtual terrain navigation, finger drawing, and static gesture recognition.

- Chapter 8 concludes the dissertation with the discussions of natural hand motion constraints, the tracking algorithms, and the automatic initialization algorithm. Proposals for future research topics are given in Section 8.2.

1.5 Contribution

In this dissertation several contributions are made to address the issues involved in an automated real time visual hand tracking. Specifically, the following topics are addressed in this work:

- A novel nonparametric representation is proposed to model the feasible hand configuration space learned from real hand motion data.
- A novel tracking algorithm, SNM search, is proposed to efficiently track human hand motions using the new configuration representation. The algorithm can also be applied to other general tracking scenarios.
- An extremely fast and robust hand detection and initialization algorithm is proposed. The algorithm can be used to fully automate the hand tracking system.
- Several vision-based HCI interfaces using the hand detection results are developed. Depending on the application scenario, different sets of gesture grammar are designed to provide a natural interface while simultaneously ease the user learning and adaption.

The motion constraint analysis and associated tracking algorithms presented in this dissertation are by no means limited to hand tracking. The proposed algorithms can be generalized to tackle other tracking algorithms involving high dimensional state space such as body tracking. The SNM is especially suitable for the cases when the objective function to be minimized is highly nonlinear. While the proposed gesture interface applications are more task specific, the design of the systems can be continued to be improved independent of hand detection and tracking algorithms.

CHAPTER 2

VISION-BASED HAND TRACKING

2.1 Introduction

Developing a fully functional gesture recognition system involves research in image processing, computer vision, machine learning, pattern recognition, statistical learning, and biomechanics. A complete description of a gesture consists of the temporal and spatial information of the arm and body motion. One crucial component in extracting the gestures from video input is to estimate the hand configurations. The hand motion generally consists of 27 DOF, 6 DOF for the global motion and 21 DOF for the finger articulation.

In this chapter we give an overview of the visual gesture tracking system (Figure 2.1). In the following chapters we describe the function of some of the key modules including various skin segmentation algorithms, a survey of different features and hand models used, and two approaches to recover hand motion.

2.2 Hand Localization

The first preprocessing step in the hand-tracking system is the hand localization. The performance of subsequent processes greatly depends on the result of segmentation. Object segmentation is a challenging problem especially when the video consist of complex background, changing lighting environment, occlusions, and complex motion. Many different features such as edge, motion, color, and texture can be used to segment a hand object and provide a bounding box for that region. It is common to combine several cues to improve the robustness of segmentation [6, 7, 8, 9, 10]. However,

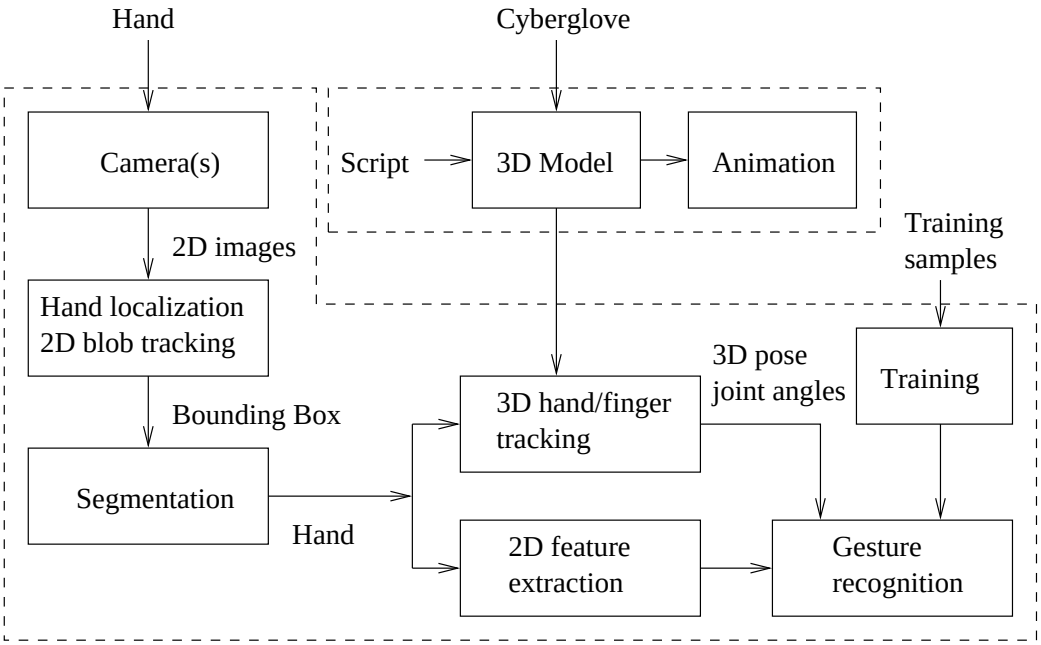
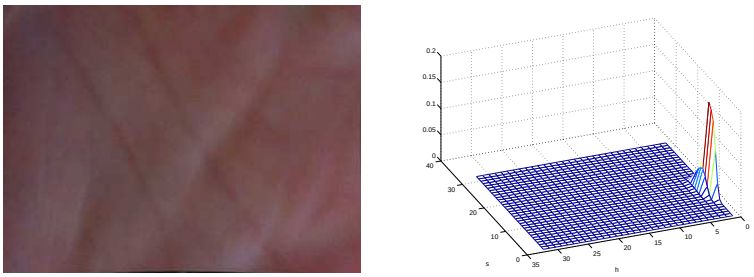


Figure 2.1 System overview.

how to effectively fuse information provided by different types of features remains a challenging problem. Depending on the application scenario, sometimes one single feature would be sufficient, and among the low-level features, color provides a strong cue in segmenting the foreground objects [11].

The skin color is often used in the preprocessing stage to provide initial hand location and segmentation [12, 13, 14, 15, 16]. Two problems must be addressed when using skin color as the segmentation cue. First, the lighting is often nonstationary and the foreground color distribution is constantly changing. Second, to achieve user-independence, the skin color model must identify invariance across different users. For the second problem [17] gives an analysis by collecting a large amount of skin data and shows that skin color distribution tends to cluster in the hue-saturation-intensity (HSI) space regardless of the race. An example of a hand image with its histogram distribution in color space is shown in Figure 2.2. As such, many HCI systems rely on the use of skin detection to locate human hands and faces as a preprocessing step, followed by tracking and recognition of the segmented object.

To model the image representations in color space, different approaches were proposed. The goal is to learn a compact and representative structure to model the color space such that it is



(a) (b)

Figure 2.2 (a) A hand image. (b) Color histogram in Hue-Saturation plane.

most efficient for segmentation and tracking. As observed in previous work, pixels tend to cluster in color space due to the similarity of appearance for the same object. One approach to model the color space is to use Gaussian mixture models[18]. The model order can be learned from a fixed set of data and the model parameters are updated online to cope with nonstationary lighting conditions. However, the new training data obtained from the new frame is not reliable since the new frame is not segmented.

The other alternative is to use a nonparametric model to characterize the color content. In [17], a color histogram is learned from a large amount of database, and Bayes rule is used to classify the foreground region. Although this model is learned in order to capture the invariance across different users, it is not able to handle change in lighting conditions. In [15], an algorithm is proposed to adapt the histogram distribution from frame to frame to handle nonstationary lighting conditions.

2.2.1 Self-organizing map

Though histograms provide an easy way to model the color distribution, a proper quantization level needs to be defined for optimal segmentation. Furthermore, an unnecessarily large data structure is created and maintained when the foreground pixel might only cluster within one small region. One alternative is to use a self-organizing map (SOM) [19]. The principal goal of the SOM is to transform an incoming signal pattern of arbitrary dimension into a lower dimensional space. In this case, the input consist of a vector of all pixels in HSI space. Detail description of an adaptive version of the SOM algorithm can be found in [20].

Some images of the skin segmentation are shown in Figure 2.3. Because no texture or spatial correlation information is used, the results are noisy. The top row shows the original image, and

the bottom row shows the segmented image. It can be seen that the hand can be segmented under different orientation with variations in skin colors. The algorithm also detects the skin region of the face, and can be used in a face detector.

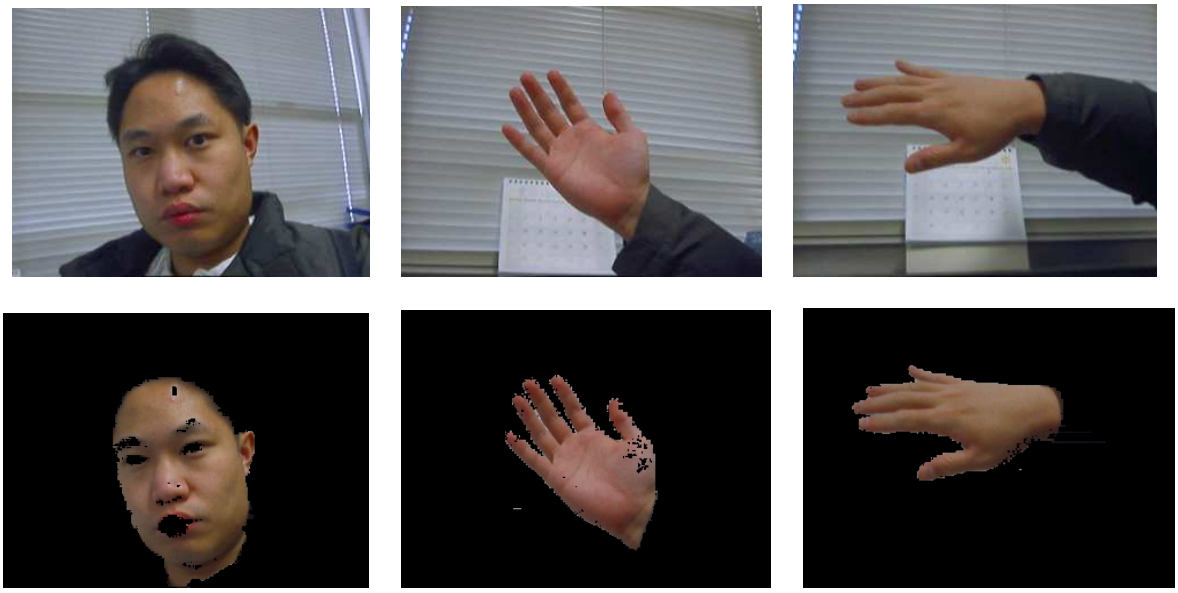


Figure 2.3 Skin segmentation using SOM.

2.2.2 CAMSHIFT

Another nonparametric approach is the Continuously Adaptive Mean Shift (CAMSHIFT) algorithm [21], which is derived from Mean Shift [16]. The basic idea of the Mean Shift algorithm is to climb the gradient of a probability distribution to find the nearest dominant mode. First, choose a search window size and initial location. Then compute the mean location inside the search window and update the location of the search window to center on the mean. For the case of discrete 2D image probability distributions, the mean is determined by

$$x_c = \frac{M_{10}}{M_{00}}, y_c = \frac{M_{01}}{M_{00}},$$

where

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

and $I(x, y)$ is the probability value at location (x, y). The original Mean Shift algorithm is designed for static distributions. The CAMSHIFT algorithm modifies this algorithm to work for dynamically changing distributions. In the tracking scenario, an object is constantly moving, and the composition of the image content is constantly changing. The hand could be closer to or further away from the camera and occupy a different size area in the frame. Therefore, a fixed window approach cannot be sufficient for the tracking task. The CAMSHIFT extends the Mean Shift algorithm by including an additional step that would dynamically update the size of the search window based on the zeroth moment information in the current search window. The color probability distribution is also computed within a 2D region centered at the search window location and slightly larger than the window size. By doing so, computation can be saved by not having to evaluate the color probability distribution over the entire image.

When brightness is low, saturation is also low, hue then becomes noisy, since there are not enough number of discrete hue pixels to adequately represent slight changes in red-green-blue (RGB) color. This then leads to large fluctuations in the values. To overcome this problem, the hue pixels are simply ignored if they have very low corresponding brightness values. Some segmentation results using CAMSHIFT is shown in Figure 2.4.

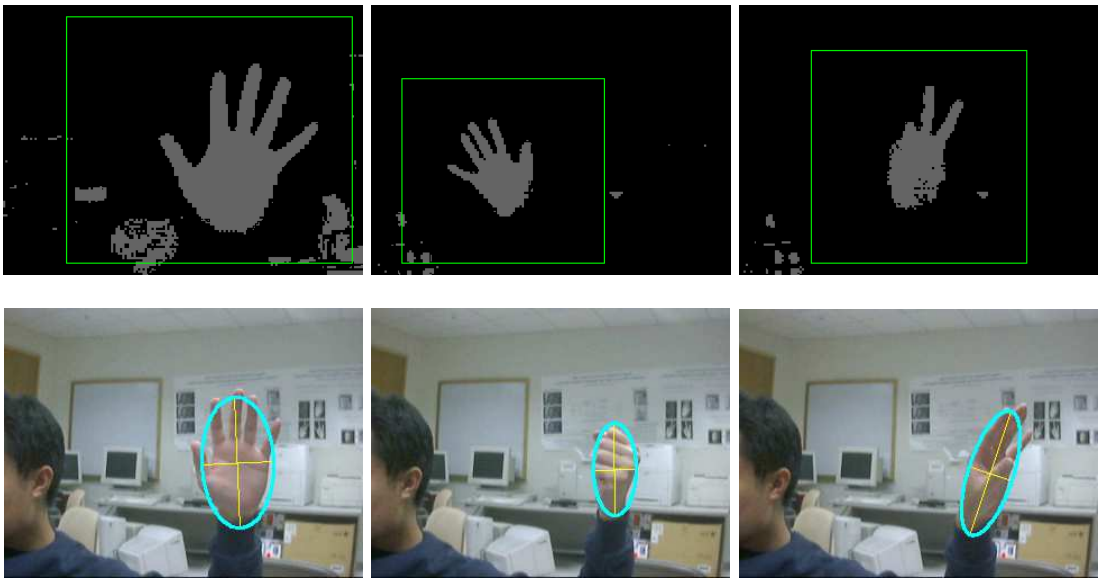


Figure 2.4 Results of segmentation using CAMSHIFT. The top row shows some frames with the hand skin region correctly segmented. The second row shows frames from another sequence in which the bounding region is modified at each frame to include proper area of the foreground.

2.3 Feature Extraction

After the hand region is identified, the next step is to extract meaningful features for tracking and recognition. Many low-level image features are commonly used in recognition, such as Gabor wavelet filter coefficients, Fourier descriptors, moments, silhouettes, contours, and eigenvalues extracted using PCA [22]. For model-based hand tracking, geometric features play an important role in inferring the exact hand shape. The detection of fingertips has shown to be sufficient for recognizing many gestures when motion constraints are considered [23, 24]. Earlier work use color markers to help correctly locate fingertip positions [23, 25]. When the application requires the knowledge of fingertip locations only when fingers are extended, the task for searching fingertips is simplified and reliable markerless algorithms can be implemented using template matching techniques. [26, 27]. The finger edges can also serve as additional cues for gesture recognition and model fitting [28, 29]. In [30, 31, 32, 33], contours of the hand are used to distinguish different hand shapes. Range data is also used in addition to color information in [10]. In [9], optical flow is computed from the image gradient in order to recover the hand motion. For view-independent gesture classification, [34] proposes to use the seven Hu moment invariants [35] extracted directly from silhouettes. The advantage of using Hu moments is its ease of implementation and the invariance to translation, rotation, and scaling. However, these invariants also limit the ability to recover the global hand pose.

2.4 Hand Model

The high dexterity of the hand makes it an important instrument in a wide range of fields. Depending on their specific functions, different disciplines employ a variety of hand models with different levels of details (LOD). In most cases, a kinematical model, abstracted as a stick figure, is sufficient to serve the purpose of understanding and synthesizing the articulated hand motion. In many medical studies, the main concern is the exact measurements of the hand skeleton such as locations of bone segments and joints, and length of each finger segment. The majority of the data collected are intended for the hand biomechanics studies, and exact appearance of the hand is not modeled. On the other hand, the goal in computer graphics is to achieve not only lifelike

motions, but also synthesis of realistic appearance. In this respect, details such as skin texture, skin deformation, and muscle deformations are important in modeling accurate hand structure and movements.

In computer vision, where models are used to help analyze the hand motions in a video sequence, a kinematical model for motion analysis and a shape model for the appearance are both needed. Unlike the high LOD required in the computer graphics, the model should have only sufficient details to mimic the appearance observed in a video sequence, yet general enough to be easily adapted to a different user. In a typical video sequence, only the general shape of the hand can be extracted reliably and the exact appearance of the skin surface and detailed structure are not available due to sensor noise, complicated image composition, and low image resolution. Therefore, employing high LOD in this scenario is overfitting the model and prohibits real time processing. Furthermore, additional information could also mean a more complicated and cumbersome initialization for every user. In a natural HCI environment, the new user should be able to start using the system as quickly and naturally as possible.

2.4.1 Kinematical model

We model the kinematical structure of the hand to facilitate the analysis of hand motions. In our research, the skeleton of a hand is abstracted as a stick figure in which each finger is represented by a kinematical chain with base frame at the palm and each fingertip as the end-effector. Figure 2.5 shows such a stick figure of the palm-side view of a right hand. This kinematics model has roughly 27 DOF based on anatomical studies.

The motion of the finger flexions, extensions, abductions, and adductions determine the finger articulation. In this hand model, there are 21 DOF, i.e., 21 joint angle parameters, that specify the shape of the hand. Each of the four fingers contributes 4 DOF. The distal interphalangeal (DIP) and proximal interphalangeal (PIP) joints each have 1 DOF, and the metacarpo-phalangeal (MCP) joint has 2 DOF, due to flexion and abduction. The thumb has a different structure from the other four fingers and is associated with 5 DOF: one for the interphalangeal (IP) joint, two for the thumb MCP, and two for the trapeziometacarpal (TM) joints, due to flexion and abduction. The hand has six additional DOF from the rotational and translational motion of the palm, with

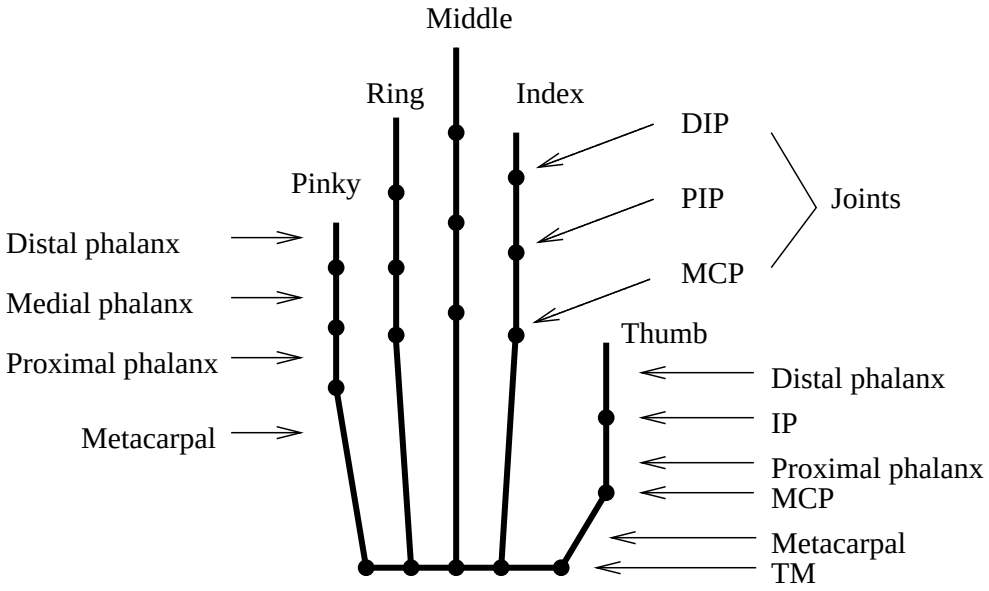


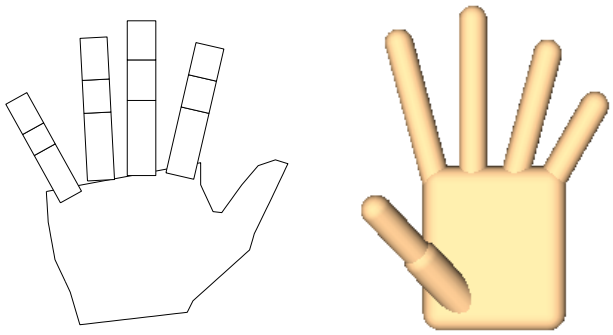
Figure 2.5 The kinematical model.

3 DOF each. When the finger lengths are given, the inverse kinematics is computed to recover the joint angles. In [28], a gradient-based method is given that derives the kinematical Jacobian.

2.4.2 Shape model

Different models have been reported to be implemented for tracking. One choice is the 2D patch model (Figure 2.6 (a)) which is easy to implement and sufficient for the task when most of the global motion occurs parallel to the image plane. However, it is not capable of handling large out-of-plane rotations [36]. On the other hand, a 3D model (Figure 2.6 (b)) is capable of handling arbitrary views of the hand, but it requires more computation to generate the corresponding projection when used in the model-based tracking [37]. For the class of 3D models, many choices have been proposed depending on the LOD desired. Previous work has employed models constructed using cubic B-spline [38], polygonal mesh [9, 29, 34], and quadric surfaces[37, 39, 40]. Though a more detailed model allows for better matching, it severely limits the generality. With more details encoded, longer time and more effort is also required when the tracking is performed for a new user. A more detailed model also increases the computation when a projection is generated. For instance, to determine whether a projected edge point generated from one finger is occluded by another finger, every segment of the other fingers has to be checked. An example of the pieces that are used

to model a hand is shown in Figure 2.7. The phalanx of each finger is modeled by a truncated cylinder and they are connected by spheres at the joint locations. The palm is constructed using two planar patches with cylinders and spheres enclosing the sides and corners. There are a total of 40 components that make up this model.



(a) (b)

Figure 2.6 (a) An example of a 2D patch model. (b) An example of a 3D quadric model.

2.5 Estimating the Hand Motion

To estimate the hand motion from video sequences, both the local finger articulation and global hand motion should be determined. Because of the high DOF involved, it is a very difficult task in addition to the problems and uncertainties introduced by image noise, self-occlusions, and complex background.

2.5.1 3D model-based approach

One approach to track the hand motion is the 3D model-based approach [2, 9, 23, 28, 38, 39, 40, 41, 42]. The idea is to compare image features between 3D hand model projections and real hand images. The shape of the 3D hand model is controlled by specifying the joint angles. The real hand state is recovered from the configurations that generate the best match. An illustration is shown in Figure 2.8 where instances of silhouette and edge of the projection is shown. With a well initialized hand model, this approach can generate a very accurate estimate. However, model-based tracking involves the estimation of parameters in high-dimensional space, which is 27 parameters

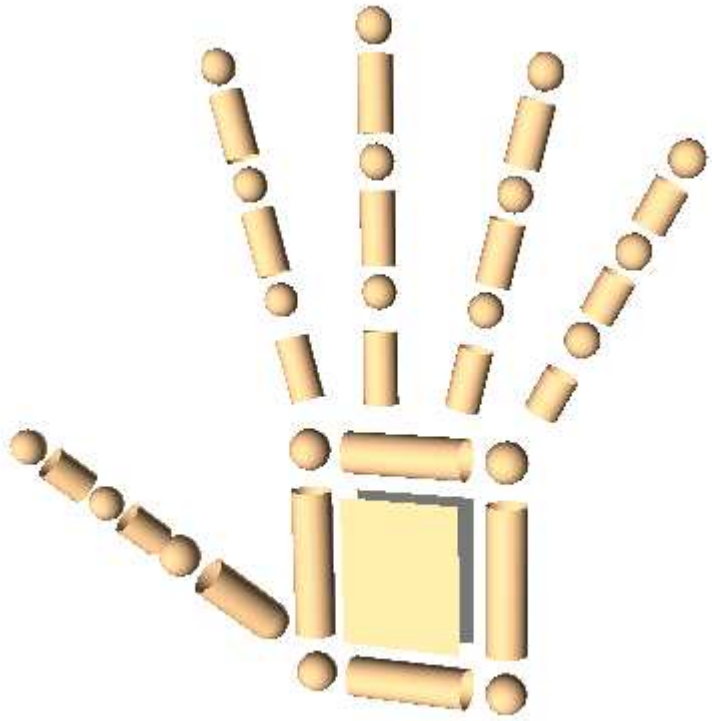


Figure 2.7 The composition of a 3D hand model.

(a) (b)

Figure 2.8 (a) An example of the 3D model and corresponding silhouette of the projections. (b) An example of the edge of the projections that fits a real hand image.

in the case of the hand. Wu and Huang [43] showed that the problem complexity can be reduced by a divide-and-conquer approach which estimates the global and local motion separately. Another problem is the generation of the projection image. Since the articulated object consists of many segments, the computations involved in determining edge pixel visibility could be very large. In

this dissertation, we will focus on algorithms for tracking the finger articulation, that are made feasible by considering the motion constraints. Most of the previous work on model-based hand tracking differs only in terms of the tracking algorithm and the model used. We will approach the more fundamental problem by modeling the feasible configuration space to reduce the search space.

2.5.2 Appearance-based approach

Another strategy for recovering the hand state is to use the appearance-based approach [22, 29, 30, 34, 37, 44, 45], which attempts to estimate the hand states directly from the image features. Contrary to the model-based approach, this method is most suitable for recognizing a finite set of gestures. For some applications, only a few distinct gesture classes will be sufficient to control the interaction, and an accurate hand motion tracking is not necessary. Since this approach is performed independently at each frame, it could be used to automatically initialize hand postures or recover the correct hand shape when model-based approach lost tracks.

The appearance-based approach basically involves a learning and classification problem. A nonlinear mapping is learned from a large number of training images. This approach can quickly estimate the hand configuration once the mapping is learned. However, it is difficult to determine the structure of the mapping function, the appropriate feature space, and the optimal training data set. Further, because it is a classification problem, gestures with similar pose and shape will be difficult to be differentiated.

CHAPTER 3

HAND MOTION CONSTRAINTS

3.1 Introduction

The difficulties of tracking the human hand arises mainly from the large DOF involved in the motion. As a result, estimating the correct hand motion and configuration parameters is equivalent to a search problem in the high-dimensional space. Other difficulties include the self-occlusions of different fingers, and clutters from the image background which introduce additional uncertainties for the estimation.

Hand and finger motions are constrained so that the hand cannot make arbitrary gestures. There are many examples of such constraints. For instance, fingers can only bend backward so much, and for many people, the little finger cannot be bent without bending the ring finger. The natural movements of human hands are implicitly restricted by such motion constraints.

