VISION AND LEARNING FOR INTELLIGENT HUMAN-COMPUTER INTERACTION

BY

YING WU

B.E., Huazhong University of Science and Technology, 1994
M.E., Tsinghua University, 1997

THESIS

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, 2001

Urbana, Illinois

ABSTRACT

It was a dream to make computers see. The research in computer vision provides promising technologies to capture, analyze, transmit, retrieve and interpret visual information. However, due to the richness and large variations in the visual inputs, the practice of many statistical learning techniques for visual motion capturing and recognition are confronted by some similar problems, such that making intelligent and visually capable machines is still a challenging task. This dissertation concentrates on two important problems: capturing and recognizing human motion in video sequences, which are crucial for the research and applications of intelligent human computer interaction, multimedia communication, and smart environments.

This dissertation presents three effective techniques for visual motion analysis tasks: non-stationary color model adaptation for efficient localization, multiple visual cues integration for robust tracking, and learning motion models for capturing articulated hand motion. Besides, this dissertation describes a novel statistical learning method, the Discriminant-EM (D-EM) algorithm, in the framework of self-supervised learning paradigm. D-EM employs both labeled and unlabeled training data and converges supervised and unsupervised learning. Many topics in the dissertation is unified by the four problems of self-supervised learning, i.e., *transduction*, *co-transduction*, *model transduction* and *co-inferencing*. Extensive experiments and two prototype systems have validated the proposed approaches in the domain of vision-based human computer interaction.

To my parents and to Jindan

ACKNOWLEDGMENTS

Above all, I would like to express my sincere thanks to my advisor Professor Thomas S. Huang for his insightful guidance, enlightening advice, and endless encouragement throughout my Ph.D. study, which has given me a great opportunity to explore various difficult but interesting problems. I was lucky and am proud of being a student of him, a great man with extraordinary vision and wisdom. Especially, I would like to thank my two mentors in Microsoft Research, Dr. Kentaro Toyama and Dr. Zhengyou Zhang for their selfless discussions and suggestions, without which I could have not made this work possible. I would also like to thank my Ph.D. advisory committee members Professor Narendra Ahuja, Professor David Kriegman, and Dr. Kentaro Toyama, for their inspiring and constructive discussions during my study.

I also would like to thank all my colleagues in the Image Formation and Processing Group and many of my friends in Microsoft Research. In particular, I would like to thank Dr. Steve Shafer, Dr. Ying Shan, Dr. Harry Shum, Dr. John Krumm, Dr. Rick Szeliski, Erik Hanson, Dr. Vladimir Pavlovic, Greg Berry, Dr. Nebojsa Jojic, Dr. Qiong Liu, John Y. Lin, Qi Tian, Sean Xiang Zhou, and Yunqiang Chen. Special thanks to John Y. Lin for his hard work of collecting finger motion data and his selfless help on finger tracking experiments and paper proofreading.

I wish to thank my family for all their endless love, support and encouragement though all the time of my study abroad. Finally, but not least, I would like to express my deep thanks to my dear wife, Jindan, for all her love, sacrifice, understanding and help, which could be felt in every word in this work.

TABLE OF CONTENTS

CHAPTER **PAGE**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 Virtual environments

In recent years, virtual reality technologies have taken us into three-dimensional (3-D) virtual worlds and pushed us to develop a new concept of human-computer interaction.

The Electronic Visualization Lab (EVL) at the University of Illinois at Chicago and the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign have developed a family of virtual environments (VEs), including CAVE, Infinity-Wall, ImmersaDesk, HIVE, and PARIS. These VEs aim to give users the feeling of immersion, allowing them to explore 3-D virtual worlds.

In the evolution of user interfaces, keyboards were the primary devices in text-based user interfaces, and then the invention of the mouse brought us the graphical user interface. What is the counterpart of the mouse when we are trying to explore 3-D virtual environments (VEs)?

In many current VE applications, keyboards, mice, wands, and joysticks are the common controlling and navigating devices. However, to some extent, such mechanical devices are inconvenient and unsuitable for natural and direct interaction, since it is difficult for these devices to provide 3-D and high degrees of freedom inputs. In many CAVE-like VEs, 3-D inputs are obtained from magnetic sensors, which