Some motion constraints may have closed-form representations, which are often employed in the current research on animation and visual motion capturing [23, 38, 46]. However, many motion constraints are very difficult to express in closed forms. How to model such constraints still requires further investigation. To begin the constraints analysis, we will first summarize in Section 3.2 some of the constraints that have been adopted in previous work. We describe three types of motion constraints and explain how we are able to initially reduce the number of parameters required to represent the hand motion to 15 from 21. Then we describe our approach to learn the intrinsic representation of the human hand motion from real hand motion data in Section 3.3. To compactly model the feasible configuration space, a semiparametric representation is proposed in Section 3.4. In Section 3.5, we propose a more flexible model using a nonparametric representation.

3.2 Natural Constraints

Natural hand motion constraints can be roughly divided into three types. The first type of constraint, usually referred to as *static constraints*, defines the maximum ranges of finger motions constrained by hand anatomy. The second type of constraint is usually referred to as *dynamic constraints* in previous work. These constraints describe the motion correlations due to the limits imposed on joints during motion. The third type of constraint is applied in performing natural motion and has not yet been fully explored. Below we will describe each type in more detail.

3.2.1 Anatomical constraints

These constraints are the limits of the range of finger motions as a result of hand anatomy. Such constraints determine the boundaries of feasible states in the feasible configuration space. They are usually called *static constraints* in previous work. We will only consider the range of motion of each finger that can be achieved without applying external forces. With an applied external force, such as bending the fingers backward using the other hand, a hand is able to achieve greater range of motion. However, this is outside the scope of current study. Anatomical constraints are usually represented by the following inequalities:

$$0^o \leq \theta_{MCP-F} \leq 90^o$$

$$0^o \leq \theta_{PIP} \leq 110^o$$

$$0^o \leq \theta_{DIP} \leq 90^o, \text{and}$$

$$-15^o \leq \theta_{MCP-AA} \leq 15^o \tag{3.1}$$

The maximum and minimum values of the joint angles are different for different people, but they generally fall within ranges of Equation (3.1). Another commonly applied anatomical constraint states that the middle finger displays little abduction/adduction motion. The following approximation is made for middle finger:

$$\theta_{MCP-AA} = 0 \tag{3.2}$$

This will subtract 1 DOF from the 21 DOF model.

Similarly, the TM joint also displays limited abduction motion and will be approximated by 0 as well:

$$\theta_{TM-AA} = 0 \tag{3.3}$$

As a result, the thumb motion will be characterized by four parameters instead of five. Finally, the index, middle, ring, and little fingers are planar manipulators, i.e., the DIP, PIP, and MCP joints of each finger move in one plane since DIP and PIP joints only have 1 DOF for flexion.

3.2.2 Motion correlations

These constraints are the limits imposed on joints during finger motions. They depict the feasible trajectories of state transitions in the configuration space. Motion correlation constraints are often referred to as *dynamic constraints* and can be subdivided into intrafinger and interfinger constraints. The intrafinger constraints are the constraints between joints of the same finger. A commonly applied intrafinger constraint [23, 38] based on hand anatomy states that for the index, middle, ring and little fingers, in order to bend the DIP joints, the PIP joints must also be bent. Their relations can be approximated as follows:

$$\theta_{DIP} = \frac{2}{3}\theta_{PIP} \tag{3.4}$$

The relation shown in Equation (3.4) is only an approximation of the finger motion since people are able to bend the PIP joints while keeping DIP joints still. Furthermore, as θ_{DIP} increases beyond roughly 60^o, this relationship ceases to hold. However, this range is sufficient for posture estimation. A 22-sensor glove will be needed in order to learn the more accurate relationship between these two joints.

By combining Equations. (3.1)-(3.4), we are able to reduce the 21 DOF model to one that is approximated by 15 DOF. Experiments in previous work have shown that hand postures can be estimated using these constraints without severe degradation in performance.

Another weak constraint that is also sometimes used in previous work relates DIP and MCP

Table 3.1 Values of σ and σ' for each finger.

Finger	σ	σ'
Index	-54^o	25^o
Middle	-45^o	20^o
Ring	-44^o	48^o

motion of the four fingers as follows:

$$\theta_{MCP} = k\theta_{PIP}, k \geq 0 \tag{3.5}$$

However, such a constraint is not very accurate and often requires the adjustment of k, which is usually set to 1/2 to begin with. As a result, this constraint is not sufficient to reduce the overall DOF of the hand model. In our approach, we will learn such constraints through empirical data. Interfinger constraints are those imposed on joints between adjacent fingers. For instance, when an index MCP joint is bent, the middle MCP joint is forced to bend as well. Lee and Kunii [23] have performed measurements on several people and obtained a set of inequalities that approximates the limits of adjacent MCP joints. Chang and Tsai [46] give a simpler representation based on results from [23]:

$$\theta_{L,MCP-F} + \sigma_R \leq \theta_{R,MCP-F} \leq \theta_{L,MCP-F} + \sigma'_R$$

$$\theta_{R,MCP-F} + \sigma_M \leq \theta_{M,MCP-F} \leq \theta_{R,MCP-F} + \sigma'_M$$

$$\theta_{M,MCP-F} + \sigma_I \leq \theta_{I,MCP-F} \leq \theta_{M,MCP-F} + \sigma'_I \tag{3.6}$$

where the subscripts I, M, R, and P denote the index, middle, ring, and little fingers, respectively. The values of σ and σ' are listed in Table 3.1. Note that their representations are no longer as simple as those in Equations (3.4) and (3.5).

There are a few more constraints regarding the thumb and finger abduction angles, and details can be found in [23, 38] on how these constraints were used to facilitate tracking. However, more motion correlation constraints exist that cannot be explicitly represented by equations.

3.2.3 Natural motions

The third class of constraints is imposed by the naturalness of hand motions and are subtler to detect and quantify. Almost nothing has been done to account for these constraints in simulating a natural hand motion. The other two classes of constraints are the results of hand bone structure, ligaments, tendons, and muscles acting upon each other that produce the feasible configuration. In contrast, the natural motion constraints differ from the other two types of constraints in that they have nothing to do with limitations imposed by hand anatomy, but rather are results of common and natural movements. For instance, the most natural way for every person to open a hand from a closed fist would be to extend all the fingers at the same time instead of curling one finger at a time. Another instance is when we curl multiple fingers together; we tend to move them together rather than one by one. Even though the naturalness of hand motions differs from person to person, it is broadly similar for everybody. This type of constraint also cannot be explicitly represented by equations in terms of joint angles. Some guesses, such as energy minimization, have been made regarding the explanations of such behaviors, but further investigation is still required.

3.3 Learning Constraints in High-Dimensional Space

It is generally difficult to explicitly represent all the constraints of natural hand motions in closed forms. However, they can be learned from a large and representative set of training samples; therefore we propose to learn the constraints by modelling a compact representation of the feasible configuration space.

3.3.1 Data collection

The articulate local hand motions, e.g., finger motions, can be represented by a set of joint angles $\Theta = (\theta_1, \ldots, \theta_n)$, where n is the number of joint angle measurements. In order to capture the hand states and motion, a glove-based device is a natural choice since it has sensors attached over hand joints and directly measures each joint angle. In our study, we employ a right-handed 18-sensor CyberGlove developed by Virtual Technologies Inc. Although the goal of vision-based hand motion analysis is to be able to recognize hand configurations without the use of attached

external devices, a glove-based device will help in collecting ground truth data, which enable the modeling and learning process in visual analysis.

The glove we are using [47] has four sensors for the thumb, a MCP and a PIP sensor for the MCP(θ_{MCP-F}) and PIP(θ_{PIP}) flexion angles for each of the four fingers, and three more abduction sensors for the abduction/adduction angles (θ_{MCP_AA}) between these four fingers. There are total of 15 sensor readings of the finger joint angles (Figure 3.1); therefore, we are able to characterize the local finger motion by 15 parameters. The remaining three sensors are used to model the palm arch and to measure wrist pitch and yaw angles. However, in our current research, we are assuming that the palm is a rigid object and that we are only interested in the local finger motions; therefore, these three sensors for the palm and wrist angle measurements are not being used. The glove can be calibrated to measure the joint angles to within 5^o, which is acceptable for gesture recognition. Finger postures that differ by 5^o would still appear to be the same posture. Finally, the glove is capable of recording approximately 100 sets of 18-sensor records per second, which will enable us to accurately record the motions with a dense data set. We have collected a set of more than 30 000 joint angle measurements by performing various natural finger motions.

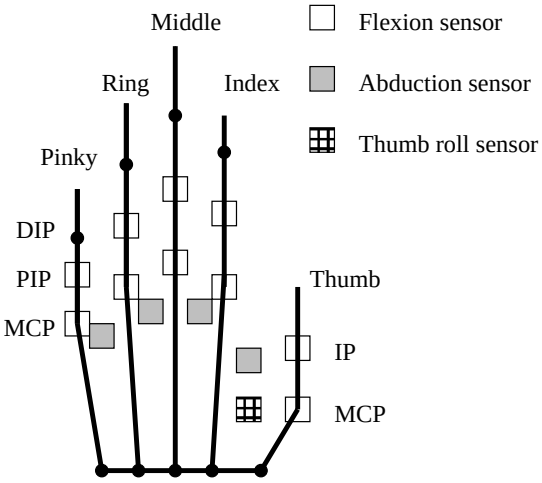


Figure 3.1 The sensor locations.

3.3.2 Initial data analysis

It would be ideal to determine a compact parametric representation of the state space, which would greatly enhance the tracking performance by easily avoiding the infeasible region and focus

on the more probable states. Due to the complex motion and the high DOF, it is not possible to come up with a simple parametric representation. However, we can still learn a more compact representation by eliminating the obvious redundancies and observing motion trajectories in Ξ.

Instead of searching in the 20-dimensional space, we would like to use various constraints to reduce the dimensionality of the joint angle space and find a smaller feasible space, which we will call the configuration space Ξ. Several commonly known constraints due to the anatomy of the hand can be used to initially reduce the dimensionality to roughly 15. To further reduce the dimensionality, we have collected more than 30 000 joint angle measurements from various hand motions using CyberGlove. Then PCA is applied to eliminate the redundancy. We can project $\Theta \in \mathcal{R}^{20}$ into a 7-dimensional subspace while maintaining 95% of the information. Therefore, the configuration space Ξ is defined in $\mathcal{R}^7$.

3.3.3 Related work

Learning the true manifold structure in a high-dimensional space is a challenging problem [48]. Previous work such as ISOMAP [49], locally linear embedding [50], and multidimensional scaling [51] generally represents the manifold using piecewise linear assumption. A clustering algorithm is first applied to identify locally similar patches, which is then approximated by a linear manifold in a lower dimensional space determined by principal components analysis (PCA) or other dimensionality reduction techniques. For the case of object recognition, an affinity measure is defined between manifolds that would maximize the class separability [52, 53, 54]. For tracking, a transition probability table is constructed to model the motion dynamics [55, 56]. It is generally a difficult problem in determining a suitable clustering algorithm and to construct an accurate manifold assumption.

3.4 Semiparametric Representation

3.4.1 Basis states

After the initial dimensionality reduction procedure, we continue to investigate the structure of the embedded manifold that characterize the hand motion in Ξ. Let us define 28 basis configurations as follows. For each basis state $\mathbf{b}_i$, each finger is either fully extended or fully curled. A subset of

the basis states is shown in Figure 3.2(a). The intuition is that each finger moves between these two extremal states, and these two configurations can be used to help identify the boundary of feasible motions. Empirical study shows that feasible hand configurations are roughly bounded by the convex hull defined by these basis states. Besides collecting static hand hand posture data, we also collected sequences of configurations during hand motion in order to study the temporal pattern. We are especially interested in the motion trajectories between the basis states.

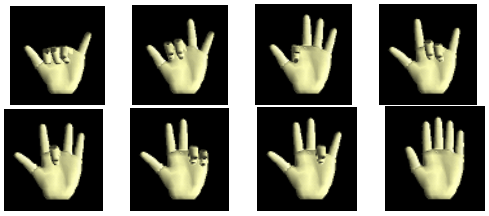
3.4.2 Observations in lower dimensional space

Since it is impossible for us to visualize data in high dimensional space such as $\mathcal{R}^7$, we take a subset of the basis states and the corresponding hand motion trajectories, and performed the same analysis as described earlier in order to visualize the result. A lower-dimensional illustration is shown in Figure 3.2(b), in which each point represents a real hand configuration in Ξ. In this example, the movements involving index, middle, and ring fingers are chosen. Since only three fingers are moving, we could reduce the dimensionality down to three without much degradation in data representation. The corresponding basis states are also shown in Figure 3.2(a), and they lie roughly at the corner of the cube whose edge is formed by a collection of the motion trajectories between the basis states. Though the interior of the cube is shown to be empty in this picture due to staged performance, if other random motion involving the three fingers are included, they will appear in the inside of the cube. This verifies the conjecture that the convex hull defined by the basis states roughly covers the entire range of feasible configuration space.

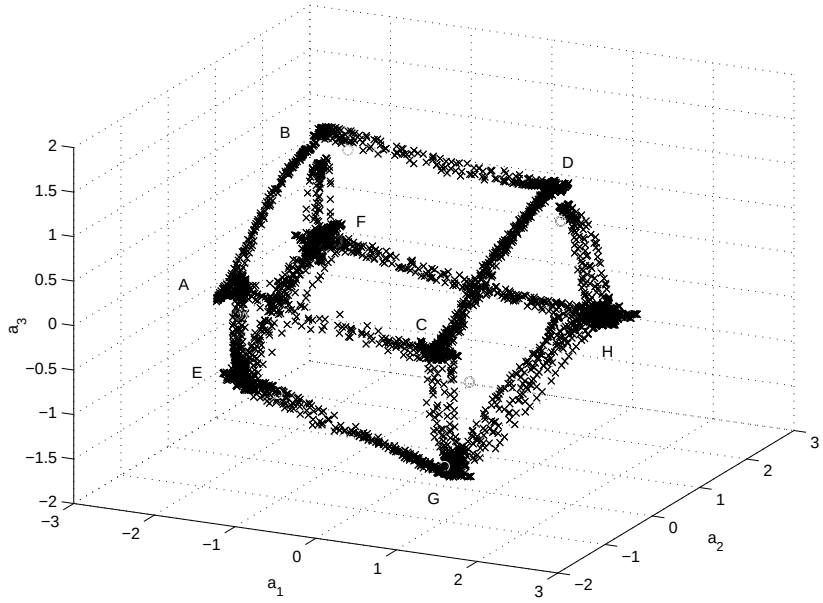
3.4.3 Semiparametric model

Our observations of the motion trajectories between basis states in Ξ show that they are roughly linear and that natural hand articulation can be characterized by these linear manifold $\mathcal{L}_{ij}$ spanned by $\mathbf{b}_i$ and $\mathbf{b}_j$, with $i \neq j$:

$$\Xi \approx \bigcup_{i,j} \mathcal{L}_{ij}, \text{ where } \mathcal{L}_{ij} = span(\mathbf{b}_i, \mathbf{b}_j) \tag{3.7}$$



(a) a subset of basis configurations



(b) linear manifolds in the configuration space

Figure 3.2 Hand articulation in the configuration space, which is characterized by a set of basis configurations and linear manifolds.

3.5 Nonparametric Representation

Although the previous representation gives a compact description of the motion trajectories, it is generally difficult to obtain a correct manifold parametrization. Furthermore, for more complicated finger motions, the linear assumption might not hold. In addition, PCA can only model the global characteristics. It is not capable of identifying the local manifold structures and their true dimensionality.

We propose an alternative that adopts a nonparametric representation and models the entire feasible space directly from the set of N_C collected Cyberglove data. We denote this discrete space representation by Ψ to differentiate from the semiparametric representation Ξ. The entire feasible space Ψ is defined by the set of $\theta_i, i = 1 \ldots N_C$, where each $\theta \in \mathcal{R}^{20}$ represents one sampled configuration. Then a Kd-tree structure is constructed so that given any point $\theta \in \mathcal{R}^{20}$, we can quickly

locate an approximated nearest neighbor $\theta' \in \Psi$. One of the benefits of using this representation is that no learning is required to find a closed form of the manifold representation of Ψ. Therefore, this approach avoids the error induced by incorrect approximations of the representation. Furthermore, the motion constraints are automatically embedded in this discrete model. The model can be better refined when more samples are collected. This advantage is gained as a trade off for the cost of the computer memory, which is inexpensive nowadays.

CHAPTER 4

TRACKING HUMAN HAND MOTION

4.1 Introduction

In Chapter 3 we described two motion constraint representations. The first one uses a semi-parametric representation to model the motion trajectories between different basis states. The second one uses a discrete representation of the feasible configuration space modeled directly from collected motion data. In this chapter we first describe the general tracking framework using the sequential Monte Carlo (SMC) approach, and then present a tracking algorithm for the semiparametric constraint representation in the subsequent sections. In Chapter 5 we will give a modified SMC tracker that is suitable for tracking in the discrete feasible configuration space.

4.2 Sequential Monte Carlo

4.2.1 Formulation

To formulate the problem of visual tracking, define $\mathbf{x}_t$ to be the target state, and $\mathbf{z}_t$ the image observations at time t. The state $\mathbf{x}_t$ is modeled as a Markov process with transition density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The image observations $\mathbf{z}_t$ are assumed to be conditionally independent given the process $\mathbf{x}_t, t \geq 0$. In practical applications, $\mathbf{x}_t$ could denote any object state such as an articulate model configuration, a set of body joint angles or motion parameters, or a shape configuration, and $\mathbf{z}_t$ corresponds to any extracted image features.

The objective is to estimate the true state $\mathbf{x}_t$ given the set of available measurements $\mathbf{z}_{0:t}$ from time 0 to t. From the Bayes formulation, we obtain the following recursive relation for the posterior

distribution [57]:

$$p(\mathbf{x}_t|\mathbf{z}_{0:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{0:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{0:t-1})} \tag{4.1}$$

$$p(\mathbf{z}_t|\mathbf{z}_{0:t-1}) = \int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{0:t-1})d\mathbf{x}_t \tag{4.2}$$

In this recursive updating rule, the prior $p(\mathbf{x}_t|\mathbf{z}_{0:t-1})$ is obtained through a prediction stage via the Chapman-Kolmogorov equation:

$$p(\mathbf{x}_t|\mathbf{z}_{0:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-1})d\mathbf{x}_{t-1} \tag{4.3}$$

assuming that the initial probability density function (pdf) $p(\mathbf{x}_0|\mathbf{z}_0)$ is known.

This recursive propagation of the pdf cannot be determined analytically except for a few restricted cases such as the Kalman Filter. It serves mainly as a conceptual tool for solving general tracking problems. One popular approach to implement the recursive filter (Equation (4.1)) is the extended Kalman filter (EKF) [58]. This linearization technique could be applicable to nonlinear models with additive Gaussian noise. Another technique, known as the Unscented Kalman Filter (UKF), uses the unscented transform in an EKF framework [59, 60]. Using UKF, $p(\mathbf{x}_t|\mathbf{z}_{0:t})$ is also approximated by a Gaussian model. A set of points are deterministically selected from this approximation and propagated through the nonlinearity filter. The Gaussian approximation parameters are then re-estimated. The main limitation with these approaches is that $p(\mathbf{x}_t|\mathbf{z}_{0:t})$ is always approximated to be Gaussian and such a description will not be sufficient for non-Gaussian density. In many real applications, such as visual tracking in noisy and complex backgrounds, multimodal densities are not uncommon.

One practical implementation of the recursive framework depicted by Equation (4.1) is the SMC algorithm, also known variously as b*ootstrap filtering* [61], the CONDENSATION algorithm [62, 63], and *particle filtering* [64]. The key idea is to represent the posterior distribution by a set of N random samples $s^i, i = 1 \ldots N$ with associated weights π^i and to determine state estimates based on $\{s^i, \pi^i\}$.

4.2.2 Exponential increase in computational complexity

One problem with the particle filters is the exponential increase in the number of particles to represent the pdf as the number of the dimensions increases. A common solution to reduce this computation complexity is by identifying a more compact subspace manifold representation of the pdf or equivalently, by finding a importance function that is easier to sample from [65]. Thus, instead of sampling from $p(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-1})$, the particles are sampled from another auxiliary distribution $q(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-1})$. After imposing the dynamical model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, the particles now approximates the distribution $q(\mathbf{x}_t|\mathbf{z}_{0:t-1})$ and the weights for $p(\mathbf{x}_t|\mathbf{z}_{0:t})$ are obtained from a compensated term:

$$\pi_t^i = \frac{p(\mathbf{x}_t^i|\mathbf{z}_{0:t-1})}{q(\mathbf{x}_t^i|\mathbf{z}_{0:t-1})} p(\mathbf{z}_t|\mathbf{x}_t^i) \tag{4.4}$$

In Section 4.3 we present an algorithm that improves the tracker performance by using an auxiliary sampling function with applications in tracking finger articulations.

4.2.3 Degeneracy problem

Another problem with the particle filter is the *degeneracy phenomenon*, which occurs when most of the samples have negligible weights. It has been shown that this is impossible to avoid as the variance of the importance weights can only increase over time [66]. As a result, a large computation effort is wasted in maintaining these particles. In [67], a measure of the degeneracy is given by computing the effective sample size N_{eff}:

$$N_{eff} = \frac{N_s}{1 + \mathrm{Var}(w_t^{*i})} \tag{4.5}$$

where $w_t^{*i} = p(\mathbf{x}_t^i|\mathbf{z}_{0:t})/q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{z}_t)$ is referred to as the *true weight*. However, Equation (4.5) cannot be evaluated exactly, and an estimate $\widehat{N_{eff}}$ is computed instead:

$$\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{N_s} (w_t^i)^2} \tag{4.6}$$

where w_t^i is the normalized sample weight. We notice that $N_{eff} \leq N_s$, and severe degeneracy is observed when N_{eff} is small.

To reduce the effects of degeneracy, one popular algorithm is to include a resampling stage when significant degeneracy is observed. Examples include stratified sampling, residual sampling [67], and systematic resampling [68]. The resampling has the advantage that the importance weights are easily evaluated and the importance density can be easily sampled.

However, even with resampling, the prediction stage combined with the use of importance sampling does not guarantee the newly generated set of samples will have significant weights. This leads us to consider the use of descent algorithms that drives the samples toward the peak of the pdf. In Chapter 5, we present another tracker which introduces an additional stage of descent search embedded in the modified particle filter.

4.3 SMC Tracking with Importance Sampling

Because the finger articulation involves a high DOF, any SMC algorithm will require a large number of samples for representing the density propagation, and an intensive computation will be unavoidable. Fortunately, we may reduce the complexity by making use of the finger motion constraints as an outside prior for the importance sampling technique. One suitable choice of the constraint is the semiparametric constraint representation described in Section 3.4. Let $f_t(\mathbf{x}_t^{(n)}) = p(\mathbf{x}_t = \mathbf{x}_t^{(n)}|\mathbf{z}_{0:t-1})$ be the tracking prior. To approximate the posterior $p(\mathbf{x}_t|\mathbf{z})$, we draw random samples from another distribution $g_t(\mathbf{x}_t)$, instead of the prior density $f_t(\mathbf{x}_t)$. Below we will give a brief description of this method. The details can be found in [42].

For natural hand motion, each hand configuration $\mathbf{x}$ should be either around a basis state $\mathbf{b}_k, k = 1, \ldots, M$, or on the manifold $\mathcal{L}_{ij}$, where $i \neq j, i, j = 1, \ldots, M$. Suppose at time frame t, the hand configuration is $\mathbf{x}_t$, and it is not near any basis states. We find the projection $\bar{\mathbf{x}}_t$ of $\mathbf{x}_t$ onto the nearest manifold $\mathcal{L}_{ij}^*$, and obtain

$$s_t = 1 - \frac{(\mathbf{x}_t - \mathbf{b}_i)^T(\mathbf{b}_j - \mathbf{b}_i)}{||(\mathbf{b}_j - \mathbf{b}_i)||}$$

Then, random samples are drawn from the manifold $\mathcal{L}_{ij}$ according to the density p_{ij}, i.e.,

$$s_{t+1}^{(n)} \sim p_{ij} = N(s_t, \sigma) \tag{4.7}$$

$$\widetilde{\mathbf{x}}_{t+1}^{(n)} = s_{t+1}^{(n)}\mathbf{b}_i + (1 - s_{t+1}^{(n)})\mathbf{b}_j \tag{4.8}$$

Next, perform random walk on $\widetilde{\mathbf{x}}_{t+1}^{(n)}$ to obtain hypothesis $\mathbf{x}_{t+1}^{(n)}$, i.e.,

$$\mathbf{x}_{t+1}^{(n)} \sim N(\widetilde{\mathbf{x}}_{t+1}^{(n)}, \Sigma_{t+1}) \tag{4.9}$$

Define the importance function to be $g_{t+1}(\mathbf{x}_{t+1}^{(n)}) = p(s_{t+1}^{(n)}|s_t)p(\mathbf{x}_{t+1}^{(n)}|\widetilde{\mathbf{x}}_{t+1}^{(n)})$, and we have

$$\begin{aligned} g_{t+1}(\mathbf{x}_{t+1}^{(n)}) &\sim \frac{1}{\sigma|\Sigma|^{1/2}} \exp\{-\frac{(s_{t+1}^{(n)} - s_t)^2}{2\sigma^2} \\ &- \frac{1}{2}(\mathbf{x}_{t+1}^{(n)} - \widetilde{\mathbf{x}}_{t+1}^{(n)})\Sigma^{-1}(\mathbf{x}_{t+1}^{(n)} - \widetilde{\mathbf{x}}_{t+1}^{(n)})\} \end{aligned}$$

Finally, the weights must be properly compensated:

$$\pi_{t+1}^{(n)} = \frac{f_{t+1}(\mathbf{x}_{t+1}^{(n)})}{g_{t+1}(\mathbf{x}_{t+1}^{(n)})} p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1} = \mathbf{x}_{t+1}^{(n)}) \tag{4.10}$$

If the previous hand configuration is at one of the basis configurations, say $\mathbf{x}_t = \mathbf{b}_k$, it is reasonable to assume that it selects any one of the manifolds of $\{\mathcal{L}_{kj}, j = 1, \ldots, M\}$ with the same probability. Consequently, random samples are drawn from a mixture density p_k. The process of generating hypotheses is illustrated in Figure 4.1 The results of using this tracking algorithm is shown in Section 4.6.

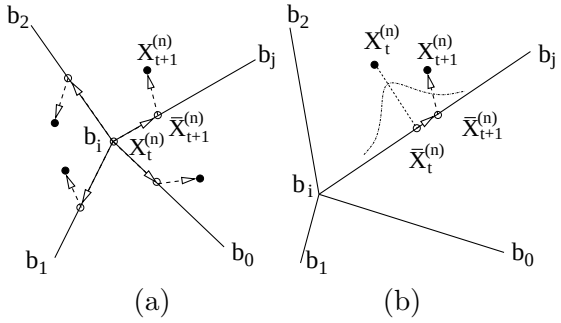


Figure 4.1 An illustration of hypotheses generation. (a) $\mathbf{x}_t$ is nearby a basis state and (b) $\mathbf{x}_t$ takes on a manifold.

4.4 Divide and Conquer

In [43], Wu and Huang propose to simultaneously estimate both global and local hand motions by a divide-and-conquer approach. Rather than estimating the optimal state parameters altogether, the global motion and local motion parameters are estimated separately and combined in an iterative manner. Different algorithms can be used to independently estimate each set of parameters, such as the algorithm based on genetic algorithm (GA) used in [43]. For the tracker presented in Section 4.3, we employed the iterative closed point algorithm [36]. The algorithm finds point correspondences between model projections and edge observation which helps to recover the global motion parameters. These global and local motion estimation steps are repeated until they converge to a local stationary point that minimizes the discrepancies between the image observation and model projection. Our experiments have shown that this approach reduces the computational complexity while effectively estimates the hand motion.

4.5 Likelihood Evaluation

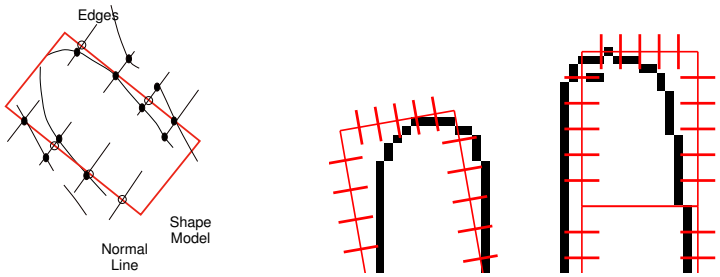
We employ edge and silhouette observations to measure the likelihood of hypothesis as in [36] (see Figure 4.2). The hand model is first projected onto the image plane to obtain the edge points that define the projected shape. If K projected model samples are generated, edge detection is performed on the points along the normal of this sample. Assuming that M edge points $\{z_m, m = 1, \ldots, M\}$ are observed, and the clutter is a Poisson process with density λ, then,

$$p_k^e(\mathbf{z}|x_k) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma_e q \lambda} \sum_{m=1}^{M} \exp -\frac{(z_m - x_k)^2}{2\sigma_e^2} \tag{4.11}$$

We noticed that edge points alone could not provide a good likelihood estimation; therefore, we also consider the silhouette measurement. The segmented foreground pixels are XORed with the projected silhouette image, and the likelihood is computed as $p^s \propto \exp -\frac{(A_I - A_M)^2}{2\sigma_s^2}$. Since a well-matched projection contributes lower cost, we define the objective function at time t to be the

negative of the likelihood function:

$$f(\mathbf{x}, \mathbf{z}) = -p(\mathbf{z}|\mathbf{x}) \propto -p^s \prod_{k=1}^{K} p_k^e \tag{4.12}$$



(a) (b)

Figure 4.2 Shape measurements.

4.6 Experiments

The algorithm proposed in this chapter is tested on synthetic hand motion in order to obtain a quantified analysis for evaluation. We also applied the algorithm to track real hand motion.

4.6.1 Simulation

It is generally difficult to obtain the ground truth of hand motions from a real video sequence. Therefore, we have produced a synthetic sequence of 200 frames containing typical hand movements. This synthetic sequence will facilitate the quantitative evaluations of our algorithm. Some examples of the synthetic motion is shown in Figure 4.3.

(a) (b)

Figure 4.3 An example of the results from tracking the synthetic hand motion sequences. (a) A synthetic hand image and (b) the image with model projection aligned.

Some of the motion parameters are shown in Figure 4.4 for comparison. The solid curves are

our estimates and the dash curves are the ground truth. The figure plot the x translation with average error of 3.98 pixels, the rotation with average error of 3.42^o, the PIP joint of the index finger with average error of 8.46^o, the MCP flexion of the middle finger with average error of 4.96^o, the PIP joint of the ring finger with average error of 5.79^o, and the MCP abduction of the ring finger with average error of 1.52^o.

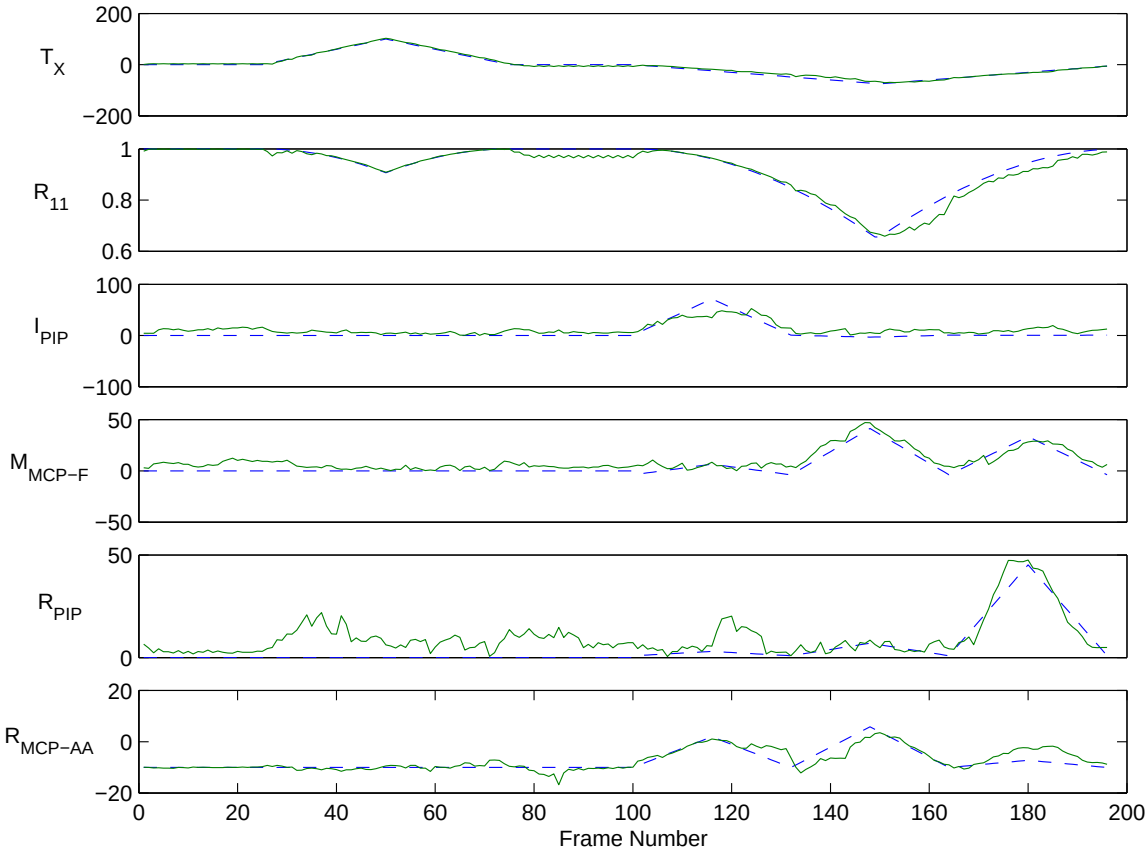
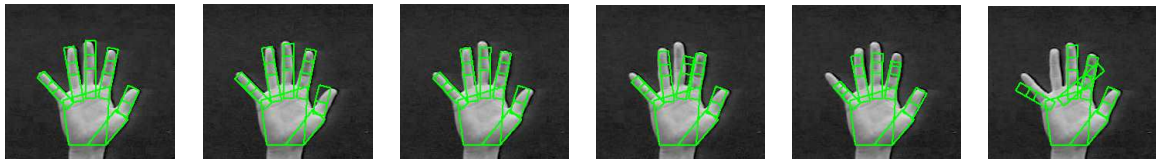


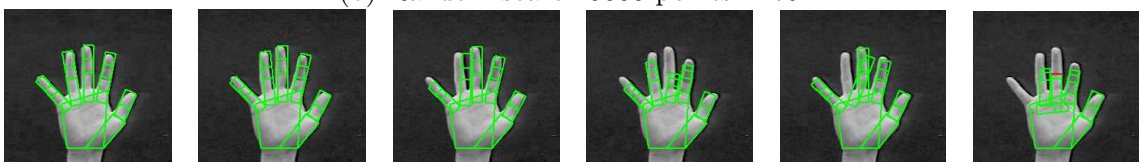
Figure 4.4 The comparison of our results and the ground truth for tracking a synthetic hand motion sequence. The blue dash curves are the ground truth, and the solid green curves are our estimates.

4.6.2 Real sequences

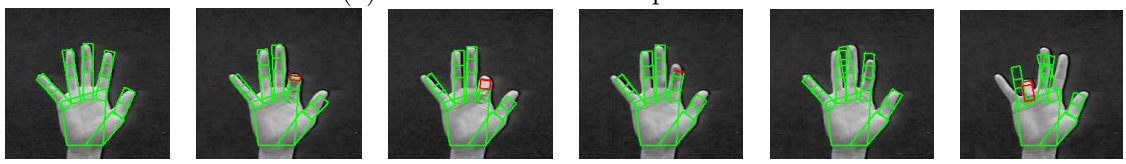
We also perform experiments on real hand motion sequence for local motion capturing(Figure 4.5). The first set of experiments focuses on the finger articulation tracking and comparisons with other algorithms. The results show that our algorithm outperforms others with a significantly less number of particles used. In the second set of experiments (Figure 4.6), global motion is included in



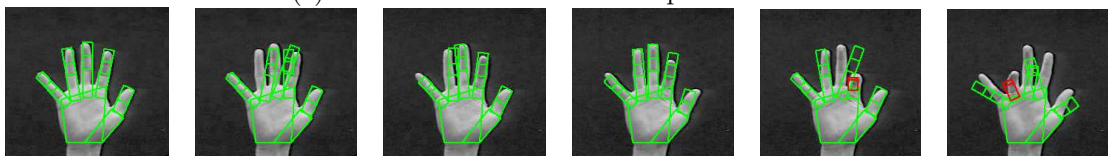
(b) Random search 5000 points in $\mathcal{R}^{20}$.



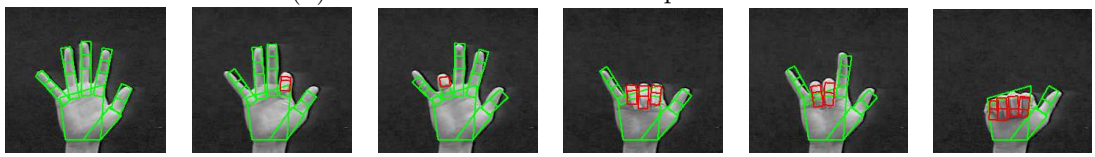
(a) Random search 5000 points in $\mathcal{R}^{7}$.



(c) CONDENSATION with 5000 points in $\mathcal{R}^{20}$.



(d) CONDENSATION with 3000 points in $\mathcal{R}^{7}$.

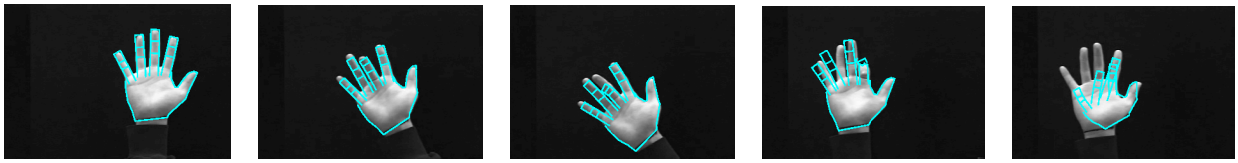


(e) Our approach with 100 samples.

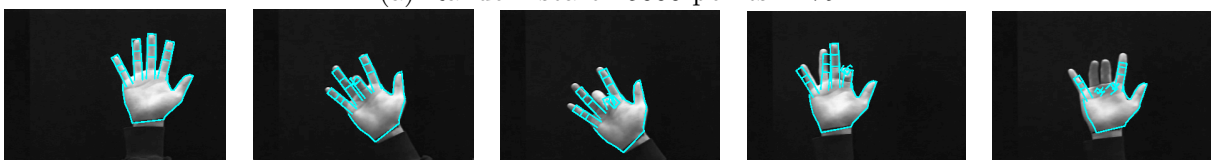
Figure 4.5 Comparisons of different methods on real hand sequences. In this sequence only finger articulation is considered. Our method is more accurate than the other two methods.

the testing sequences and comparisons are also made with other algorithms.

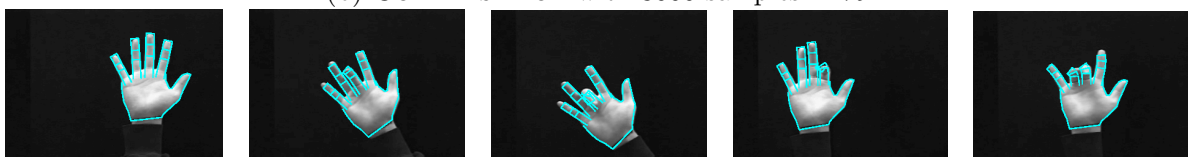
In both experiments, we first compare with the random search scheme in the $\mathcal{R}^7$ space with 5000 random samples. Since it makes use of no constraints, the performance is poor for local motion estimation and also degrades the global pose estimation. The second scheme uses CONDENSATION with 3000 samples in $\mathcal{R}^7$. It performs better than the first method, but it is still not robust enough. The third scheme is the proposed method, and it works accurately and robustly. The articulation model makes the computation more efficient and the local motion estimation enhances the accuracy of hand pose determination. In these experiments, a 2D patch model in the 3D space is used to generate the model projection. The implementation is done in MATLAB which takes roughly 1 min to process each frame. Although the hand state estimate is accurate, and significant computation is



(a) Random search 5000 points in $\mathcal{R}^7$.



(b) CONDENSATION with 3000 samples in $\mathcal{R}^7$.



(c) Our approach with 100 samples.

Figure 4.6 Comparisons of different methods on real hand sequences. In this sequence the hand undergoes both local finger articulations and global rotations and translations. Our method is more accurate than the other two methods.

reduced, the processing time is still too long. Also, the patch model can not handle global motions with large out-of-plane rotation.

CHAPTER 5

STOCHASTIC NELDER-MEAD SIMPLEX SEARCH

5.1 Introduction

The particle filter is a top-down approach that has the advantage of providing multiple hypotheses which help to resolve the ambiguities encountered in tracking. But depending on the resampling scheme and the importance function chosen, many of the generated particles might become insignificant. On the other hand, the bottom-up approach performs a sequential search for the current estimate $\mathbf{x}^*$ in the vicinity of the previous estimate $\mathbf{x}^*_{t-1}$. Algorithms in this class are termed direct search methods and are described in Section 5.2. In Section 5.3 the Nelder-Mead (NM) simplex search is introduced which works for arbitrary objective function and state space. We describe in Section 5.4 how it can be adapted to work in the discrete feasible configuration space. Then a new algorithm that combines the strength of both top-down and bottom-up approaches is presented in Section 5.5. Several experiments are performed for the algorithm proposed and results are shown in Section 5.6.

5.2 Direct Search Method

Given the object state $\mathbf{x}^*_{t-1}$ at time $t-1$, the goal is to identify $\mathbf{x}^*_t$ which minimizes certain objective function $f(\mathbf{x})$ through an iterative descent method. When the objective function is

differentiable, a recursive procedure to estimate $\mathbf{x}^*$ proceeds as follows:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k D^k \nabla f(\mathbf{x}^k) \tag{5.1}$$

where α is a positive scalar that determines the step size, D is a positive definite symmetric matrix, and $\nabla f(x)$ is the gradient of the objective function. $Df(x)$ together determines the descent direction. With appropriate assumptions, the iteration will converge to a local minimum. However, in many real applications, the gradients are not readily accessible or are computationally expensive to evaluate. In many cases, the objective function itself is nonlinear and the closed form is difficult to obtain; therefore, we must consider the nongradient descent algorithms. For instance, in visual tracking, the objective function is often defined in terms of the matching function, such as the negative of the likelihood function $-p(\mathbf{z}|\mathbf{x})$ (Section 4.5). Although we can always compute a matching score given any model configuration and the current input image, the function itself is highly nonlinear.

5.3 Nelder-Mead Simplex Search

Having only the access to the function values $f(\mathbf{x})$, we propose to implement the NM search method [69, 70], which belongs to the class of direct search methods. This algorithm is ideal for searching in the discrete space Ψ since the objective function defined over this space is also discontinuous. The NM method attempts to minimize a scalar-value nonlinear function of n real variables using only function values, without any knowledge of the gradient information, whether explicit or implicit. The NM method maintains at each iteration a nondegenerate simplex S, which is a geometric object defined by a convex hull of $n+1$ points $\{\mathbf{x}_0, \ldots, \mathbf{x}_n\}$ in $\mathcal{R}^n$.

Through a sequence of geometric transformations, the initial simplex moves, expands, or contracts towards the minimum. At each step, the worst vertex with highest cost $\mathbf{x}_{max} = \arg\max_{\mathbf{x}\in S} f(\mathbf{x})$ is replaced by a new vertex $\mathbf{x}_{new}$ which has a smaller function value. The basic form of NM method is outlined in Figure 5.1.

The iteration for NM-method typically terminates when a maximum number of iteration is

Iteration of the Simplex Method
Compute the centroid

$$\hat{\mathbf{x}} = \frac{1}{n}(\sum_{i=0}^{n} \mathbf{x}_i - \mathbf{x}_{max}) \tag{5.2}$$

Step 1: (Reflection Step) Compute

$$\mathbf{x}_r = (1+\alpha)\hat{\mathbf{x}} - \alpha\mathbf{x}_{max} \tag{5.3}$$

Then compute x_{new} according to the following three cases:

1. ($\mathbf{x}_r$ has min cost) If $f(\mathbf{x}_{min}) > f(\mathbf{x}_r)$, go to Step 2.
2. ($\mathbf{x}_r$ has intermediate cost) If $\max\{f(\mathbf{x}^i)|\mathbf{x}^i \neq \mathbf{x}_{max}\} > f(\mathbf{x}_r) \geq f(\mathbf{x}_{min})$, go to Step 3.
3. ($\mathbf{x}_r$ has max cost) If $f(\mathbf{x}_r) \geq \max\{f(\mathbf{x}^i)|\mathbf{x}^i \neq \mathbf{x}_{max}\}$, go to Step 4.

Step 2: (Attempt Expansion) Compute

$$\mathbf{x}_{new} = \gamma\mathbf{x}_r + (1-\gamma)\hat{\mathbf{x}}. \tag{5.4}$$

Define

$$\mathbf{x}_{new} = \begin{cases} \mathbf{x}_{exp} & \text{if } f(\mathbf{x}_{exp}) \leq f(\mathbf{x}_r), \\ \mathbf{x}_r & \text{otherwise,} \end{cases} \tag{5.5}$$

and form the new simplex by replacing the vertex $\mathbf{x}_{max}$ with $\mathbf{x}_{new}$.

Step 3: (Use Reflection) Define $\mathbf{x}_{new} = \mathbf{x}_r$. and form the new simplex by replacing the vertex $\mathbf{x}_{max}$ with $\mathbf{x}_{new}$.

Step 4: (Perform Contraction) Define

$$\mathbf{x}_{new} = \begin{cases} \mathbf{x}_{new} = \beta\mathbf{x}_{max} + (1-\beta)\hat{\mathbf{x}} & \text{if } f(\mathbf{x}_{max}) \leq f(\mathbf{x}_r), \\ \mathbf{x}_{new} = \beta\mathbf{x}_r + (1-\beta)\hat{\mathbf{x}} & \text{otherwise,} \end{cases} \tag{5.6}$$

and form the new simplex by replacing the vertex $\mathbf{x}_{max}$ with $\mathbf{x}_{new}$.

Figure 5.1 The Nelder-Mead simplex search algorithm.

reached or the simplex becomes sufficiently small:

$$\sum_{j=0}^{n} \| \mathbf{x}_j^k - \mathbf{x}_j^{k+1} \|^2 < \epsilon \tag{5.7}$$

where $\mathbf{x}_j^k \in S^k$ and S^k is the simplex generated at the k^{th} iteration.

5.4 Two-Stage Nelder-Mead Search

Given the hand state θ_t, the NM method begins by generating an initial simplex S_t^0 around θ_t, and the iterative procedure guides the search towards a minimum. Although NM method is suitable for searching when the gradient is not easily accessible, the basic form of NM does not take advantage of the motion constraints embedded in the feasible space representation Ψ. Instead of performing the unconstrained simplex search in the continuous domain, we restrict the simplex vertices θ_i to be one of the samples $\theta_j \in \Psi$. At the k^{th} iteration, a new vertex θ_{new} is generated as described in Section 5.3, and a nearby configuration θ' is located to replace $\theta_{max} \in S_t^{k+1}$:

$$\theta' = [\theta_{new}]^+ = \arg\min_{\theta \in \Psi} \| \theta_{new} - \theta \| \tag{5.8}$$

There are many algorithms and data structures designed for locating a nearest sample point in a set from any given location, such as Voronoi diagram [71], Kd-tree [72], and Approximate Nearest Neighbors [73]. In our experiment, we implemented the Kd-tree structure. Since the NM method will converge towards a minimum, the nearest neighbor does not need to be very exact. By constraining the searching to the discrete space Ψ, we enforce the prediction to be feasible states.

Since the data we collected cannot possibly cover the entire feasible space, there exist gaps and discontinuities in Ψ. Searching only in the discrete domain will not guarantee an optimal convergence; therefore, we propose to perform a two stage hierarchical simplex search. The first stage conducts the search in the discrete domain starting with a large simplex. Once the simplex terminates with a set of vertices in a smaller region, we then continue the iteration in the continuous domain with a more strict termination condition.

5.5 Stochastic Nelder-Mead

Like all bottom-up search algorithm, the NM search algorithm presented in Section 5.3 fails when the cost function has multiple minima. Because of the noise presented in image feature extraction and the nontrivial definition of the cost function, $f(\mathbf{x})$ can not be convex. Furthermore, the result of the NM algorithm outputs only the maximum likelihood estimate $\mathbf{x}_t^* = \arg\max_{\mathbf{x}} p(\mathbf{z}_t|\mathbf{x})$ when the cost function is defined as $f(\mathbf{x}) = -p(\mathbf{z}_t|\mathbf{x})$.

To tackle the problems of particle degeneracy in top-down approach and the nonconvex objective function for bottom-up approach, we propose a stochastic NM search algorithm by incorporating the NM algorithm within the SMC tracking framework. Rather than using a set of random samples to model the pdf evolution, we use a set of simplices, each generated from a mode of the pdf to locate the new modes through NM search. The new algorithm shares the advantages of both approaches to reduce the limitations induced by employing the NM search or particle filtering alone. From the bottom-up approach perspective, the implementation of multiple hypotheses and resampling greatly increases the chances of reaching global minima. And by considering the prior $p(\mathbf{x}_t|\underline{\mathbf{z}}_{0:t-1})$ (Equation. (4.1)) a more accurate hand state estimation can be achieved. From the top-down approach perspective, each particle will be close to a mode and have a significant weight. Cham and Rehg [74] also employed an iterative Gauss-Newton method to produce the modes of the likelihood $p(\mathbf{z}_t|\mathbf{x}_t)$. In our case, NM method is employed to handle the more general cases. The additional stage of the NM descent algorithm is similar to the propagation depicted in $p(\mathbf{x}_t|\mathbf{z}_{0:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-1})d\mathbf{x}_{t-1}$, where the dynamics $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is often modeled as a Gaussian. In the case of NM search, the iterative procedures automatically drives the simplex towards a new mode. One concern with this algorithm is the loss of diversity in particles as most of the particles will be forced to converge to a few peaks. This effect, known as *sample impoverishment*, can be reduced by having a larger perturbation when generating the initial simplex from each node. The complete algorithm is shown in Figure 5.2.

At frame t
FOR $i = 1$ to N

- **Sequential Importance Sampling**
  - Draw samples $\tilde{\mathbf{x}}_t^i$ from $q(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-1})$.
- **NM Simplex Search**
  - Generate one initial simplex for each particle: $S^{i0} = \{\mathbf{x}_j^0 = \tilde{\mathbf{x}}_t^i + \Delta_j, j = 1 \dots n\}$ where Δ_j is a random perturbation vector.
  - Perform NM search until termination criteria is met to reach S^{i*}.
- **Parameter Update**
  - Obtain the new particle from the centroid of the final simplex:

$$\mathbf{x}_t^i = \frac{1}{n+1} \sum_{\mathbf{x}_j \in S^{i*}} \mathbf{x}_j$$

  - **IF** the simplex converged according to Equation. (5.7), then compute the importance weight as

$$\pi_t^i = \frac{p(\mathbf{x}_t^i|\mathbf{z}_{0:t-1})}{q(\mathbf{x}_t^i|\mathbf{z}_{0:t-1})} p(\mathbf{z}_t|\mathbf{x}_t^i)$$

  ELSE assigned 0 to π_t^i.

END

- **Resampling**
  - Mutiply/discard particles using the systematic resampling scheme [68] to obtain a new set of N particles.

Figure 5.2 The stochastic Nelder-Mead search algorithm.

5.6 Experiments

Though inspired by the hand tracking problem, the SNM algorithm itself can be applied to other general tracking problems. Some results of applying SNM in a general template tracking scenario are shown in Section 5.6.1, and the hand tracking results are shown in Sections 5.6.2 and 5.6.3.

5.6.1 2D object tracking

The 2D templates are initialized in the first frame. The object state has 3 DOF, namely, $\{T_x, T_y, \theta\}$. Two tracking sequences are shown. Both trackers use normalized cross-correlation as the matching function to determine $p(\mathbf{z}|\mathbf{x})$. In Fig. 5.3, we track a person walking in the hallway. This sequence was motivated by the surveillance and security applications in which it is critical to be able to track human movements in a scene. In Fig. 5.4, the finger motion is accurately tracked. The results can be used for HCI applications that require pointing interfaces, such as a virtual mouse. In both sequences, five simplices are used to search in each frame. The results show that the algorithm can effectively track objects in cluttered background. The tracker operates at real-time on a 2-GHz Pentium PC for both sequences.

Figure 5.3 Tracking a person walking in the hallway.

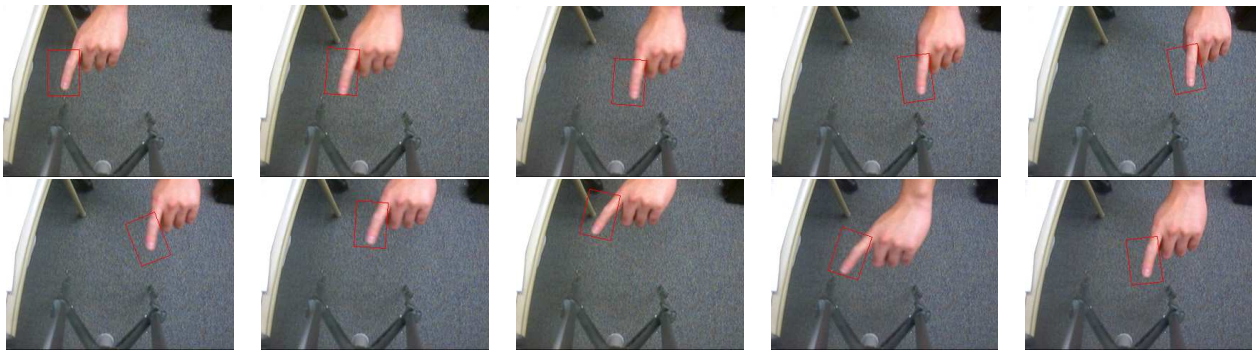


Figure 5.4 Tracking one finger pointing motion.

5.6.2 Capturing global hand motion

This section shows the results of testing the SNM method on real hand motions. We use a 3D hand model and each finger phalanx is represented using a truncated cylinder. Hand projection is generated as described in [39] and is superimposed on the real hand image. All experiments are performed in a cluttered background. The first sequence demonstrates the robustness of the 3D model-based tracking in which the hand undergoes both translation and rotations (Figure 5.5). The sequence shows that the 3D model can handle significant occlusions. The fingers are assumed rigid in this case. We used 10 simplex search for each frame.

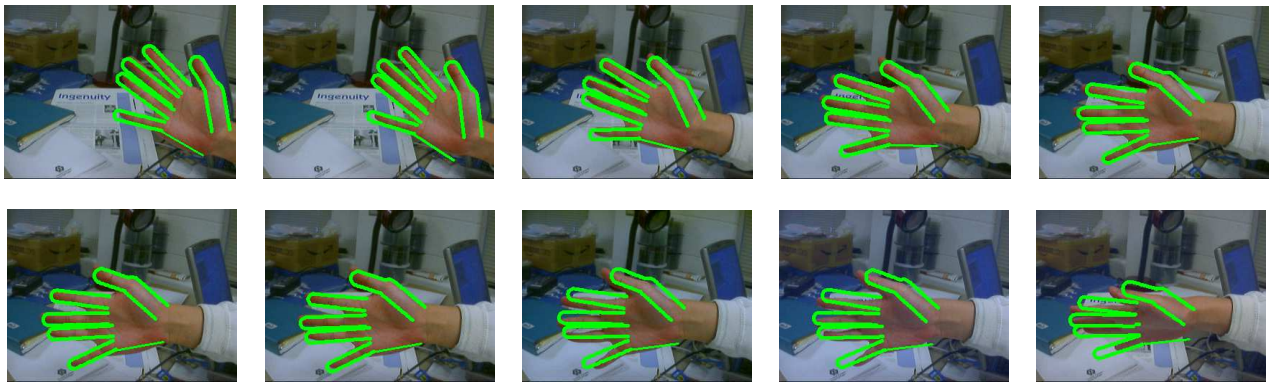


Figure 5.5 Tracking global hand motions involving translation and out-of-plane rotation. The projected model edge points are superimposed with the real hand image.

5.6.3 Capturing articulated hand motion

In this video sequence, the fingers bend and extend while the hand moves simultaneously (Figure 5.6). This experiment shows the robustness of our articulation tracking algorithm. We used 30 simplices for finger articulation tracking and 10 simplices for global motion. We have also tested the sequence using CONDENSATION algorithm with 5000 samples, and the algorithm fails after about 10 frames. In addition to the superimposed model projection, a reconstructed 3D hand model is shown below each corresponding image for better visualizations. The experiment results show that our algorithm is robust and successful in tracking complex hand motions in a cluttered environment. However, the current implementation still takes about 2-3s to process each frame on a Pentium 2-GHz PC.

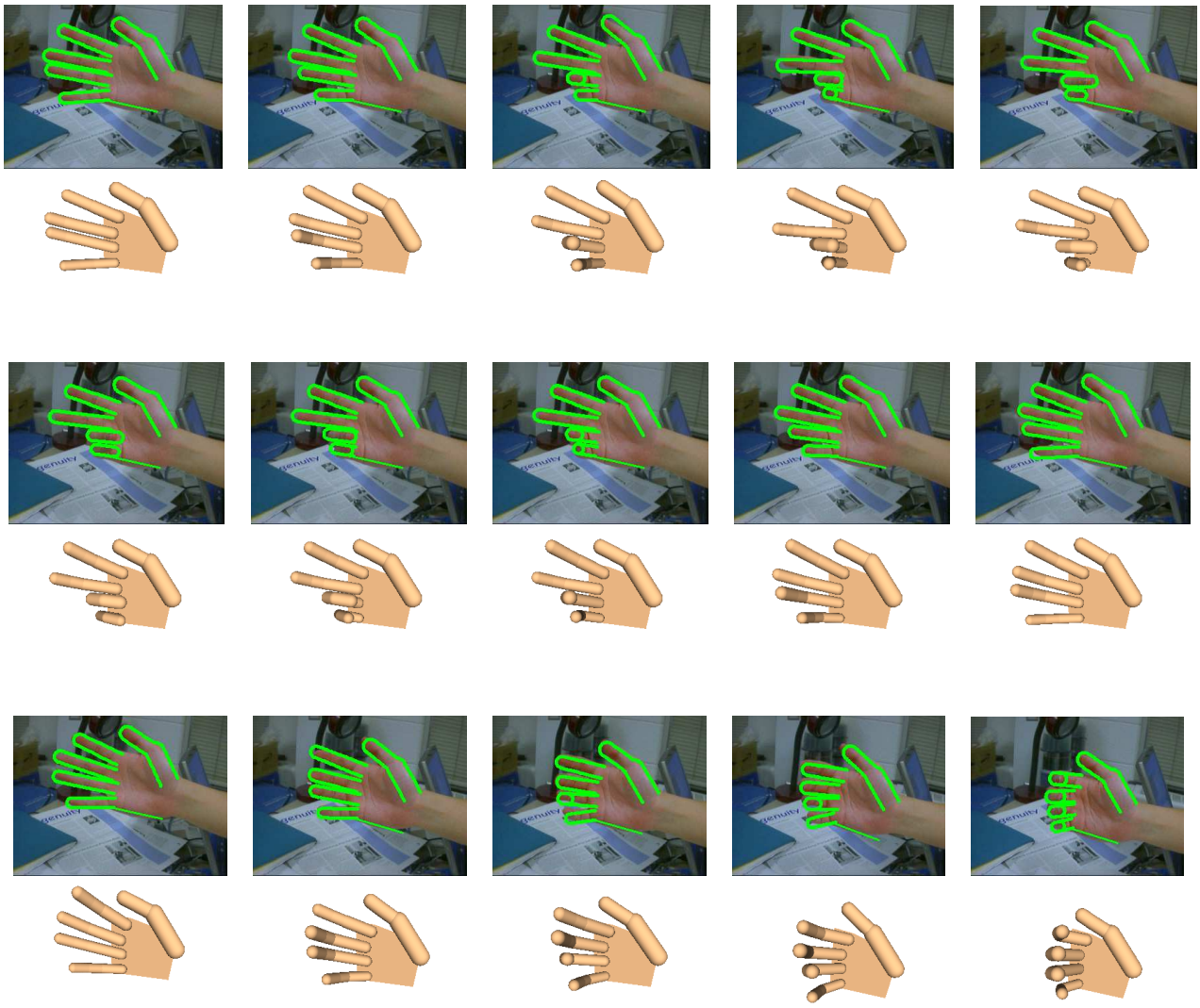


Figure 5.6 Simultaneously tracking finger articulation and global hand motion. The projected edge points are superimposed with the real hand image. Below each real hand image, a corresponding reconstructed 3D hand model is shown for better visualization.

CHAPTER 6

AUTOMATIC MODEL INITIALIZATION

6.1 Introduction

One of the crucial step in the articulate hand motion tracking is the initialization step. Given the complexity of the tracking algorithm itself, most of the existing tracking methods view the initialization as a separate problem and perform manual initialization. With careful maneuvers, one could obtain fairly accurate hand geometry and initial motion parameters. One automatic registration algorithm was proposed in [33] using shape context by assuming open hand gesture and frontal view. Assuming proper hand shape, their approach could give a good estimate of global orientation and location. However, this approach still requires the knowledge of the user's hand geometry beforehand. If only motion parameters are sufficient, one approach is to use appearance-based approach by incorporating the global motion parameters as part of the estimating parameters in the database and also training process. The user-dependence depends on the features used. Little past research work addresses the issues for fully automatic model acquisition and initialization. An ad hoc method was given in [75] which takes the estimated the center of the palm using the center of the mass of segmented hand region. The result of this estimation would depend largely on the finger abduction angles and outliers, which should not be included when inferring the palm geometry. Another approach [76] proposes identifying and connecting links of an articulated model from a sequence of frames.

To achieve true automatic initialization and user-independence, it will be beneficial to derive an algorithm that could automatically determine hand geometry parameters and simultaneously determine the global motion parameters, including position and orientation with as few assumptions

as possible. In this chapter, we present an algorithm for accomplishing this task using palm and fingertip detection. The process of initialization in this chapter refers not only the estimation of initial global motion parameters but also the estimation of hand geometry.

One potential application of our detection algorithm can is to use the result to perform hand geometry recognition in biometrics. The current verification systems [77, 78, 79] require the user to place the hand at certain locations defined by a few rigid pegs in order to register the hand images for verification. This limitation can be relieved if an algorithm is available that can automatically register the images and compute the abduction angles.

6.1.1 Assumptions

In the following we list a few assumptions made and the justifications. In order to automatically measure hand geometry, the system must be presented with a clear view of the hand. Contrary to the appearance-based approach in which a hand shape can be determined regardless of the hand pose and the hand geometry, all relevant parameters should be presented in the view in order to construct the model. This limits the number of possible views we could accept.

- To facilitate the process, our system requires the user to show a frontal view of an open palm that is relatively parallel to the image plane, regardless of left and right hand. This constraint permits a good estimate of the finger width and the size of the palm and the location of each finger.

- All five fingers must be extended and visible. This will ensure the correctness in the estimation of finger length. The assumption can be relaxed to allow 0 abduction angles if and only if a good edge map is present in order to provide sufficient cue. In most cases, however, the quality of the image obtained from a generic pc camera is usually low and noisy, and edge segmentation result is not reliable. Therefore, we rely on the silhouette to perform the task, and in this case, it is desirable to have the fingers open with sufficient abduction angle. Also, the effect of color bleeding and smearing is most amplified in the region between fingers when they are close together resulting in inaccurate segmentation results.

- Middle finger abduction is insignificant. This assumption comes from Equation (3.2), which

was also used in tracking. This assumption will be used to help determine the initial orientation of the hand and as a calibration step.

- The fingers are expected to point upwards in a setting shown in Figure 6.1. This is a reasonable assumption as it is most natural to raise one's hand in a setting like this when the interaction happens in front of the computer and the camera.

Though these constraints must be enforced, we try to limit the number of constraints necessary in order to relax the learning and burden from the user side. On the other hand, the user still has the freedom to place the hand in arbitrary orientation and position.

Figure 6.1 System setup.

6.1.2 Top-down versus bottom-up

To estimate the hand geometry parameters in order to construct a hand model automatically from the top-down approach, one generates several possible hand model candidates in different poses in order to approximate the model from the best match. This approach can be robust when the segmentation is noisy. However, the main drawback comes from the large number of parameters that we have to estimate. These include hand location and orientation, palm size, finger length, width and abduction angle for each finger, choice of left or right hand, and the location of the base of the fingers. These consists of at least 20 parameters. Random and exhaustive search for the best estimate is impossible with current computing technology.

On the other hand, by constructing the hand model from the bottom-up approach, we could exploit the physical and anatomical constraints provided by the hand model and the estimation involved in computation can be broken down into smaller problems. Rather than estimate in $\mathcal{R}^{20}$, we would be looking at several estimations in $\mathcal{R}^3$, which is a much easier problem. And by exploiting the kinematical model knowledge, we could refine the estimates of each module through co-inference process. Therefore, in our approach, we choose to build and infer the hand model from bottom-up and divide the problem into smaller modules. Basically, we break it into the following smaller problems: palm detection, fingertip detection, thumb detection, finger fitting and abduction angle estimate. The hand orientation is determined by the middle finger orientation. Once the thumb is identified, we could also determine whether it is a left or right hand.

6.2 System Overview

Figure 6.2 shows an overview of the automatic initialization system. The system can be roughly divided into the following modules: hand localization, palm detection, fingertip detection, and estimation of the remaining parameters for initial model construction. As can be seen, at each step, the output of each module can also be used as input for stand alone applications. For instance, the orientation and location of the segmented hand blob can be used for navigation controlling purposes [80] and the detected fingertip can be used for interactions that requires fine pinpoint input such as drawing [81]. In fact, we implement some of the gesture interface based on these cues and the systems are presented in Chapter 7.

Figure 6.3 shows a captured screenshot of the user interface. The video input is segmented and displayed using the CAMSHIFT application built by OpenCV. An OpenGL window is also shown to provide additional graphical feedback which can render 3D contents, which can be used for applications such as virtual environment navigation. A third window is displayed for text output.

6.3 Hand Localization

An initial segmentation is performed to detect skin regions. In [26], an infrared camera is employed to obtain clean segmentation based on the range of temperature. We choose to use an

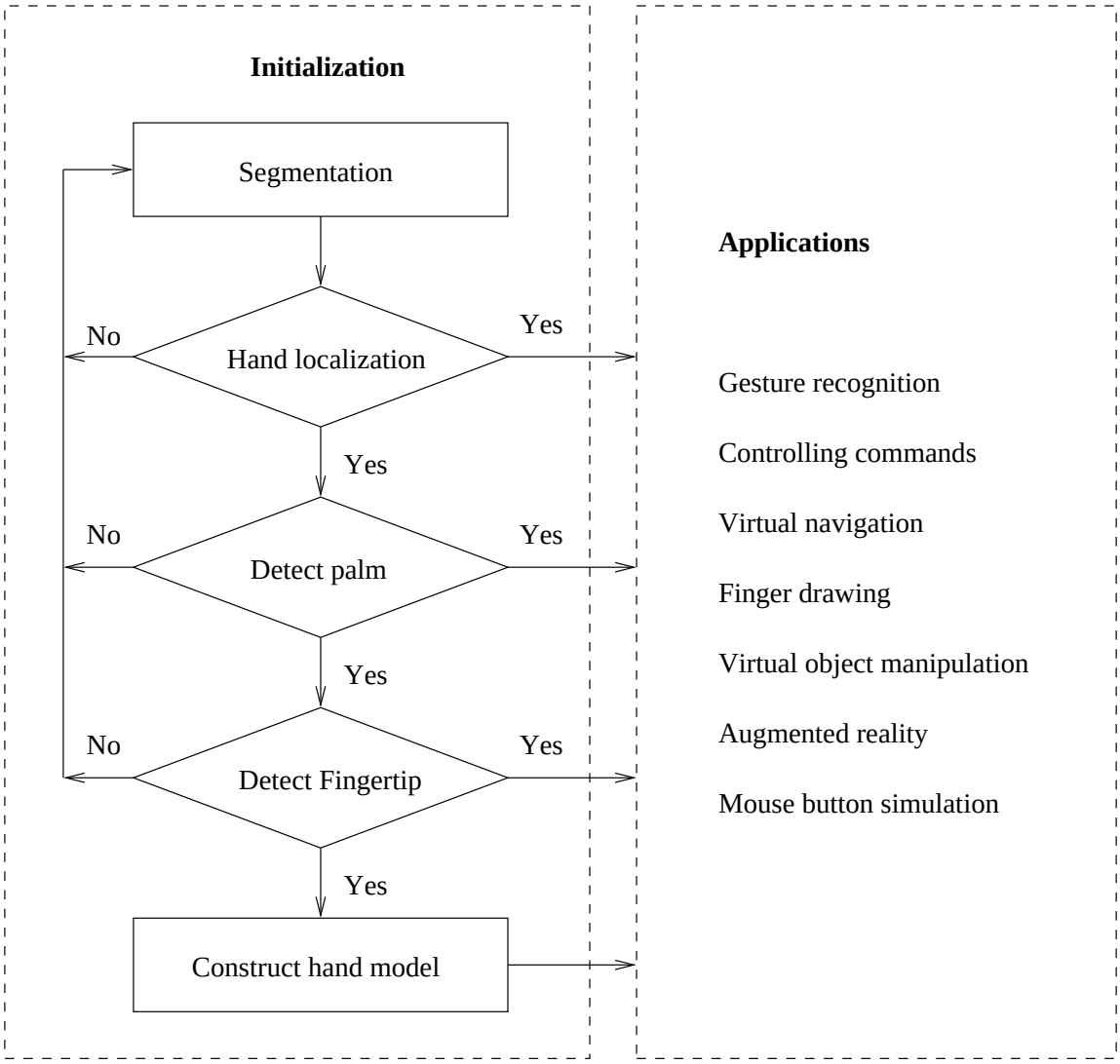


Figure 6.2 System overview.

off-the-shelf PC camera and decided to use CAMSHIFT [21] to perform this task. Our algorithm can work for somewhat noisy segmentation results and the performance can be improved by using cameras with higher qualities or infrared camera. Our subsequent algorithms are built on top of the application layer provided in OpenCV by Intel. The user is free to conveniently select foreground regions for training online. As the remaining algorithms rely heavily on the segmented silhouette, it is essential to have a robust segmentation algorithm.

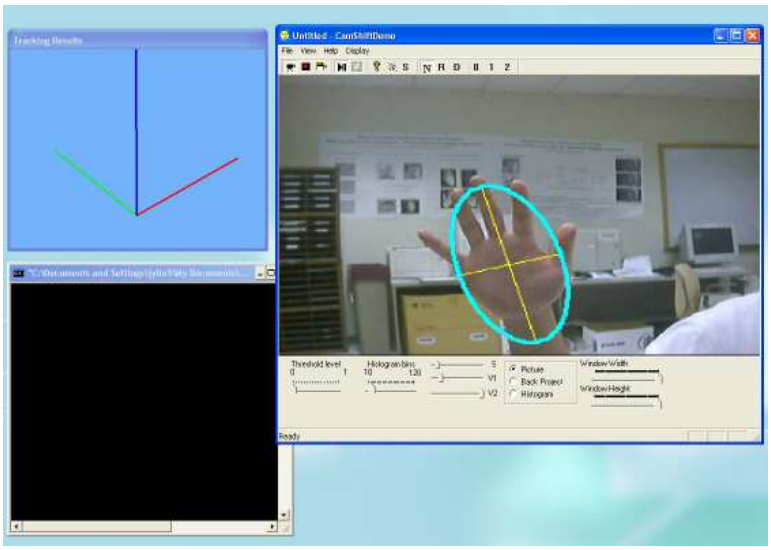


Figure 6.3 A screenshot of the user interface.

6.4 Palm Detection

In this section an algorithm is proposed for fast palm detection using top-down circle fitting approach.

6.4.1 The detection algorithm

The palm can be approximated using a circular object when the hand is facing the camera. We use a template to approximate the palm with a square and a smaller circle inside. Using straightforward template matching techniques, one can identify the location and scale of the palm by scanning through all possible ranges. Though the region outside of the circle should be filled with background pixels, because of the extended fingers, it is inevitable to have foreground pixels in the region outside the circle. The goal is to determine the largest circle fitting the palm and the following score is used to determine the proper dimension.

$$f(\mathbf{x}) = \sum_{(x,y)\in\mathbf{F}} I_{\text{foreground}}(x,y)/Area(\mathbf{F}) + \sum_{(x,y)\in\mathbf{B}} I_{\text{background}}(x,y)/Area(\mathbf{B}) \tag{6.1}$$

where $\mathbf{x}$ denotes the set of location and scaling parameters x, y, s. A hypothesis is considered valid if $f(\mathbf{x}) > T$ for some threshold T.

In our implementation, the final palm estimation is determined using the average of the quali-

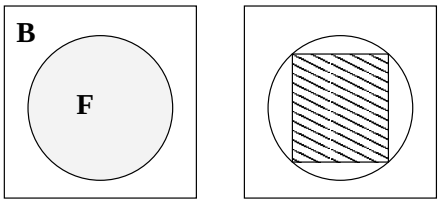


Figure 6.4 (a) Template for palm detection. (b) Shaded region denotes the feature for fast false hypothesis elimination using integral image.

fying candidates with a tight threshold set. Occasionally, due to noisy segmentation, a solid circle might not be present and no candidate will have a score above T. In this case the threshold is reduced. By assuming small motion, the one with highest score is kept provided that it is in the proximity of the previous estimate with similar size. Similarly, to cope with the noisy image input, a temporal smoothing is enforced. The parameters at current frame are only updated if the new estimate differs from the previous estimate with a significant amount:

$$dist(\mathbf{x_t}, \mathbf{x_{t-1}}) > \epsilon_1, \text{and} \;\; dist(r_t, r_{t-1}) > \epsilon_2$$

However, by checking every pixel for every template matching for every scale and location will take up too much computation power. We propose implementing several methods to speed up the process and to maintain the robustness and stability of the detection: integral image, dynamic search region, and adaptive subsampling.

6.4.1.1 Integral image

The matching procedure can be sped up by approximating the circle using a square and eliminating false hypothesis quickly using integral images. A false hypothesis can be easily determined if the interior of the circle is not filled with enough foreground pixels. In our implementation we check the area inside the shaded square within the circle (Figure 6.4) and by using this feature, integral image representation can be used to help speed up the process. Integral image [82] (see Figure 6.5), also known as summed-area tables [83], is an intermediate image representation that facilitates the computation of arbitrary rectangular area of a grey scale image. The value at location (x, y) of the

integral image I_i stores the sum of the pixel values in image I_o the left and above of (x, y):

$$I_i(x, y) = \sum_{x \leq x', y \leq y'} I_o(x', y')$$

The integral image is constructed recursively, which can be done in one pass:

$$\begin{aligned} &I_i(x, y) = I_i(x, y-1) + I_i(x-1, y) + I_o(x, y) - I_i(x-1, y-1) \\ &I_i(-1, y) = 0, I_i(x, -1) = 0 \end{aligned} \tag{6.2}$$

Using this intermediate representation, the sum of an arbitrary rectangle can be computed easily using only four references:

$$S(x, y, w, h) = I_i(x, y) + I_i(x + w, y + h) - I(x, y + h) - I(x + w, y)$$

where x, y denotes the the position of the upper left corner of the rectangle, and w and h are the width and height, respectively.

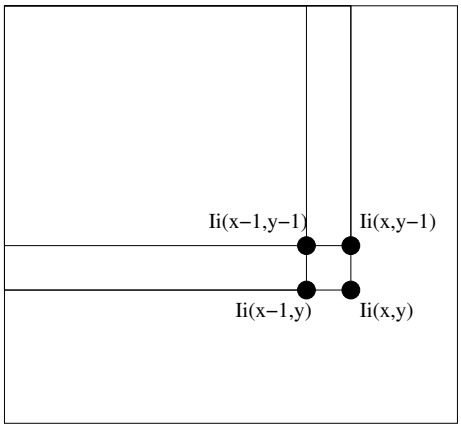


Figure 6.5 Integral image. The value at location (x, y) stores the sum of the pixel values to the left and above (x, y) from the original image. The construction of integral image can be done in one pass (Equation (6.2)).

The advantage of using a square inside the circle with width $2\sqrt{2}r$ rather than using a square bounding the circle with width $2r$ is that, for the former case, we can set the threshold high and be certain that it is a filled square that fills the interior of the circle. On the other hand, the latter case includes background region so that even if the interior area is above the threshold, there is a

possibility that the interior foreground pixel might not be distributed evenly and compactly within the circle.

6.4.1.2 Dynamic search region

Based on the bounding box provided by CAMSHIFT, a tracking window, which is slightly larger than the CAMSHIFT window is computed to focus the search in the relevant area. The size of the template is also dependent on the size of the tracking window and is empirically determined to be $0.2 * min(W, H) \leq w \leq 0.4 * min(W, H)$ where W, H denotes the width and height of the tracking window and w is the width of the template window.

6.4.1.3 Adaptive sampling

Another resource allocation scheme is implemented by adopting adaptive subsampling. It is computationally expensive to check every pixel, especially when the hand region is large. Therefore, the algorithm subsamples the image with the decimation factor s, which is automatically determined based on the size of the template window. Thus, when the hand is close to the camera, the tracking window is enlarged, and a larger template is required to perform detection. A larger s is used in this case in order to speed up the matching while maintaining the accuracy. When the search window is small, a more detailed search is required and a smaller s is used. Using a fixed s would lead to oversampling for large search window and undersampling for small search window.

6.4.2 Results

In this section we show some results of the palm detection. As can be seen, the detection is fairly robust to noisy segmentation results (Figure 6.6) and also to different number of extended fingers. The estimated location and radius parameters are plotted for a sequence of 1500 frames (Figure 6.7). In this sequence, the hand moves around in the image plane and with different depths. We select a segment of the sequence and zoom in to show that the detection result is quite smooth within small windows (Figure 6.8). In Figure 6.9 a comparison is also shown for the case with and without temporal smoothing. The dash line in red is the original estimate and the solid line in blue is the smoothed result which shows a more stable output.

Figure 6.6 Palm detection results. The corresponding frame number is also shown under each image.

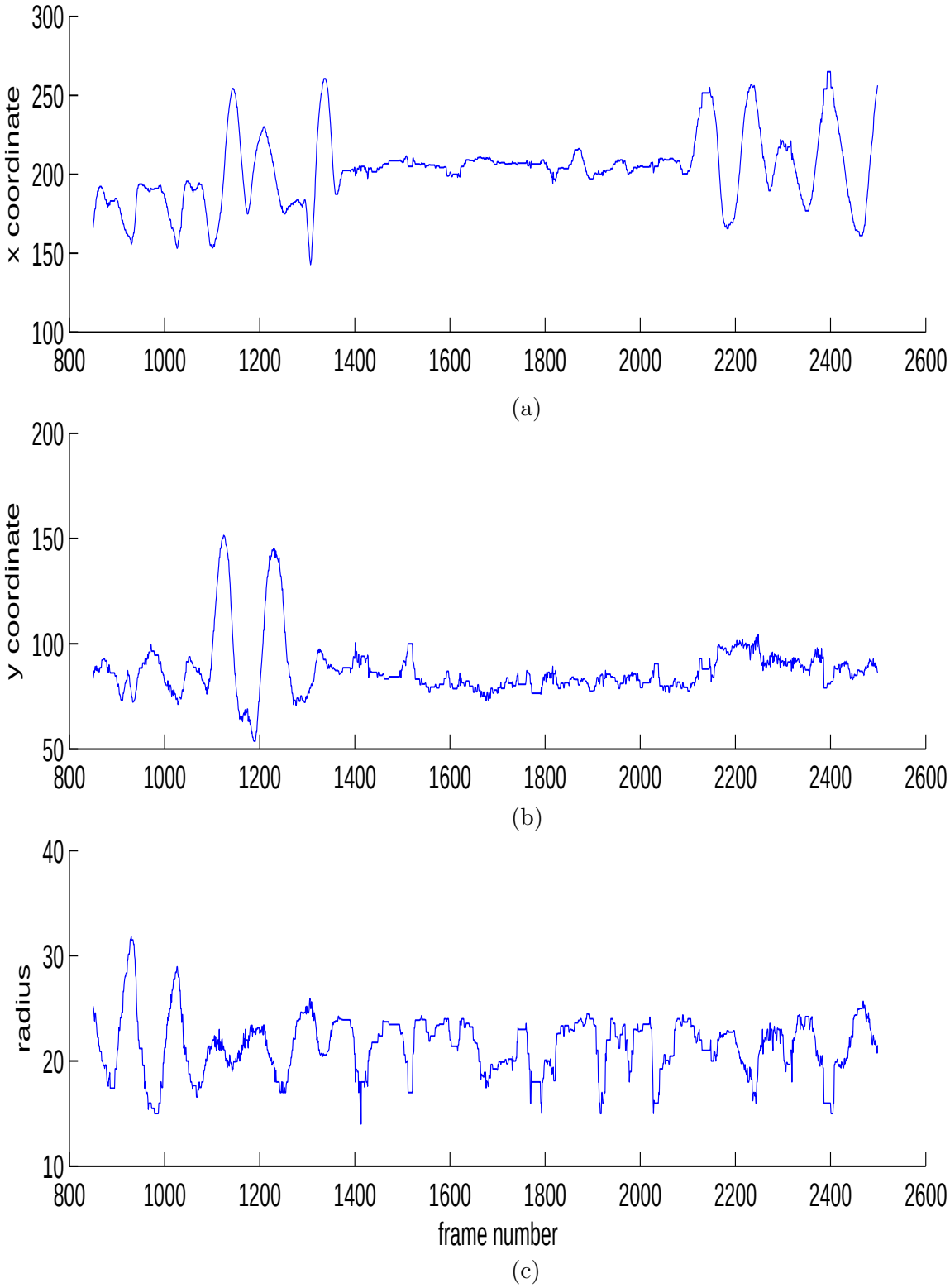
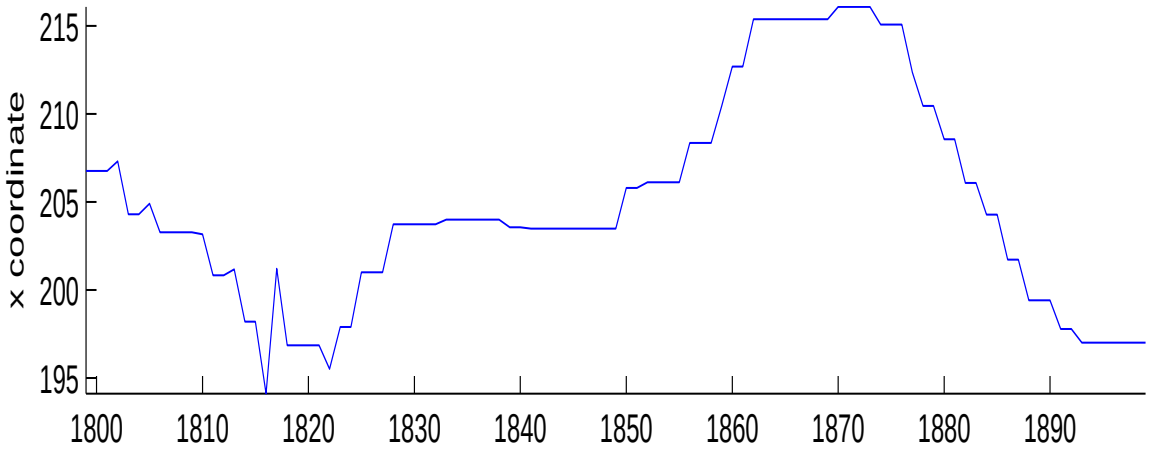
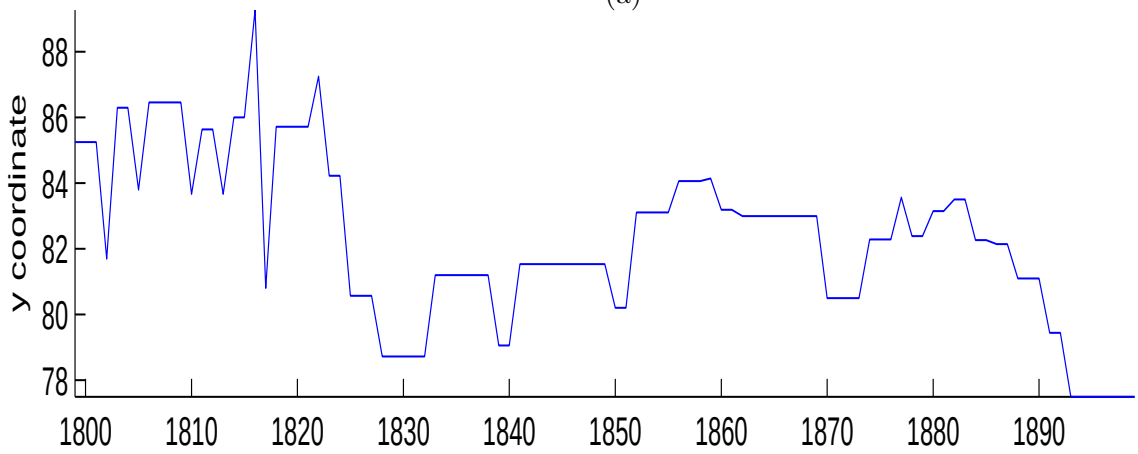


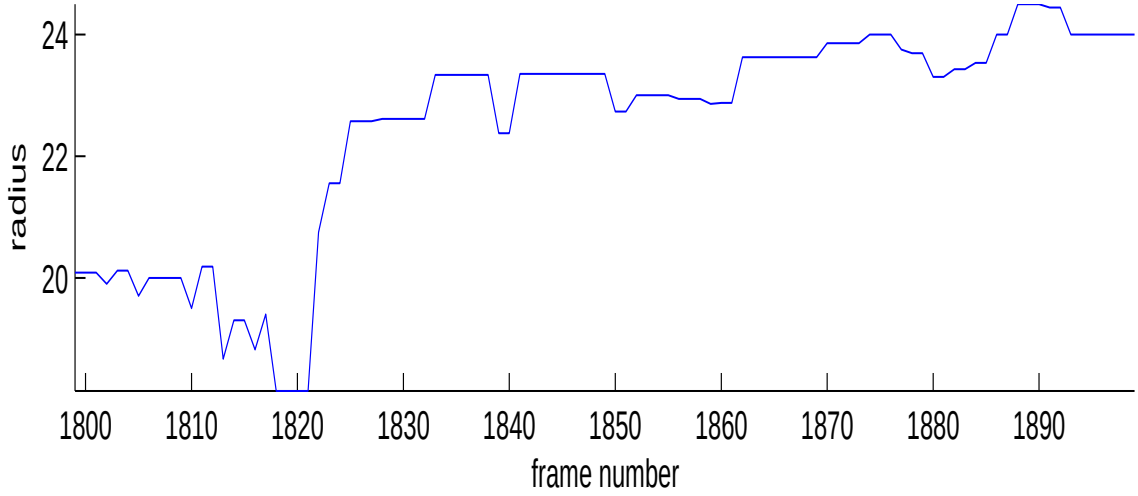
Figure 6.7 Plot of palm position and size parameters x, y, r. (a) x. (b) y. (c) r.



(a)



(b)



(c)

Figure 6.8 Plot of palm position and size parameters x, y, r. (a) x. (b) y. (c) r. In this figure, a zoomed-in section of the plot shown in Figure 6.7 is displayed. The results shows that the parameters estimated are quite smooth and stable over a small window and are suitable for gesture interfaces.

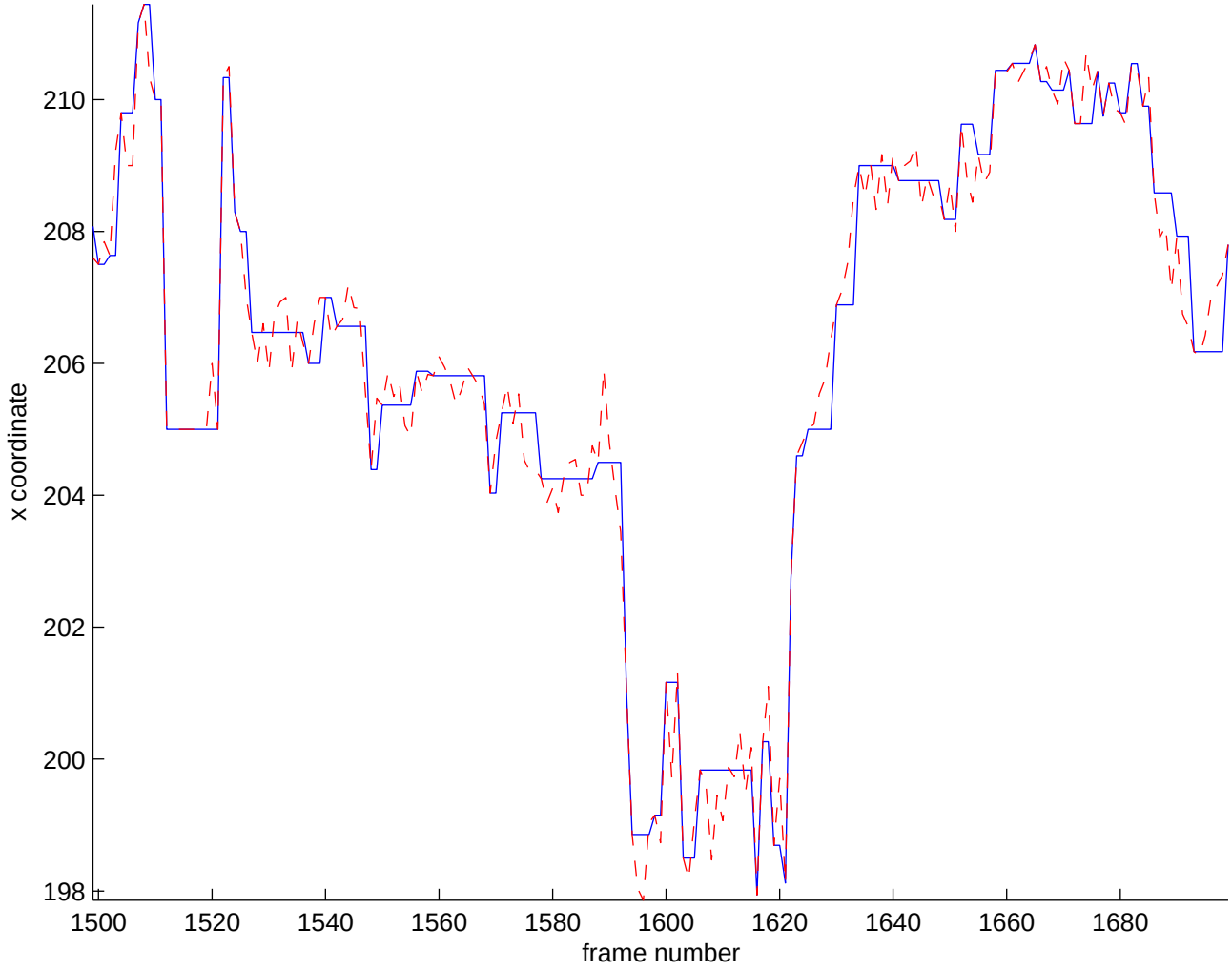


Figure 6.9 Comparison of palm parameter estimation. The noisy red dash line corresponds to estimation without temporal smoothing. The more stable blue solid line represents the estimation with temporal smoothing.

6.5 Fingertip Detection

In this section an algorithm is proposed for fast fingertip detection once the palm is detected. Based on the palm location and size, we are able to constrain the search space for possible fingertip candidates. Learning algorithm can also be used such as in [84].

6.5.1 Fingertip representation

The fingertip detection algorithm proposed in this section is inspired from the work of [26, 27]. Rather than using a hand model to identify fingertip locations, the algorithm uses template

matching technique that can be efficiently implemented to run in real time.

Figures 6.10 and 6.11 show some examples of typical fingertip images. One can observe that a fingertip can be well represented using simple geometrical templates shown in Figure 6.12. Basically, the shape of the fingertip in 2D images resembles a circle attached to one end of a truncated cylinder. This originates from the observation that a finger can be approximated by a cylinder with a hemispherical cap. Two observations were pointed out in [27]. First, a filled circle can be used to match the fingertip and also estimate finger width. Second, the majority of the perimeter pixels on the boundary of the box are nonskin pixels, and the skin pixels should be connected in a continuous chain. This representation has been shown to work robustly and efficiently, and the observations could be used to help classify fingertip hypotheses. A template like the one used for palm (Figure 6.4(a))is used to search for fingertip candidates.

Figure 6.10 Example of fingertips from the original image.

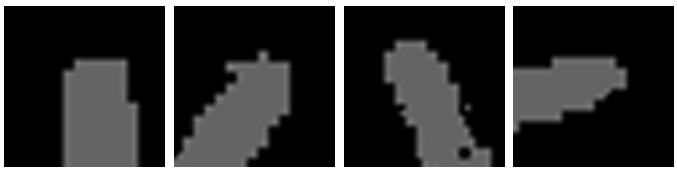


Figure 6.11 Example of fingertips after color segmentation is performed to classify skin pixels. The results shows that the output of the segmentation algorithm can be very noisy.

6.5.2 Fingertip detection

In the implementation proposed in [26], the distance between the hand and the camera is relatively constant. In their setting the camera is fixed to view from the top of a desk and the hands are expected to move right above the desk. Therefore, the search window has a fixed size and a fixed-size template is used to detect all possible fingertip candidates. In [27], an extension is made to differentiate the thumb and the pinky fingertips by assuming the ratio of diameters between thumb and pinky fingertip is roughly 1.5.

Figure 6.12 Approximation of finger shape.

In our approach, the palm is used to help to guide the search for the fingertips. Therefore, our algorithm can work with arbitrary distance between the hand and the camera as long as the entire hand is visible. Once a palm template is determined, the template is searched for different scales determined by the palm size. We used $0.2r_{palm} < r_{fingertip} < 0.4r_{palm}$ in our implementation. By not fixing the search to a fixed ratio allows the algorithm to obtain finer estimations of each finger width and also allows higher tolerance to the noise in palm parameter estimation.

Since the template used for fingertip is exactly the same as that of the palm, the same search procedure is applied which maximize the matching score defined by Equation (6.1). A sufficient number of foreground pixels must be present in the center of the circle as well as the background pixels outside the circle. The same performance enhancement is also applied, including the use of integral image, dynamic search range and adaptive subsampling. The search range is defined to be within a circle with radius $f(r_{palm})$ centered at palm center. Two scores were used to reduce the chance of missing a correct fingertip. Sometimes with noisy segmentation results, the fingertip does not always resembles a circle. Therefore, we allow a candidate to be accepted for a lower threshold if it is near a previously identified fingertip location. This ensures that at least one reasonable estimate will remain in the hypothesis pool.

However, using the method described in Section 6.4, the algorithm also picks up many false positives, including noisy background and regions that belong to the interior of a finger. Additional verifications must be performed to ensure the validity of a fingertip candidate by exploiting the knowledge of the hand structure. In the following we enumerate some scenarios we encounter (Figure 6.13) and the algorithms we propose to eliminate false alarms:

- Since the fingertip template is a much smaller one compared to the palm, it often picks up noisy background objects. However, the fingertip must be part of the connected component to the hand. By performing connected component analysis, we could easily determine whether the detected candidate belong to a background object.

- Using only the criteria defined in Equation (6.1) is not sufficient, and often the regions on the side of the fingers are also selected as valid candidates. To remove these kinds of false candidates, we examine pixels surrounding the perimeter of the square template. At least two adjacent sides of the square template must be completely filled with background pixels. If multiple pixels in a diagonal direction on the boundary are foreground pixels, then the candidates must be located within a finger.

- While the previous two methods effectively remove false candidates, several candidates still remains that are close to the true fingertip locations. Contrary to the palm detection approach in which we take the average of the top matches, here we choose to eliminate hypotheses that are in the neighboring region of a candidate with the highest matching score. Figure 6.14 shows some results of the comparisons. Experiment results show that a better estimation could be obtained by taking the best match. In [26], the top 20 candidates are selected and the multiplicity is reduced by keeping the highest match in a neighboring region. While this might be a sufficient number, there is still a chance of losing a true fingertip estimate, and with noisy segmentation results, some fingertip hypothesis might not have a very high matching score.

Finally, temporal smoothing is performed for each remaining fingertip candidate by searching for the nearest previous estimate and compare the location and radius. Update is performed only if sufficient change is detected. The threshold must be carefully tuned. If it is too tight, then the

Figure 6.13 Examples of false fingertip detections. In particular, we try to eliminate three types of false alarms: (1) hypotheses belonging to background object, (2) hypotheses near the edge or the interior of a finger, and (3) hypotheses that are close approximation of the true candidate.

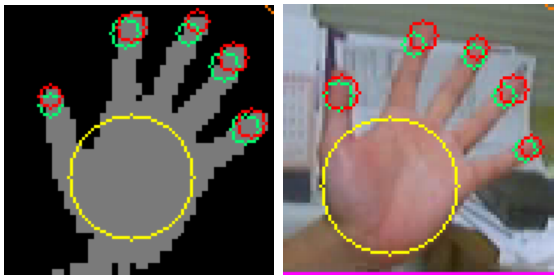


Figure 6.14 Comparisons.

noise will dominate and lead to unstable estimation. If it is too loose, then the estimation will be insensitive to a smooth fingertip motion. A summary of the fingertip detection scheme is shown in Figure 6.15.

- Determine proper dimension and scaling parameters.
- Template matching. Accept hypotheses with score $f(x, s) > T$
- Identify unconnected components as false detections.
- Remove hypotheses that belong to the interior region of a finger
- Suppress candidates with lower matching scores within the same neighborhood.

Figure 6.15 Fingertip detection algorithm.

6.5.3 Results

The fingertip detection algorithm, which runs in conjunction with the palm detection algorithm, can run for up to 60 Hz on a Pentium 2.8-GHz PC. In Figure 6.14 we show some comparisons between estimating the fingertips using hypotheses averages and the hypothesis with highest matching score. For each finger, the most likely hypothesis is drawn using a red circle. The detection results using the algorithm outlined in Figure 6.15 is shown in Figure 6.16. Similar to the palm detection, we are able to obtain robust and stable estimations. When more fingers are bent, the area of the palm shown are generally smaller than the case when all five fingers are extended. However, since we do not assume any fixed fingertip size, the algorithm can still robustly detect the fingertip location with correct dimension estimation. In Figure 6.17 we show the result of number of fingertips presented at every frame during a 600-frame sequence. Some glitches are observed due to motion blur and the ambiguities resulted from finger motion. When a finger bend down or extend, the joint angles are sometimes mistaken as a fingertip for a brief moment.

6.6 Thumb Identification

Once all the fingertips are detected, we would like to be able to determine whether the hand is a left or a right hand given that the frontal view is observed. The fingertip location is first sorted in a counterclockwise order with respect to the center of the palm starting from the $-y$ axis. This step is taken to facilitate subsequent process for constructing the hand model. With the assumptions that palm is open and in the upright position, the sorting helps to identify the thumb and pinky as the two fingertips at the beginning and the end of the sorted fingertip list.

Based on the detected fingertip location and dimension, the thumb can be determined using several methods. Von Hardenberg and Bérard [27] proposed to use the size of the circle to differentiate the thumb from pinky. However, this can be an unreliable estimate due to the noisy estimation of the thumb. Though the location can be reliably identified, the size of the thumb circle is often noisy. When the hand frontal view is present, the thumb is actually viewed from the side most of the time, and the simple geometrical object approximation shown in Figure 6.12 would not work very well with the thumb. Another method proposed in [75] first estimates the center of

Figure 6.16 Results of fingertip detection. The corresponding frame number is shown under each image.

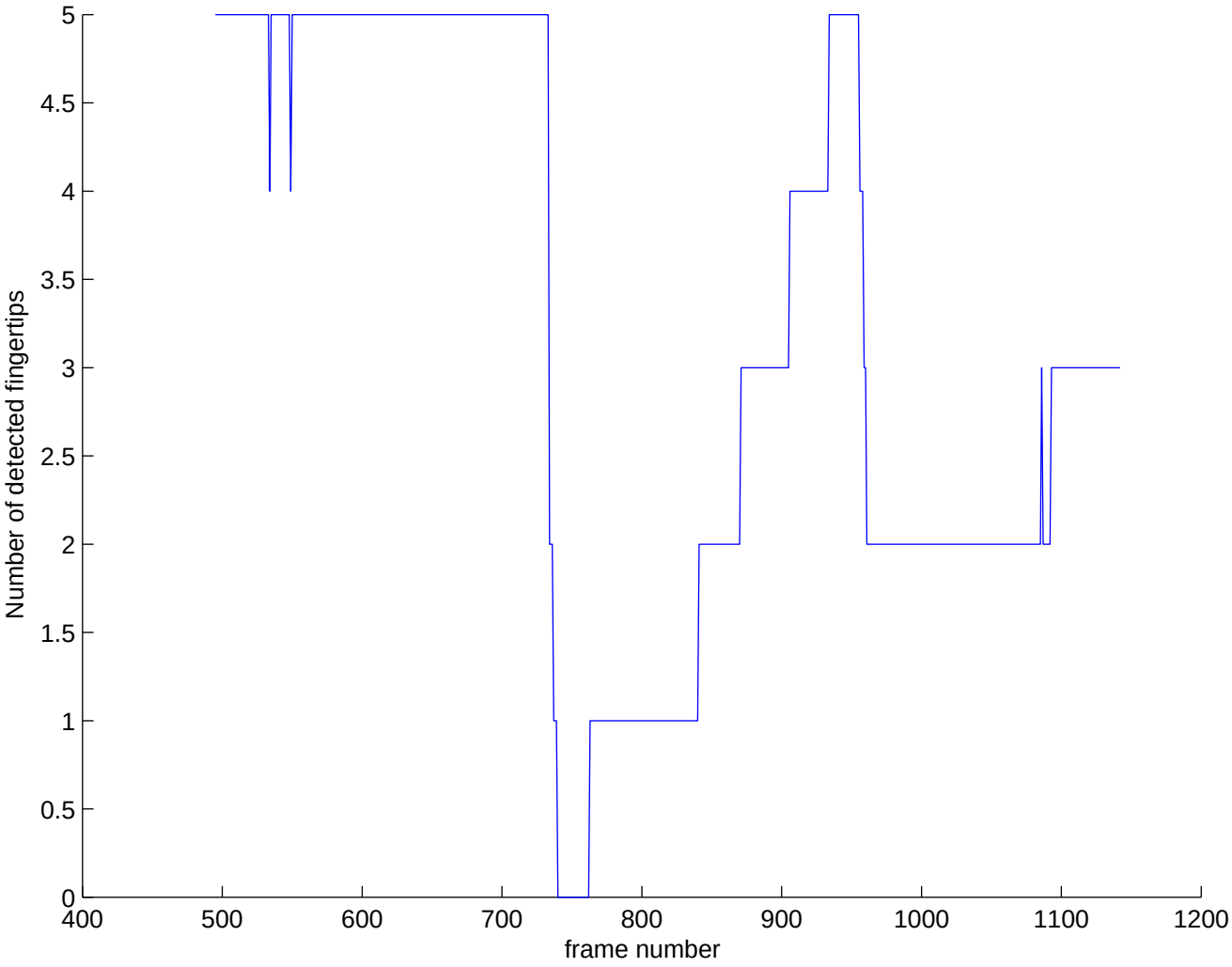


Figure 6.17 Number of detected fingertips.

the wrist, and then, using the fact that the thumb is closer to the center of the wrist, the thumb is thus identified. However, the estimate of the wrist center can be unreliable.

One alternative is to use the contour information to help differentiating the thumb. An example is shown in Figure 6.18 in which the red contour pixels correspond to the fingertips and the blue pixels corresponds to the valleys between fingers. We can easily identify the peaks and valleys that are associated with the fingertip and base of the finger using the distance profile computed from the distance between each contour pixel and the palm center (Figure 6.19). The valley between the thumb and the index finger is further away from the index fingertip than the valley between the ring and the little finger to the ring fingertip. However, like edge segmentation, contour processing

Figure 6.18 Hand contour processing. The red contour pixels correspond to the fingertip region, and the blue pixels identify the valleys between two fingers.

can be very noisy, and making the decision using one single frame is unreliable, especially when the hand region is small and resolution is low. We implement a voting scheme that accumulates the decision outcome over a period of n frames. The majority vote determines the fingertip candidate for thumb. Some results of the final decision is shown in Figure 6.20. Knowing the position of the thumb with respect to other fingertip then immediately identifies whether a left or a right hand is present.

6.7 Constructing the Hand Model

Based on the palm and fingertip detection results, we can infer and construct an initial hand model by fitting thumb and finger models, and we can obtain the initial motion parameters of the hand including location, orientation, and scale.

6.7.1 Finger fitting

Based on the observations shown in Figure 6.12, we can estimate the finger geometry by fitting a rectangular template. The width of the finger is determined as the twice the radius of the corresponding fingertip radius. Using the location of the fingertip as the constraint, one end of the rectangle must coincide with the fingertip. To identify the best fitting candidate, the parameter estimation can be formulated as an optimization problem with the finger orientation θ and the unknown finger length l as the parameters. However, we could easily reduce false candidates by

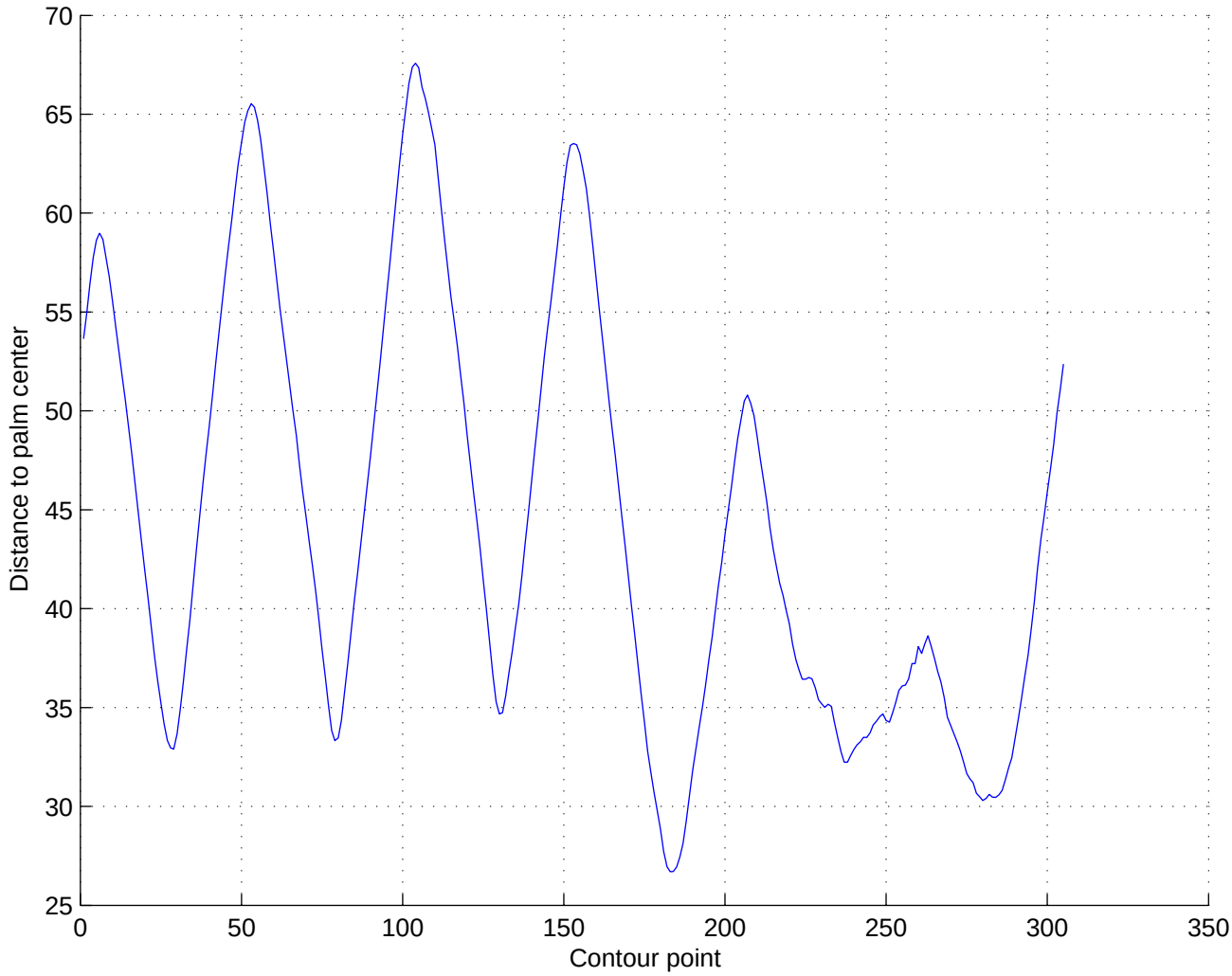


Figure 6.19 Contour pixels distance profile.

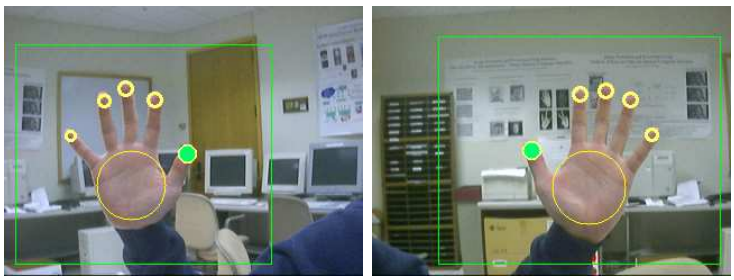


Figure 6.20 The result of detecting the thumb and identifying left and right hand.

exploiting prior model knowledge, knowing that a fingertip must be connected to a hand. Therefore, we know the other end of the rectangular template must locate nearby the perimeter of the palm. This knowledge simultaneously reduces the number of possible candidates both in terms of θ and

l together. Further speed improvement can be made by using different levels of subsampling to quickly eliminate the incorrect hypotheses. This method can also be used to determine whether a fingertip is a valid candidate or a false alarm instead of performing connected component analysis.

One key concern is the estimated location of the finger base which directly affect the finger length estimation. The MCP joint of a finger does not locate exactly at the edge of the palm. For the i^{th} fingertip with radius r_i, we generate candidates of finger base location on the circle centered at palm center with radius $r = r_{palm} - 2 * r_i$ and have obtained a good estimate of the hand geometry. Also, since the fingertips are ordered, we can use it to help avoid collisions in finding the base of the fingers. When the length of each finger is determined, we could use anthropometric data to obtain an averaged estimate of the length of each phalanx [85].

At this point we could use the three sets of estimated parameters from palm, fingertip, and finger to refine the overall estimation result using algorithms like co-inferencing [8] or simply use an iterative coordinate descent procedure. This is especially useful if the initial fingertip estimation is noisy and incorrect. Even if the palm and fingertip size are estimated correctly and independently, this refinement could still be used to obtain a better estimate of the motion parameters, including position of each object and the abduction angle between each fingers.

6.7.2 Thumb fitting

Thumb fitting presents another difficult problem since the structure of the thumb is quite different from the structure of the other four fingers. The three phalanges of the thumb do not always align in a straight line when the user naturally opens his hand. Therefore, one can not simply fit a single rectangular template and hope to obtain a good structure approximation. We propose using two rectangular templates to estimate the thumb structure. The first template matches the distal and proximal phalanges. The second template matches the metacarpal phalanx and the width of the of this rectangle is slightly larger than the other template. An approach similar to that described in Section 6.7.1 is used to reduce the search space.

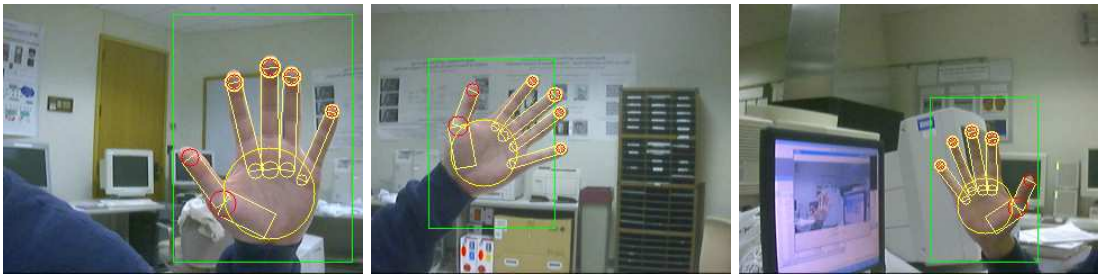


Figure 6.21 Constructing an initial 2D hand model.

6.7.3 Initial global motion parameters

The initial location and scale is determined using the location of the palm. Since a circle is used to approximate the palm, it does not provide any orientation information, and additional cues must be considered. Here we exploit the fact that middle finger abduction in the natural configuration is often 0 (Equation (3.2)), especially when the middle finger is at its natural configuration. Since the user is expected to open his or her hand naturally for the initialization, we can assume that the middle abduction angle is almost zero. Consequently, the orientation of the middle finger is used as the initial orientation of the hand.

6.7.4 Results

The results of constructing the initial 2D hand model from the palm and fingertip detection are shown in Figure 6.21. The algorithm is tested for hands at different location, orientation, and scale. Figure 6.22 shows the 3D model inferred from the 2D data. We use the model shown in Figure 2.7 which consists of 40 quadrics components. The three images shown in the figure each has different position, orientation, scaling, and finger abduction angle parameters. To verify the results of constructing the 3D model, we use the learned model to perform model-based hand motion tracking. Because of the large amount of computation involved in the 3D model-based approach, we only track one DOF in order to achieve real time performance. Specifically, the out-of-plane rotation is tracked to demonstrate the capability of using a 3D model in tracking and also the accuracy of model parameter estimation from 2D data (see Figure 6.23).

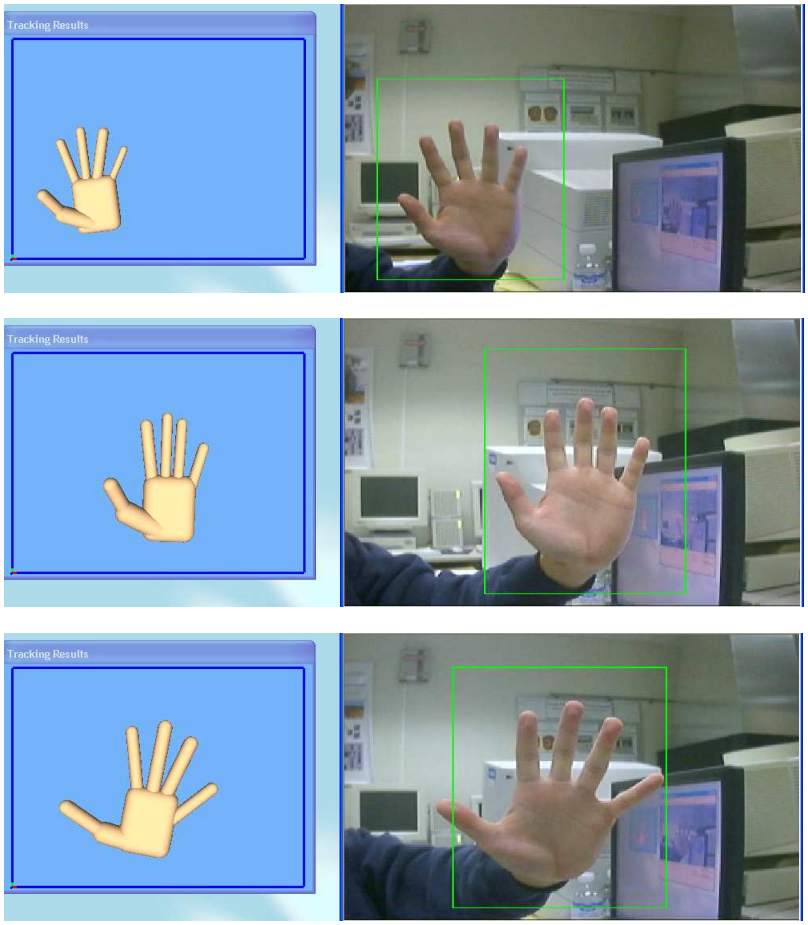


Figure 6.22 Results of inferring the 3D hand model from 2D data. Each image shows a hand with different position, orientation, scaling, and finger abduction angles.

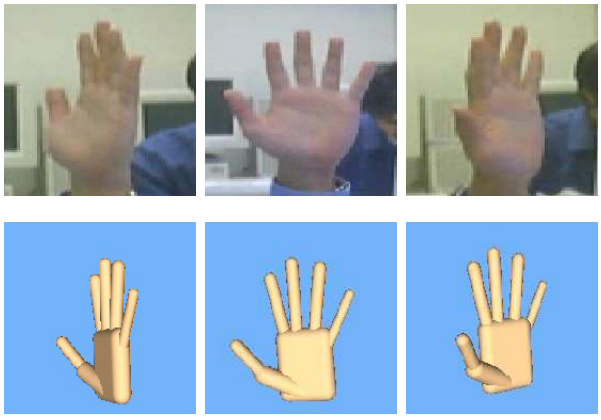


Figure 6.23 Results of using the learned 3D model to track out-of-plane rotation.

CHAPTER 7

VISUAL GESTURE INTERFACES

7.1 Introduction

This chapter describes four vision-based gesture interface systems including virtual object manipulation, virtual terrain navigation, finger drawing and gesture recognition. The systems demonstrates how user can interact with the computer in various scenarios through the use of the bare hand as a natural means of communication. Through the use of different gestures, users are capable of performing complicated actions and multidimensional control that are difficult to achieve using just the mouse and the keyboard. Many of the systems presented in this section are based on the hand detection results described in Chapter 6.

7.2 Virtual Object Manipulation

Object manipulation is a fundamental and important task in virtual reality, and augmented or mixed reality environments. To provide the user a truly immersive experience in interacting with the virtual object using gesture control, the system must be able to effectively interpret hand motions while allowing user to control in a simple, intuitive and efficient manner. In this section we describe an interactive system that allows the users to control the position and orientation of several objects.

7.2.1 System description

To simulate a simple AR environment, we define the image plane as the input plane and place several artificially created 2D objects directly on the screen. The users can freely move these objects around in this plane. Two examples are used. The first one places several colored balls on the image plane which the users can control using just two fingers. The second example demonstrates the advantage of using the hand over the mouse to perform multidimensional control. A box is used in this example and the user can rotate and move it around at the same time. Using the mouse input, only one of the button actions can be performed in conjunction of mouse movement. Consequently, one can only perform rotation and translation separately.

7.2.2 Gesture interface

Three modes are defined for this system. When all five fingers are detected, a reset command is issued. In this case, all the virtual objects are returned to their original locations and orientations. When exactly two fingers are present, the system determines whether an object is selected and appropriate motion update is performed. For all other hand configurations, the motion of the hand does not trigger any visual feedback.

The criteria for triggering a `select_object` action is based on the distance between two fingertips. If the distance between the two fingertips are smaller than the diameter of an object, and the midpoint of the line segment connecting the two fingertip centers are within the object boundary, then the object is considered selected (Figure 7.1). In this case, the position of the object is updated based on the midpoint between the two fingertips. Since the user can perceive feedback only by moving the hand close to the object drawn on the screen, it is natural to use absolute coordinates to update the object position. However, for the case of orientation update, a more intuitive manipulation is to use relative motion. The difference between orientations of the line segment L1 in consecutive frames is used to update the object orientation. A flow chart of the interface design is shown in Figure 7.2

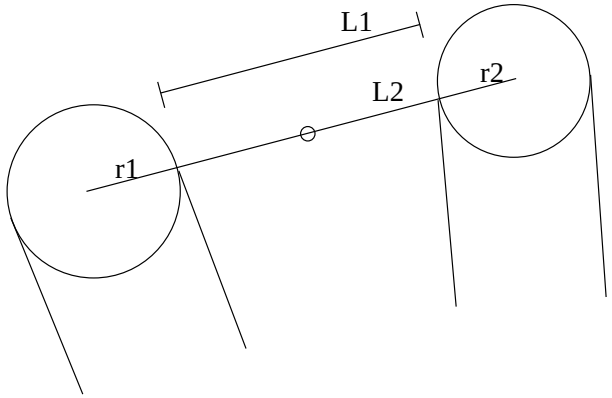


Figure 7.1 The distance $L1$ between two fingertips is used to decide whether a virtual object is selected. The midpoint between the centers of the fingertips determines the new location of the object.

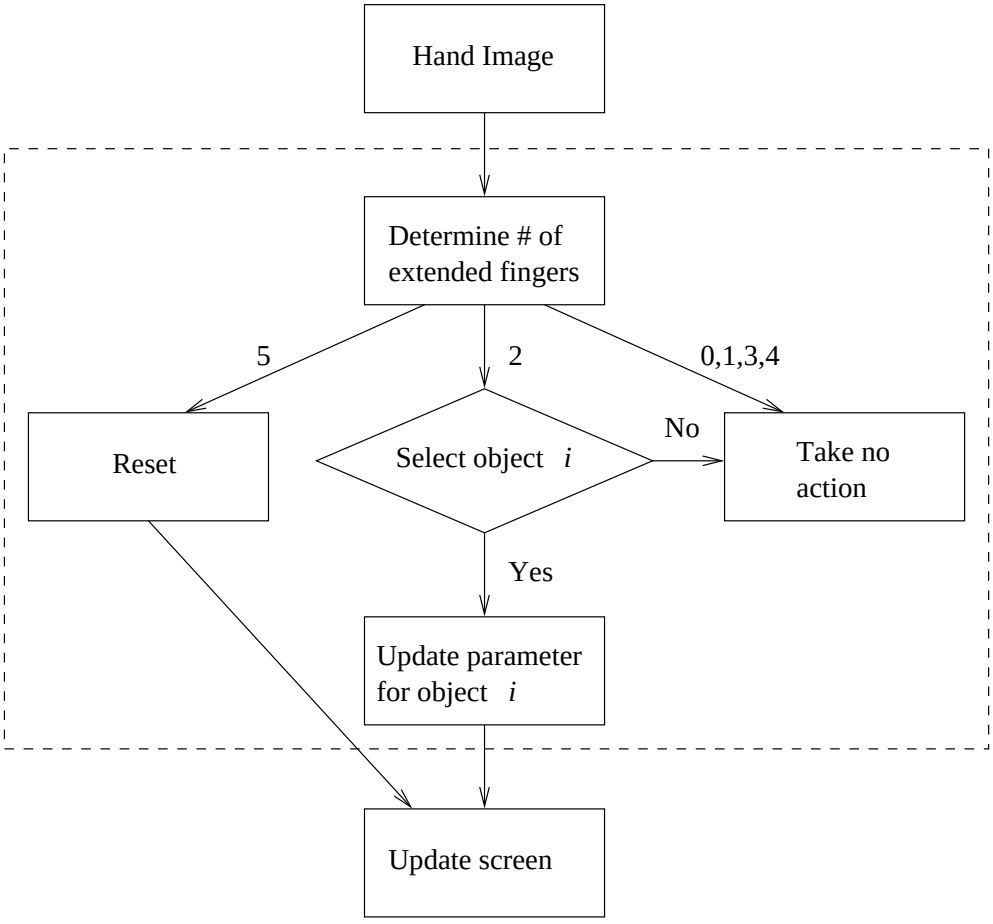


Figure 7.2 The flowchart of the gesture interface.

7.2.3 Results

The results of using this system is shown in this section. Figure 7.3 demonstrate that the algorithm can work for any two fingers that the user finds convenient. The hand can operate from any distance since only the distance between fingertips controls the object movement. In Figure 7.4, an example of simultaneously controlling the object orientation and translation is shown. Figure 7.5 shows that the user can freely control several objects on the screen. When five fingers are present as in the first frame, the object parameters are reset. The same set of objects can also be manipulated by different user during the same session. Since we implemented a very robust and stable fingertip detector, the control and manipulation can be done fairly smoothly and reliably.

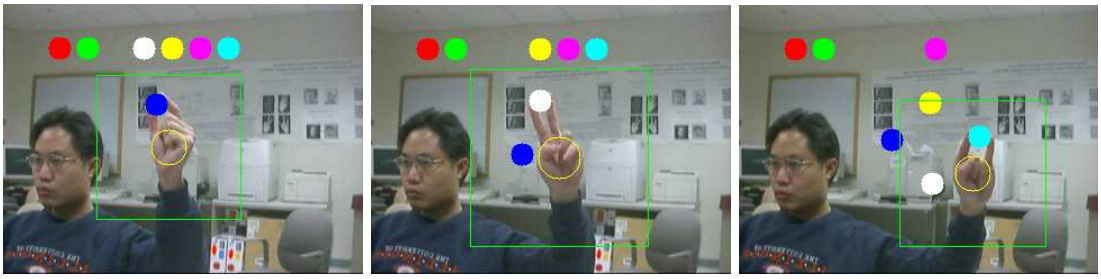


Figure 7.3 The system can be controlled using any two arbitrary fingers that are convenient for the users.

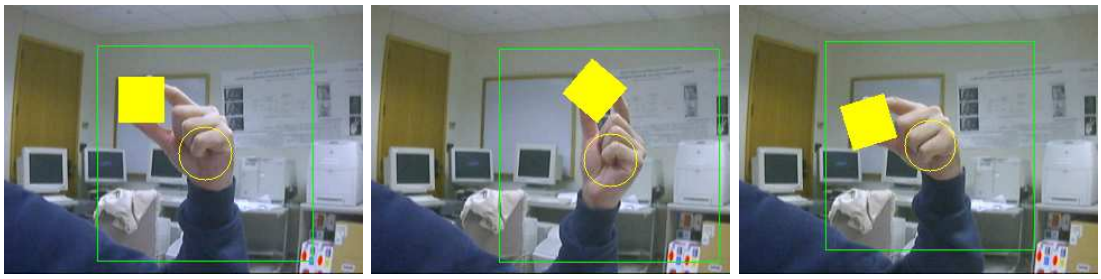


Figure 7.4 An example showing simultaneous control of object orientation and position.

7.3 Virtual Environment Navigation

Terrain rendering and navigation has found applications in many domains such as flight simulator training, video games, virtual tourism, and urban planning. The army can use virtual terrain navigation to help in training, mission planning and rehearsal, and simulations. Civil engineers can

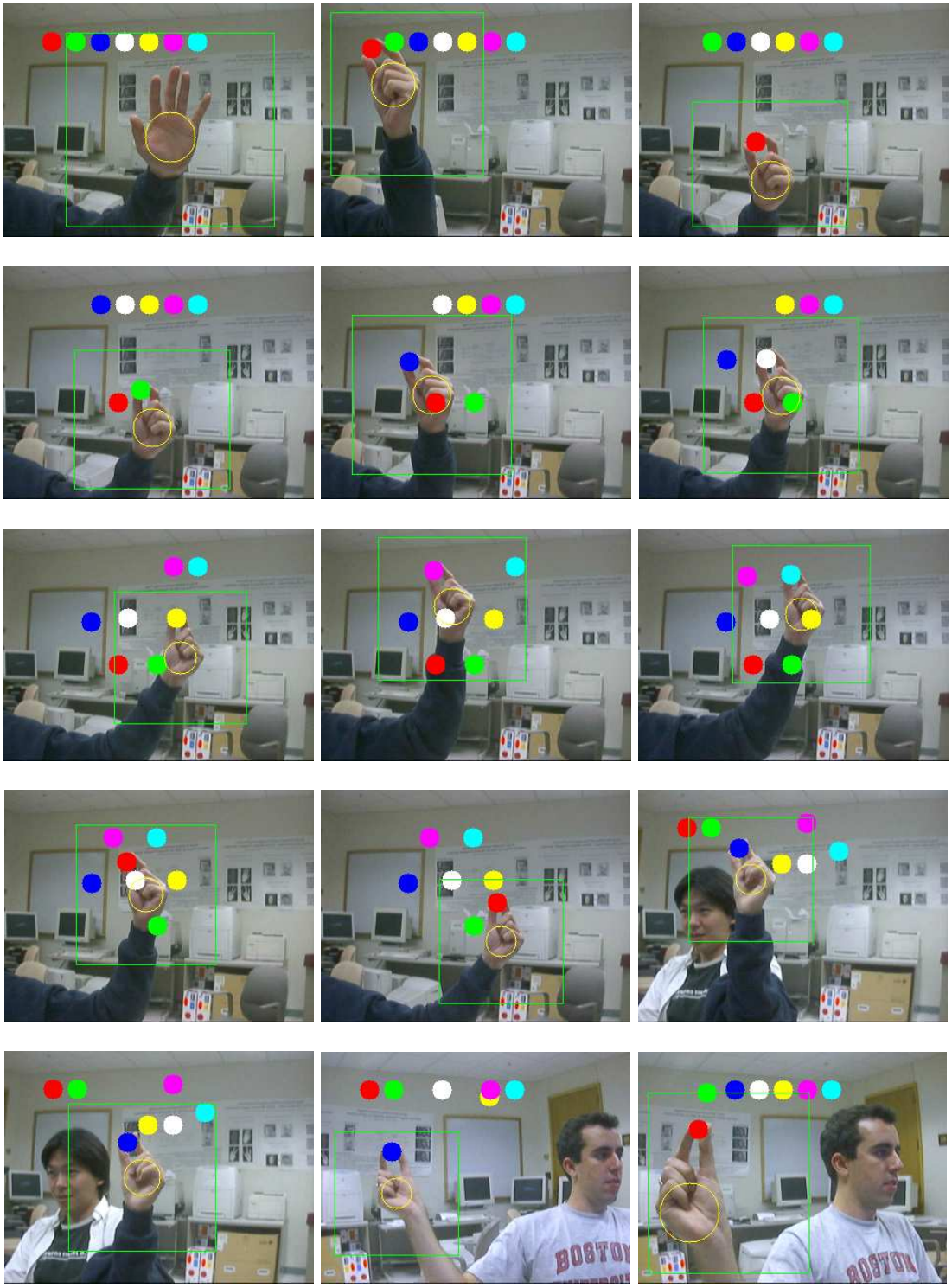


Figure 7.5 An example of reliable object manipulation involving several objects. Different users can also interact with the same set of objects during the same session.

benefit from virtual terrain walkthrough to visualize architectural design and conduct wide scale planning.

To achieve an immersive control in navigation, the user should be allowed to update several motion parameters simultaneously, including yaw, pitch, and roll angles in addition to spatial position. In this scenario, input devices such as the mouse and keyboard will not provide sufficient control. While a wand could achieve the goal, here we show an example that can provide the same equivalence. In this section we describe an efficient implementation of a large terrain visualization system and a navigation framework using gestural input.

7.3.1 Large-scale terrain visualization

With the advances in scanning technology and the constant decrease in the cost of memory, large terrain data sets have become more widely available for researchers and engineers to use in constructing detailed mesh visualization of landscapes. A mesh data set containing millions of polygons is not uncommon nowadays. However, with the current video rendering power, it would be difficult to perform a real-time fly-through of the terrain if every polygon is to be drawn at every frame. Therefore, it is essential to perform fast and convincing approximations of the high resolution mesh at every frame to accommodate the need for a fast display. We follow the approach proposed by Lindstrom and Pascucci [86] which employs a top-down view-dependent refinement scheme to approximate different levels of details. Though extensive work has been devoted to the problem of terrain visualization and level-of-detail management, we choose Lindstrom and Pascucci's algorithm for its desirable features, including independence of error metric, computational efficiency, efficient view culling and triangle stripping, and implementation simplicity. In this section we briefly describe the algorithm.

The goal of the view-dependent refinement is to build a mesh, regardless of a given view, with a small number of triangles that offer a good approximation of the original high-resolution mesh. An alternative approach is the bottom-up simplification which recursively merges pairs of triangles starting from the highest resolution. One main advantage of using top-down refinement is that the refinement complexity is linear in the size of the approximating mesh, whereas the simplification complexity depends on the size of the highest resolution mesh.

The mesh is constructed from points that lie on a regular, rectilinear grid. Using a subdivision algorithm based on longest edge bisection, a hierarchy of mesh structure is constructed. Starting from a square with only four triangles on the most coarse level, each triangle continues to be refined by bisecting the longest edge. Together with a nested error term constructed, a valid mesh is generated at each frame that satisfy the following criteria:

$$j \in M \Longrightarrow i \in M, j \in C_i \tag{7.1}$$

where M is a mesh, i and j are the indices of mesh vertex, and C_i is the set of children of i in the mesh hierarchy. While the error metric can be independent of this algorithm, one property must be enforced:

$$\rho(\delta_i, \mathbf{p}_i, \mathbf{e}) \geq \rho(\delta_j, \mathbf{p}_j, \mathbf{e}), \quad \forall j \in C_i \tag{7.2}$$

in which $\mathbf{p}_i$ is the position of mesh vertex i, ρ is the projected screen space error, δ_i is the object (or world) space error term and $\mathbf{e}$ is the viewpoint.

In [86], a straightforward algorithm for top-down refinement and on-the-fly triangle strip construction is given. The refinement procedure builds a triangle strip $V = (v_0, v_1, v_2, \ldots v_n)$, which represents a sequence of vertices. When implemented using OpenGL [87], `GL_TRIANGLE_STRIP` would speed up rendering as compare to `GL_TRIANGLE`. Also, with this representation a repeated call to `glvertex` is called once for every triangle during rendering.

Because of the hierarchical structure, view culling can be done simultaneously with the refinement process. Using the six planes of the view frustum, we can determine whether a vertex i and the meshes corresponding to the vertices that are descendants of i are visible by the visibility of i. Child is visible only if its parents are due to the nested error terms defined. Consequently, a continuous surface is always guaranteed.

An example of one view of the terrain is shown in Figure 7.6. The part of the terrain surface that is closer to the viewpoint are rendered with more details. The polygons that constitute the surface further away tend to be smaller and can be approximated with one large polygon without much degradation in viewing quality.

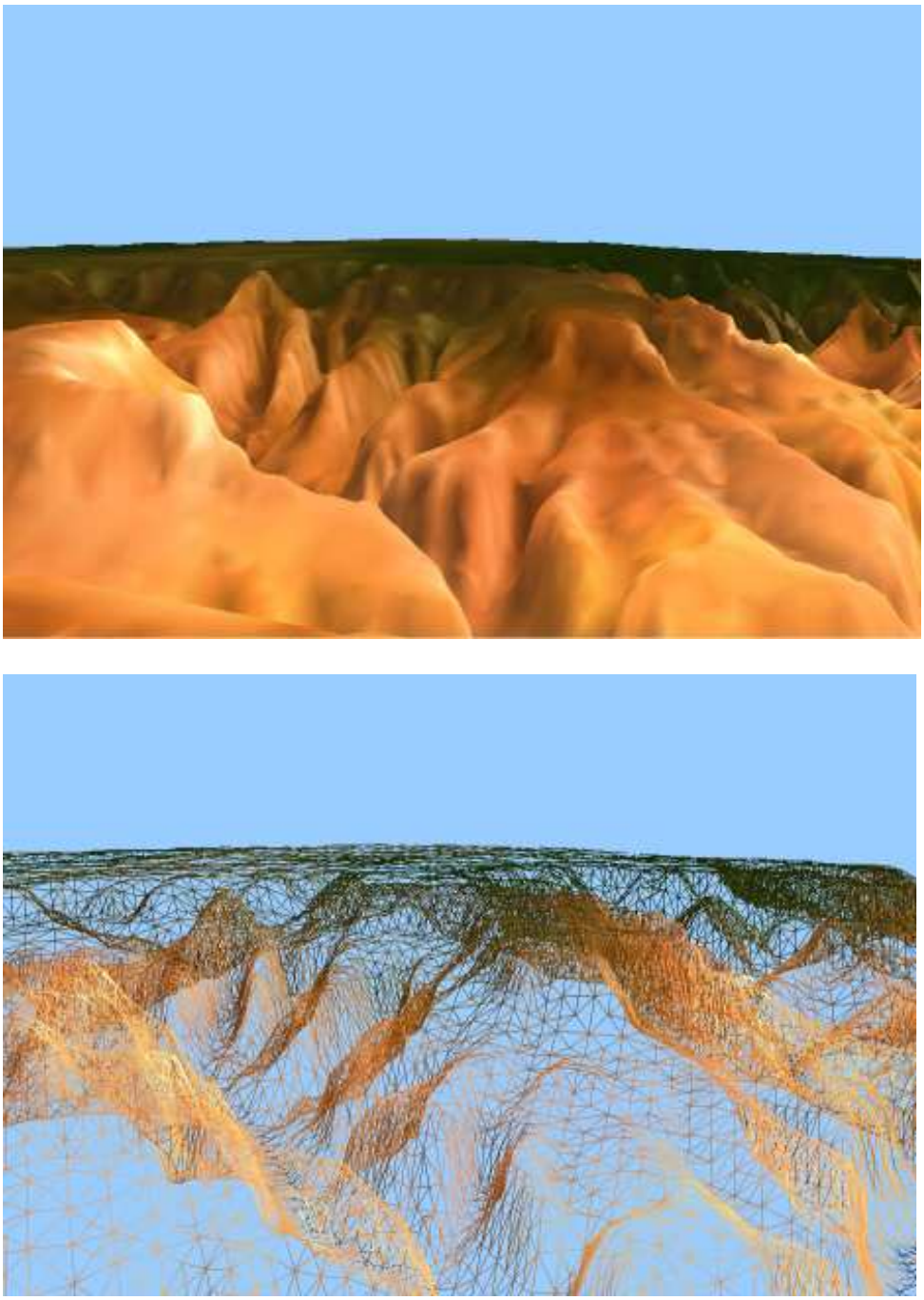


Figure 7.6 (a) A screenshot of the Grand Canyon terrain data set. (b) The corresponding mesh structure.

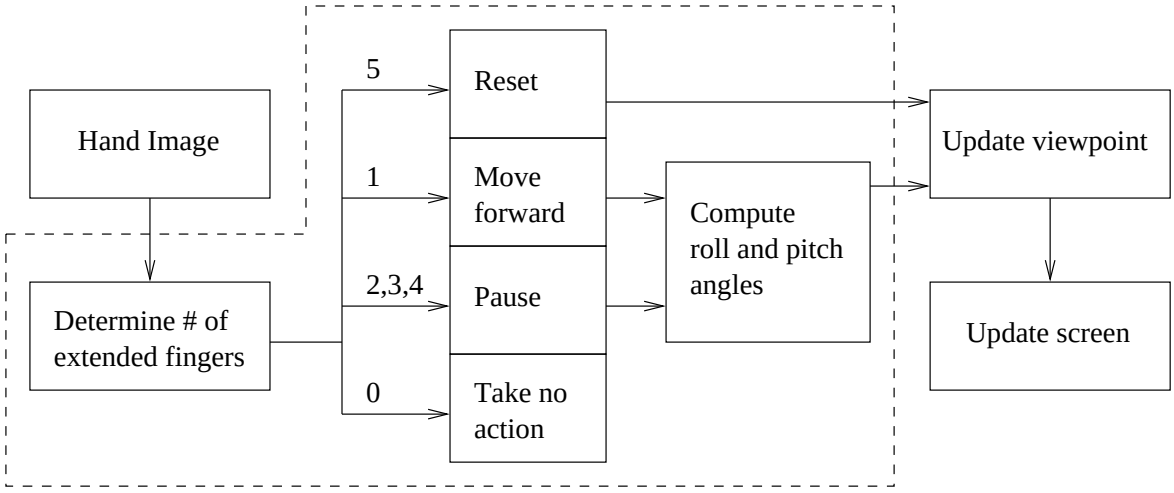


Figure 7.7 The flowchart of the gesture interface implemented for virtual terrain navigation.

7.3.2 Gesture interface

Four modes are defined depending on the number of extended fingers (Figure 7.7):

- *Reset*: Since the size of the terrain mesh is not infinitely large, the user might occasionally be traveling out of boundary. Because of frustum culling, the user will sometimes lose a reference point for returning to the terrain surface. Rather than restarting the program, the user can reset the viewpoint simply by showing all five fingers.

- *Move forward*: When exactly one finger is detected, the viewpoint moves forward to simulate the fly-through. Additionally the orientation of the finger is used to determine the degree of roll angle. The user can conveniently make turns during the navigation simply by pointing the finger in desired direction.

- *Pause*: When the number of fingers detected is between one and five exclusive, then the forward motion is paused. However, the user can still turn left and right using depending on the orientation of the longest finger extended. A convenient usage of such a feature would be to use index finger to control normal fly-through and extend the thumb when a pause in motion is desired to look around the neighboring area.

- *Take no action*: If no extended fingers were detected, then no action is taken. This usually corresponds to the case when user removes his hand away from the camera view.

For `move_forward` and `pause` modes, the roll and pitch angles are computed. Each angle is discretized into a few bins to produce a smoother state output. From our user studies, the exact estimation of the angle value is not necessary to produce a smooth fly-through simulation. The key is to have controls that provide sufficient states of turning left, right, up, and down.

The roll angle is determined by the orientation of the extended finger relative to the x-axis. Each bin corresponds to a mode of rotating speed. When the finger points upward, corresponding to the 90^o bin, the roll angle is zero and only forward motion is triggered. As the finger rotates closer to the x-axis, a larger turning angle is applied to change the viewing angle at each frame. To provide the user with visual feedback of the gesture control, a circle is drawn with the center located at the base of the finger and with radius equal to the length of the finger. The current rotation angle is indicated with a yellow dot, while other available angles are drawn with orange dots along the circle. Some examples of the finger navigation is shown in Figure 7.8

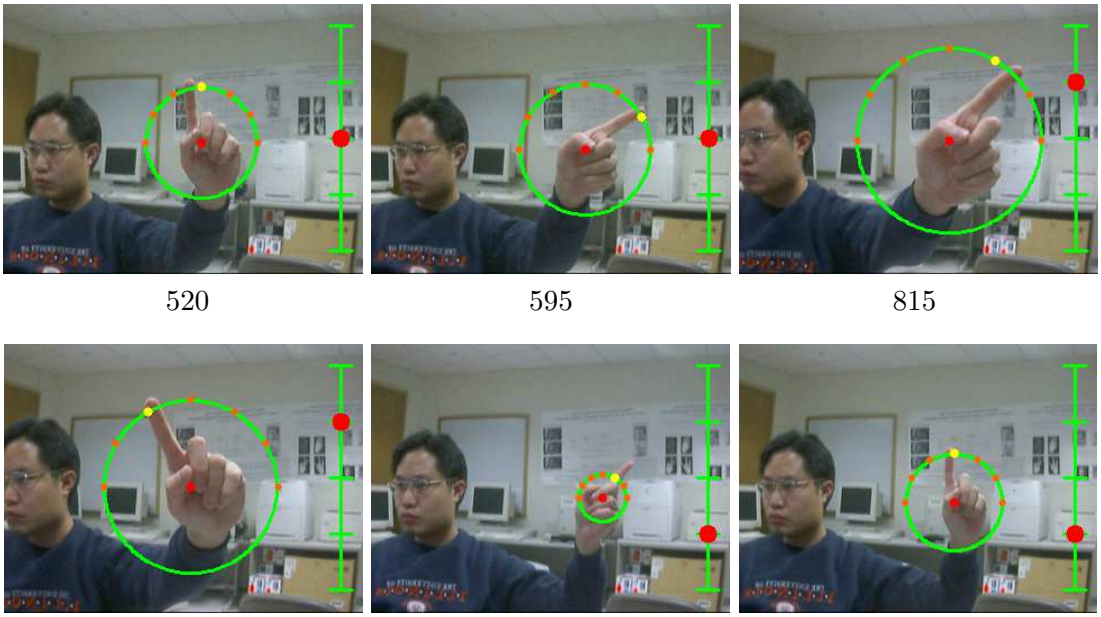


Figure 7.8 The gesture interface for virtual terrain navigation. The frame numbers are shown under each image.

The pitch angle is determined by the size of the palm. When the palm moves closer to the camera, a downward motion is simulated and when the the palm pulls back away from the camera, an upward movement is triggered. This control scheme is similar to the use of joysticks in flight simulator control in which a pull-back action causes the plane to pull up. A bar indicator is shown

on the right of the screen to display the pitch angle (Figure 7.8). We estimated the radius of the palm to be around 20 to 25 pixels for a normal distance a user generally place his hand from the camera. For palm size in this range, the pitch angle is set to 0.

The users can have a more natural control by using the finger itself to estimate the orientation rather than using the fingertip and palm center to determine the orientation. The pointing motion is generally independent of the palm motion and depends on the abduction angle only. Otherwise, using the palm center as a reference point will require a significantly more hand movement to simulate a turning action.

Furthermore, estimating the two angles based on two different hand parts allows the orientation estimation to be relatively insensitive to the palm estimation. One might consider using the fingertip size directly to control the pitch angle, but since it is a much smaller object compared to the palm, a small change in the estimation due to noise will result in a large percentage of change in the size, thus producing unstable controls. For this reason, the palm, being the larger object, is chosen for a more stable control.

Frame 1225 in Figure 7.8 shows an extreme case of a noisy estimation in which the estimated palm size is obviously too small. However, the algorithm can still correctly identify the finger orientation.

7.3.3 Results

A sequence of roughly 1000 frames is recorded and some plots of the estimated orientation angle is shown in Figure 7.9. The pitch angle control is also plotted in Figure 7.10. Though the detection result itself is somewhat noisy, the results show that the discretization produces smoothed control parameter values, and it is sufficient for the purpose of a natural navigation control.

Note that as the size of the palm becomes smaller, it is expected to cause more error and noise for the orientation estimation. This is due to the low resolution of the segmented foreground area which is worsened by the sensor noise. This phenomenon can be observed between frames 1400 and 1600. We also observe most of the flickering in the control state occur near the threshold values that separate two discrete angle values. This happens when the finger transitions from one discrete state to another, and the momentary erratic behavior due to noise is therefore unavoidable.

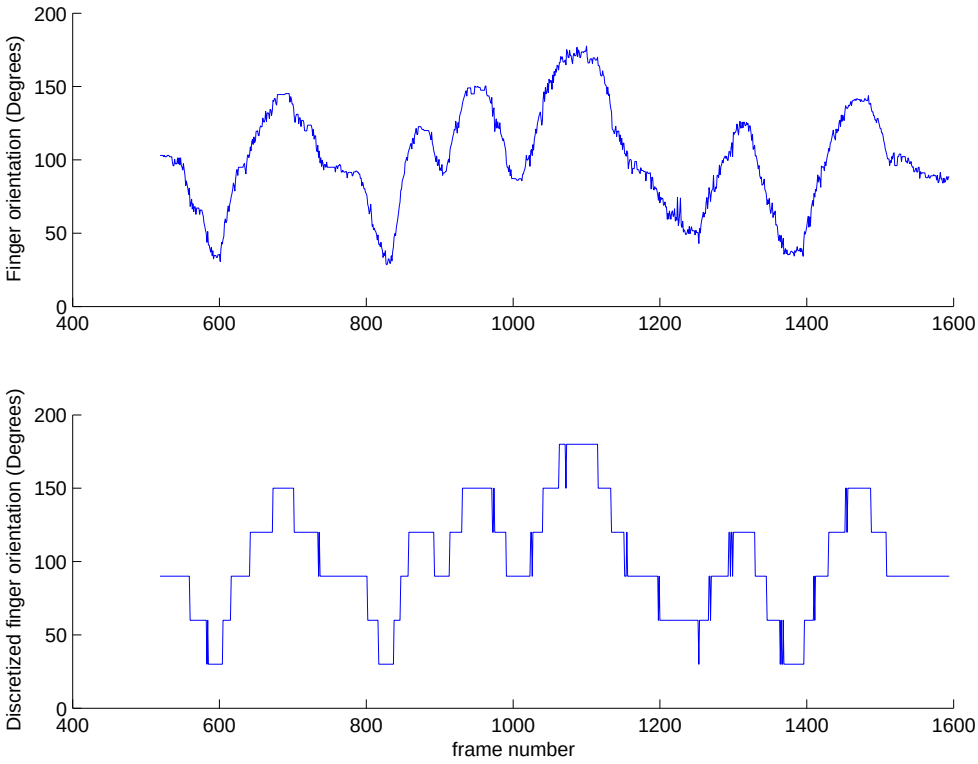


Figure 7.9 Estimated finger orientation and the discretized output.

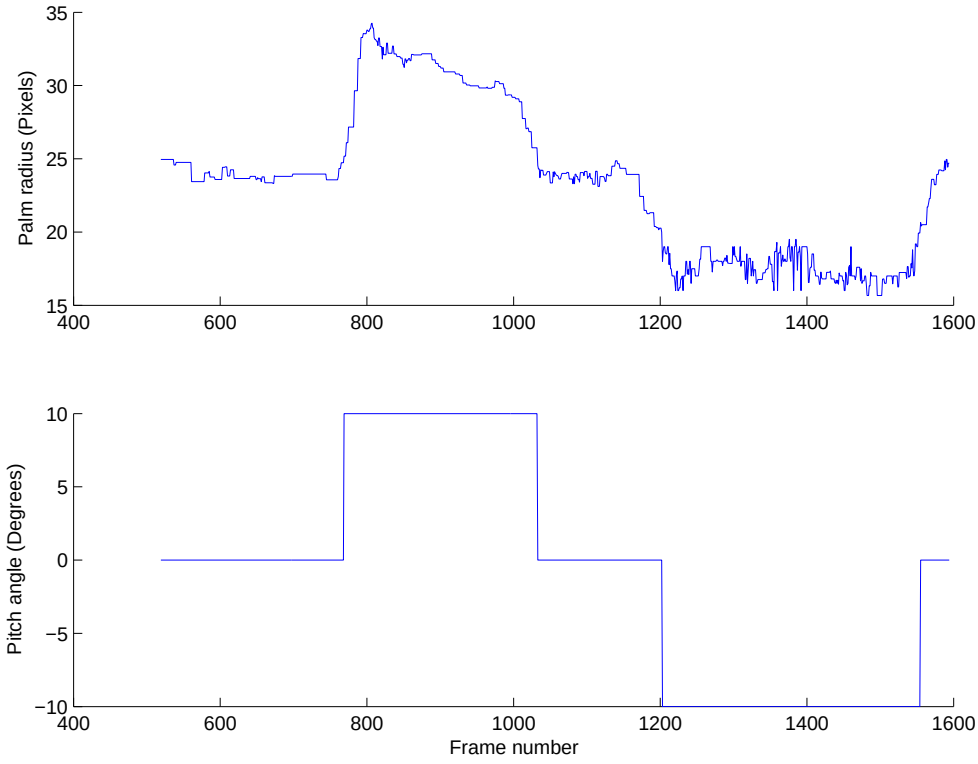


Figure 7.10 Estimated palm size and corresponding pitch angle.

7.4 Finger Drawing

Once the hand and fingertips are reliably detected, one natural application is to simulate mouse pointing and perform finger drawing using one fingertip. This section describes our implementation of one such a system. This application has been implemented in previous work using different fingertip detection algorithms [26, 81, 88]. In most cases, only one fingertip is expected to be present in the scene. While this assumption allows for a more robust algorithm, it limits the possibility of involving different number of gestures to control the application with a more interactive control.

Finger drawing application can be used for AR environments, such as highlighting or selecting an object displayed on a display device. It can also be used for instant messaging in which drawings are transmitted in addition to text messages. In this case, the users are not required to be quipped with special input device for drawing. Any PC camera with reasonable resolution will work. Since the drawing application also records the history of fingertip motion, it can provide an enhanced communication mechanism that is superior than just the fingertip location in the current frame.

7.4.1 Gesture Interface

Four modes are defined for this interface (Figure 7.11):

- *Clear*: When all five fingers are extended and visible, it is interpreted as a reset and clear command. All strokes will be cleared, and the current color is maintained.
- *Draw*: When exactly one finger is detected, the fingertip location at every frame is used to construct a continuous stroke.
- *Pause*: When the number of fingers are between 1 and 5 exclusive, it signals a pause command and the system stops drawing. A new stroke is started when `Draw` command is resumed. This allows the user to start a new stroke at a different location.
- *Take no action*: If no extended finger is detected, then no action is taken.

To further enhance the naturalness in using the interface, when the user's finger moves rapidly, the stroke is automatically disconnected. This is used to avoid the trouble for having to alter the number of fingers every time a new stroke is intended to be created. However, this is not a

very reliable method since if two different strokes are too close to each other, then the distance moved between them within a short time period will not be large enough to be considered for a fast movement and a pause will not be signaled. This becomes especially apparent when the hand is far away from the camera, and takes up a small region. Nevertheless, for a hand located at a reasonable distance, this procedure provides an extra convenience for strokes that are sufficiently far apart. A color palette is also provided on the top of the screen for user to select. The user simply points his finger in the color region and the corresponding color is chosen regardless of the current command state. As a visual feedback, the current color is shown on the upper righthand corner, (Figure 7.12).

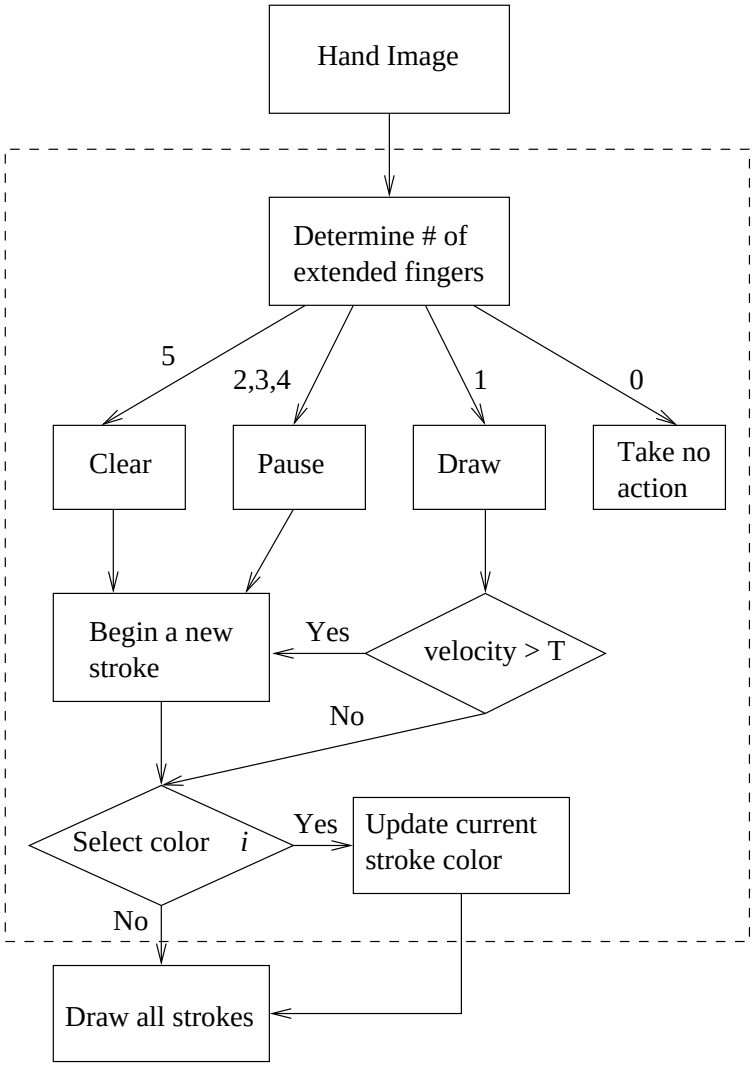


Figure 7.11 The flowchart of the gesture interface implemented for finger drawing.

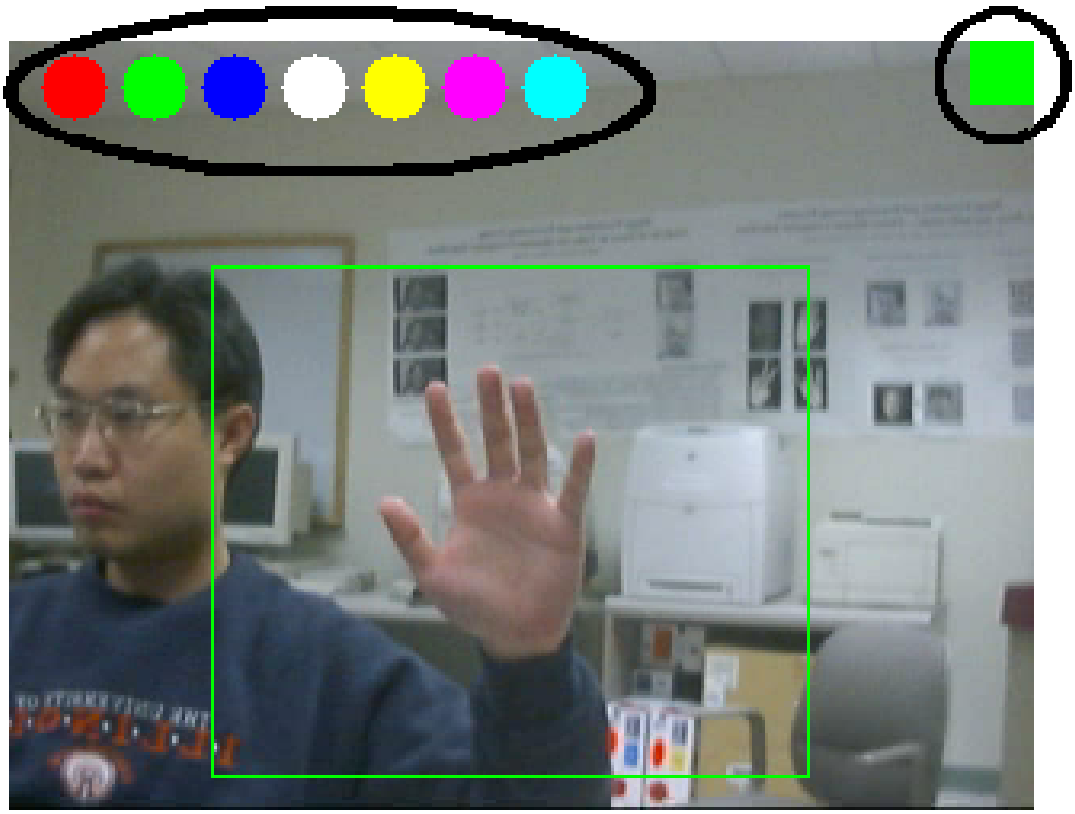


Figure 7.12 The interface for finger drawing. The color palette is located on the top of the screen and the current color is displayed on the upper right hand corner

7.4.2 Results

Some results using this finger drawing interface are shown in this section. In Figure 7.13 we show an instance of users signing their name. Other users also participate in the system testing and they are able to master the interface within a matter of minutes. In Figure 7.14 a more complicated task is shown for drawing a stick figure using different colors for different body parts. With a robust segmentation and detection result, the drawing can be done quiet smoothly.

Figure 7.13 Signing name. The corresponding frame number is shown under each image.

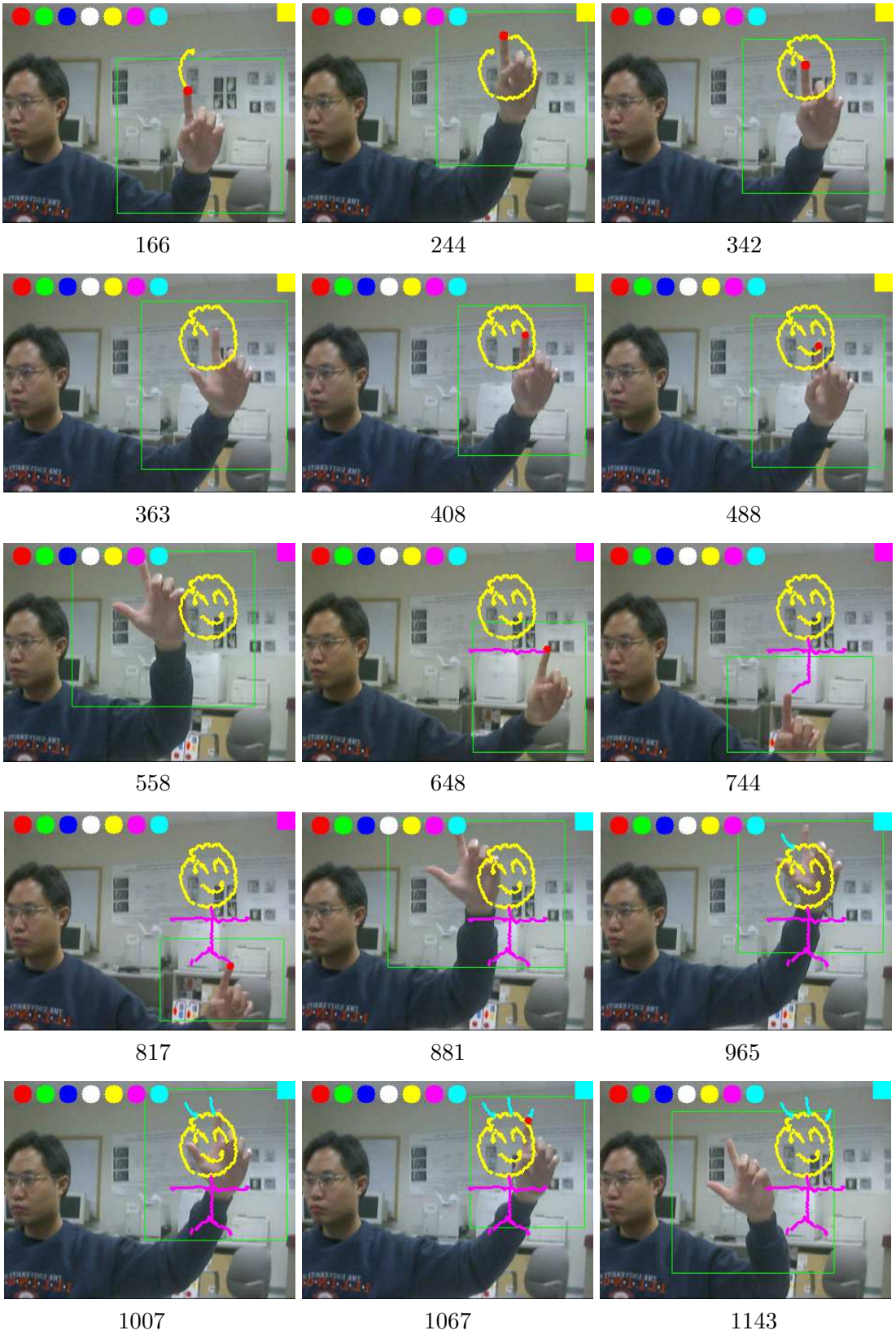


Figure 7.14 Drawing a stick figure. The corresponding frame number is also shown under each image.

7.5 Gesture Recognition

Gesture recognition plays an important role in immersive HCI experience. While there is no known general algorithm that can achieve reliable recognition rate for an arbitrary set of gestures, a small number of specific gesture vocabulary is often sufficient for many HCI tasks. In this section we focus on the scenario of recognizing three gestures: rock, paper, and scissors, inspired from the traditional children's game. The computer can interpret the gesture the user specified through the perceptual interface and a random gesture could be generated to play with the user. Two approaches are experimented for this task. Since the set of gestures involved is relatively small, both of them work quite successfully.

7.5.1 Model-based approach

A rock-paper-scissors recognition scheme can be easily devised by counting the number of fingers detected. The three hand signs rock scissors and paper are each associated with 0-1 finger, 2-3 fingers, and 4-5 fingers, respectively. The performance of the recognition result largely depend on the matching result. With robust finger detection scheme, it can work very reliably by fitting a hand model to the image features to identify number of fingers extended. The recognition results using the model-based approach are shown in Figure 7.15.

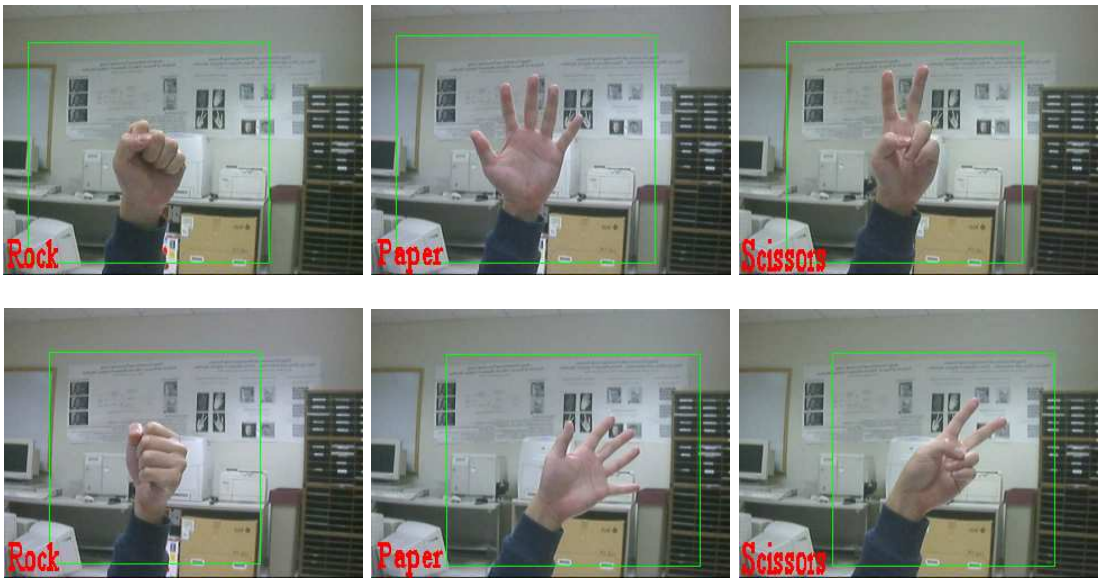


Figure 7.15 Correct recognition results for rock-paper-scissors using the model-based approach.

7.5.2 Appearance-based approach

Yet another approach is to apply an appearance-based approach that could cover a large range of input images and to assume no fingertip detection results. The approach simply trains a mapping from the extracted image features to one of the gesture class which consists of rock, scissors, or paper in this particular case. In this case, we first obtained the gray level image of the segmented hand region and resize it to 100×100. Then PCA is performed, and the top n eigenvector and eigenvalues are retained. Obviously, this is a highly nonlinear classification problem. We use nonlinear support vector machine (SVM) [89] to construct a nonlinear classifier for the three gesture classes. The Gaussian radial basis function is used as the kernel. The hand image is first segmented and resized to 100×100. Some examples of the training images are shown in Figure 7.16.

Figure 7.16 A sample set of training images for SVM.

Some of the successful results of using SVM to perform recognition is shown in Figure 7.17. Since the number of gestures involved in this application is small, the method can work quite reliably. Since the training is done off-line, fast recognition online rate can be achieved.

Figure 7.17 Correct recognition results for rock-paper-scissors using the appearance-based approach.

7.6 Discussions

In this chapter we present several applications using the hand gestures as the control input including gesture recognition, object manipulation in AR environment, virtual terrain navigation, and finger drawing. The use of gestural input in place of mouse and keyboard input is shown to be a more expressive method because of the high degrees of freedoms allowed. One key design issue involved in all gesture interface is the definition of a gesture command set that is easy for users to adapt to and intuitive to use. The commands we defined in our applications are rather intuitive and most users can master them in a matter of minutes. The number of gestures we used are quite small for each application, and they are all very natural to perform. However, further studies should be devoted to the development of a set of grammar that is easy to learn and intuitive to use. With a set of more complex grammar, we can perform a more immersive and complicated motion. For instance, in the case of virtual environment navigation, we could perform additional tasks such as manipulating objects at the same time by having a gesture to switch between menu mode, navigation mode, or manipulation mode. Or we could have a set of gestures that directly trigger commands in each mode without switching between them.

Another extension is to allow the presence of more than one hands. One appropriate setup would be to have the camera looking down from the top with hands moving on a desk. This setup avoid the confusion caused by the face, which would be a problem whether using color segmentation or infrared segmentation. The segmentation can also be better controlled this way by selecting a uniform color for the desktop. One such application is built in [90], which was intended for augmented reality. In their application, the left hand is used for menu function, offering a more

variety of actions. A further extension of having multiple hands is to allow more than one user to interact simultaneously on a desk such as for the case in military simulation and rehearsal or for architects exchanging ideas on landscape design.

The graphics presentation can also be improved by exploiting other graphics libraries to make better renderings. Currently, the virtual objects shown overlapping the video sequences are drawn using OpenCV functions which are intended for 2D drawings. With a more powerful graphics library like OpenGL, we could render 3D objects with proper lighting conditions to better visualize the feedback.

CHAPTER 8

CONCLUSIONS

8.1 Summary

With the rapid advancement in computing technology, new functionalities of the computers have emerged, changing the traditional communication and display technology that we have been familiar with. Using a monitor, a mouse, and a keyboard is no longer the only way to interact with the computer, and a more natural HCI interaction is desired. Visual gesture interface offers such an alternative. In this dissertation we address several issues involved in making a real-time hand tracking system realizable.

In Chapter 2 we give an overview of the hand tracking problem. The model-based hand tracking approach could potentially provide an accurate estimate given a good initialization. However, to estimate the current hand posture, the high DOF of the hand prohibits exhaustive search for real-time processing. The LOD of the 3D hand model also affects the processing speed.

In Chapter 3 we discuss various motion constraints available for reducing the tracking complexity. We propose two representations to model the feasible configuration space, and for each representation, a corresponding tracking algorithm is designed in Chapters 4 and 5. The first approach uses a semiparametric representation to compactly model many of the feasible natural motions. Although it is sufficient for many common hand motions, it is difficult to be generalized for arbitrary motion. A nonparametric model is proposed to address this issue. By using the discrete configurations collected directly from the real motion data to represent the entire feasible space, the tracker is capable of tracking arbitrary motions. To search in this discrete space, we designed the SNM search algorithm, which can also be generalized to track other objects.

Model-based hand tracking has been a difficult and computation-intensive problem. Most research effort has been focused on the tracking itself and little research addresses the issue of initialization. To fully automate the system, we present a real-time algorithm for learning the initial model geometry and motion parameter in Chapter 6. Based on the detection results, we are able to build several robust and real-time visual gesture interfaces, as shown in Chapter 7.

8.2 Future Directions

A robust visual hand tracker can be utilized as an important input module in a natural HCI interface. The advantage of having multidimensional input using the hand gestures is accompanied by the problem of estimating the articulated hand with high DOF. It was common in previous studies to reduce the problem complexities by introducing the hand motion constraints. While many of the constraints are simple and effective in reducing the dimensionality of configuration space, we are interested in identifying more constraints that does not have simple closed form expressions.

In Chapter 3 we propose a discrete representation to model the feasible configuration space. Currently this model does not incorporate any motion dynamics in the structure. We should study how we can embed the motion dynamics in the searching to help improving tracker performance while at the same time maintain the flexibility of using the model to allow capture of arbitrary motions. Another issue we should look into is the study of the collected motion data itself. We should design a method to identify the most representative data set, which should also be user-independent.

The NM simplex search works well in practice for many optimization problems. Its features are especially suitable for the case of visual tracking where the objective function is highly nonlinear. However, it is known to have no theoretical analysis for its convergence. We should perform additional experiments to study the performance of NM algorithm in tracking the hand motion and other general tracking applications in different domain.

Another problem that plagues many visual tracking system is the definition of a discriminant matching function that should output a distinctive peak at the exact match and result in as little false alarms as possible. Using edge and silhouette along can result in ambiguities when the back-

ground is cluttered. Using a combination of features have been reported to produce a more robust discrimination power. More effort should be devoted to design a better set of features and a proper matching function for hand tracking.

One extension can be made to improve the robustness of the tracker by combining model-based and appearance-based approaches. Assuming the model geometry is known, the additional cue provided by the appearance-based approach can be served in a re-initialization stage to recover the system from losing track or to correct the tracker estimation in case of accumulating errors.

One simplified scenario for re-initialization is to require the user to show the frontal view of the palm with all fingers extended. Based on this assumption, we propose an algorithm not only for automatically initializing the motion parameters but also for simultaneously learning the model geometry. A 3D model is learned based on the 2D information using a bottom-up approach. Although the current approach is quite robust in determining the location of palm and fingertips, it is somewhat noisy for estimating the fine dimension of model geometry such as the width and length of the finger. One possible remedy is to combine top-down and bottom-up approaches in an iterative manner to refine the estimation.

Finally, based on the fingertip and palm detection algorithms introduced in Chapter 6, we are able to build many interesting applications using a visual gesture interface. Currently, we use a small set of gestures for each application to allow the users to quickly master the interface. While the robustness of the detection algorithm allows us to design a more complicated gesture grammar, we need to be careful about selecting the proper subset that is simple for users to adapt to and natural to perform.

REFERENCES

[1] Y. Wu and T. S. Huang, "Hand modeling, analysis and recognition for vision-based human computer interaction," *IEEE Signal Processing Magazine*, vol. 18, pp. 51–60, May 2001.

[2] V. Pavlović, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human computer interaction: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 677–695, July 1997.

[3] J. Martin, V. Devin, and J. L. Crowley, "Active hand tracking," in *Proc. of IEEE Third Int. Conf. on Automatic Face and Gesture Recognition*, 1998, pp. 573–578.

[4] Y. Wu and T. S. Huang, "Vision-based gesture recognition: A review," in *Gesture-Based Communication in Human-Computer Interaction*, A. Braffort, R. Gherbi, S. Gibet, J. Richardson, and D. Teil, Eds. Lecture Notes in Artificial Intelligence 1739, Berlin: Springer-Verlag, 1999, pp. 93-104.

[5] C. Colombo, A. Del Bimbo, and A. Valli, "Visual capture and understanding of hand pointing actions in 3-D environment," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 33, no. 4, pp. 677–686, Aug. 2003.

[6] T. Starner, J. Weaver, and A. Pentland, "A wearable computer based American sign language recognizer," in *Proc. of IEEE Int'l Symposium on Wearable Computing*, 1997, pp. 130–137.

[7] Y. Azoz, L. Devi, and R. Sharma, "Reliable tracking of human arm dynamics by multiple cue integration and constraint fusion," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 905–910.

[8] Y. Wu and T. S. Huang, "Robust visual tracking by co-inference learning," in *Proc. of IEEE Int. Conf. on Computer Vision*, vol. II, 2001, pp. 26–33.

[9] S. Lu, D. Metaxas, D. Samaras, and J. Oliensis, "Using multiple cues for hand tracking and model refinement," in *Proc. IEEE Int. Conf. on Computer Vision*, vol. II, 2003, pp. 443–450.

[10] R. Grzeszcuk, G. Bradski, M. H. Chu, and J.-Y. Bouguet, "Stereo based gesture recognition invariant to 3D pose and lighting," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, 2000, pp. 826–833.

[11] M. Swain and D. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.

[12] R. Kjeldsen and J. Kender, "Finding skin in color images," in *Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 312–317.

[13] M. Fleck, D. Forsyth, and C. Bregler, "Finding skin in color images," in *Proc. European Conference on Computer Vision*, vol. II, 1996, pp. 592–602.

[14] X. Zhu, J. Yang, and A. Waibel, "Segmenting hands of arbitrary color," in *Proc. Fourth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 2000, pp. 446–453.

[15] L. Sigal, S. Sclaroff, and V. Athitsos, "Estimation and prediction of evolving color distribution for skin segmentation under varying illumination," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 152–159.

[16] D. Comaniciu and P. Meer, "Robust analysis of feature space: Color image segmentation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1997, pp. 750–755.

[17] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1999, pp. 274–280.

[18] Y. Raja, S. J. McKenna, and S. Gong, "Colour model selection and adaptation in dynamic scenes," in *Proc. European Conf. on Computer Vision*, 1998, pp. 460–474.

[19] T. Kohonen, "Self-organized formation of topological correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59–69, 1982.

[20] Y. Wu, Q. Liu, and T. S. Huang, "An adaptive self-organizing color segmentation algorithm with application to robust real-time human hand localization," in *Proc. Asian Conf. on Computer Vision*, 2000, pp. 1106–1111.

[21] G. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, 1998, http://developer.intel.com/technology/itj/q21998/articles/art_2.htm.

[22] Y. Wu and T. S. Huang, "View-independent recognition of hand postures," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. II, 2000, pp. 88–94.

[23] J. Lee and T. Kunii, "Model-based analysis of hand posture," *IEEE Computer Graphics and Applications*, vol. 15, pp. 77–86, Sept. 1995.

[24] C. Noelker and H. Ritter, "Visual recognition of continuous hand postures," *IEEE Trans. Neural Networks*, vol. 13, no. 4, pp. 983–994, 2002.

[25] J. Davis and M. Shah, "Visual gesture recognition," *Vision, Image, and Signal Processing*, vol. 141, pp. 101–106, 1994.

[26] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," *IEEE Transactions on Computer Graphics and Applications*, vol. 22, pp. 64–71, Nov/Dec 2002.

[27] C. von Hardenberg and F. Bérard, "Bare-hand human-computer interaction," in *Proc. ACM Workshop on Perceptive User Interfaces*, Orlando, FL, 2001, pp. 1–8.

[28] J. Rehg and T. Kanade, "Model-based tracking of self-occluding articulated objects," in *Proc. of IEEE Int. Conf. Computer Vision*, 1995, pp. 612–617.

[29] V. Athitsos and S. Sclaroff, "Estimating 3D hand pose from a cluttered image," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. II, 2003, pp. 432–439.

[30] N. Shimada, K. Kimura, and Y. Shirai, "Real-time 3-D hand posture estimation based on 2-D appearance retrieval using monocular camera," in *Proc. of Int. WS. RATFG-RTS*, 2001, pp. 23–30.

[31] A. Imai, N. Shimada, and Y. Shirai, "3-D hand posture recognition by training contour variation," in *Proc. of the Sixth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 2004, pp. 895–900.

[32] E.-J. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," in *Proc. of the Sixth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 2004, pp. 889–894.

[33] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, June 2003, pp. 127–133.

[34] R. Rosales, S. Sclaroff, and V. Athitsos, "3D hand pose reconstruction using specialized mappings," in *Proc. of IEEE Int. Conf. on Computer Vision*, Vancouver, Canada, July 2001, pp. 378–387.

[35] M. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. on Information Theory*, vol. IT-8, pp. 179–187, 1962.

[36] J. Lin, Y. Wu, and T. S. Huang, "Capturing human hand motion in image sequences," in *Proc. of IEEE WMVC*, 2002, pp. 99–104.

[37] J. Lin, Y. Wu, and T. S. Huang, "3D model-based hand tracking using stochastic direct search method," in *Proc. of the Sixth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 2004, pp. 693–698.

[38] J. J. Kuch and T. S. Huang, "Vision-based hand modeling and tracking for virtual teleconferencing and telecollaboration," in *Proc. of IEEE Int. Conf. on Computer Vision*, Cambridge, MA, June 1995, pp. 666–671.

[39] B. Stenger, P. R. S. Mendonça, and R. Cippola, "Model-based 3D tracking of an articulated hand," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. II–310–II–315.

[40] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura, "Hand gesture estimation and model refinement using monocular camera - ambiguity limitation by inequality constraints," in *Proc. of the Third Conf. on Face and Gesture Recognition*, 1998, pp. 268–273.

[41] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cippola, "Filtering using a tree-based estimator," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 1063–1070.

[42] Y. Wu, J. Lin, and T. S. Huang, "Capturing natural hand articulation," in *Proc. of IEEE Int. Conf. on Computer Vision*, vol. II, 2001, pp. 426–432.

[43] Y. Wu and T. S. Huang, "Capturing articulated human hand motion: A divide-and-conquer approach," in *Proc. of IEEE Int. Conf. Computer Vision*, 1999, pp. 606–611.

[44] J. Segen and S. Kumar, "Shadow gesture: 3D hand pose estimation using a single camera," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1999, pp. 479–485.

[45] T. Heap and D. Hogg, "Towards 3D hand tracking using a deformable model," in *Proc. of IEEE Int. Conf. Automatic Face and Gesture Recognition*, Killington, VT, 1996, pp. 140–145.

[46] C.-C. Chang and W.-H. Tsai, "Model-based analysis of hand gestures from single images without using marked gloves or attaching marks on hands," in *Proc. of Fourth Asian Conf. on Computer Vision*, vol. 2, 2000, pp. 923–930.

[47] J. Kramer, M. Yim, L. Edwards, A. Boronkay, and J. Tian, *VirtualHand Software Library Reference Manual*, Virtual Technologies, Inc., 1998.

[48] M. Verleysen, "Learning high-dimensional data," in *Limitations and Future Trends in Neural Computation*, S. Ablameyko et al., Eds. Amsterdam: IOS Press, 2001, pp. 141-162.

[49] J. Tenenbaum, "Mapping a manifold of perceptual observations," *Advances in Neural Information Processing*, vol. 10, pp. 682–688, 1998.

[50] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, Dec. 22, 2000.

[51] F. W. Young and R. M. Hamer, *Multidimensional Scaling: History, Theory and Applications*, New York: Erlbaum, 1987.

[52] R. Basri, D. Roth, and D. Jacobs, "Clustering apperances of 3D objects," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 414–420.

[53] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, "Clustering appearancs of objects under varying illumination conditions," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, 2003, pp. 11–18.

[54] A. Elgammal and C.-S. Lee, "Inferring 3D body pose from silhouettes using activity manifold learning," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

[55] M. Brand, "Shadow puppetry," in *Proc. of IEEE Int. Conf. on Computer Vision*, 1999, pp. 1237–1244.

[56] A. Heap and D. Hogg, "Wormholes in shape space: Tracking through discontinuous changes in shape," in *Proc. of IEEE Int. Conf. Computer Vision*, 1998, pp. 344–349.

[57] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Transactions of Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[58] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.

[59] E. A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. of IEEE Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control*, Cambridge, UK, 2000, pp. 343–356.

[60] E. A. Wan and R. van der Merwe, "The unscented kalman filter," in *Kalman Filtering and Neural Networks*, S. Haykin Ed. New York: Wiley Publishing, 2001, pp. 221-280.

[61] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and non-Gaussian Bayesian state estimation," in *IEE Proceedings-F*, vol. 140, 1993, pp. 107–113.

[62] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *Proc. of European Conf. on Computer Vision*, Cambridge, UK, 1996, pp. 343–356.

[63] A. Blake and M. Isard, *Active Contours*, London: Springer-Verlag, 1998.

[64] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," in *IEE Proc. on Radar and Sonar Navigation*, vol. 146, 1999, pp. 2–7.

[65] M. Isard and A. Blake, "Icondensation: Unifying low-level and high-level tracking in a stochastic framework," in *Proc. of ECCV*, 1998, pp. 767–781.

[66] A. Doucet, N. G. de Freitas, and N.J. Gordon, *Sequential Monte Carlo Methods in Practice*, New York: Springer-Verlag, 2001.

[67] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *J. Amer. Statist. Assoc.*, vol. 93, pp. 1032–1044, 1998.

[68] G. Kitagawa, "Monte Carlo filter and smoother for Non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[69] F. H. Walters, L. R. Parker, S. L. Morgan, and S. N. Deming, *Sequential Simplex Optimization*, Boca Raton: CRC Press, 1991.

[70] D. P. Bertsekas, *Nonlinear Programming*, Nashua: Athena Scientific, 1999.

[71] F. Aurenhammer and R. Klein, *Handbook of Computational Geometry*, Amsterdam: North-Holland, 2000.

[72] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed. Berlin: Springer, 2000.

[73] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proc. of the Thirtieth Annual ACM Symposium on Theory of Computing*, 1998, pp. 604–613.

[74] T.-J. Cham and J. M. Rehg, "A multiple hypothesis approach to figure tracking," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1999, pp. 239–245.

[75] S. Ahmad, "A usable real-time 3D hand tracker," in *Proc. IEEE 28th Asilomar Conference on Signals, Systems and Computers*, 1995, pp. 1257–1261.

[76] N. Krahnstoever and R. Sharma, "Articulated models from video," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 894 – 901.

[77] R. Zunkel, "Hand geometry based verification," in *Biometrics: Personal Identification in Networked Society*, A. Jain et al., Eds. Boston: Kluwer Academic, 1998, pp. 857-861.

[78] A. K. Jain, A. Ross, and S. Pankanti, "A prototype hand geometry-based verification system," in *Proc. of 2nd Int'l Conf. on Audio- and Video-Based Biometric Person Authetication*, 1999, pp. 166–171.

[79] A. K. Jain and N. Duta, "Deformable matching of hand shapes for verification," in *Proc. of Int'l Conf. on Image Processing*, 1999, pp. 857–861.

[80] J. Segen and S. Kumar, "Look ma, no mouse!," *Communications of the ACM*, vol. 43, pp. 102–109, July 2000.

[81] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer, "Visual panel: Virtual mouse keyboard and 3D controller with an ordinary piece of paper," in *Proc. of Workshop on Perceptive User Interfaces*, 2001, pp. 1–8.

[82] P. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, 2004, pp. 511–518.

[83] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. of ACM Siggraph*, vol. 18, 1984, pp. 207–212.

[84] Y. Wu, K. Toyama, and T. S. Huang, "Self-supervised learning for object recognition based on kernel discriminant-em algorithm," in *Proc. of IEEE Int. Conf. on Computer Vision*, Vancouver, Canada, vol. I, July 2001, pp. 275–280.

[85] B. Buchholz, T. J. Armstrong, and S. A. Goldstein, "Anthropometric data for describing the kinematics of the human hand," *Ergonomics*, vol. 35, pp. 261–273, 1992.

[86] P. Lindstrom and V. Pascucci, "Visualization of large terrains made easy," in *Proc. of IEEE Visualization*, 2001, pp. 363–370, 574.

[87] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL Programming Guide*, 3rd ed., Reading: Addison Wesley, 2000.

[88] N. Ukita, A. Terabe, and M. Kidode, "Wearable virtual tablet: Fingertip drawing interface using an active-infrared camera," *Information Processing Society of Japan Journal*, vol. 45, no. 3, pp. 977–990, 2004.

[89] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[90] K. Oka, Y. Sato, and H. Koike, "Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems," in *Proc. of the IEEE Int'l Conf. Automatic Face and Gesture Recognition*, 2002, pp. 429–434.

VITA

John Lin received his B.S. degree in electrical engineering with a minor in computer science in 1998 and an M.S. in electrical engineering, under the direction of Prof. Thomas S Huang, in 2000, all from the University of Illinois at Urbana-Champaign. In 1999, he joined the Image Formation and Processing Group at the Beckman Institute, University of Illinois at Urbana-Champaign, as a graduate research assistant. He was a research intern with the Mitsubishi Electric Research Lab and IBM T.J. Watson Research Center during summer 2001 and 2002, respectively.

His primary research interests focus on computer vision, computer graphics, statistical learning, tracking articulate hand motions, and vision-based human computer interactions.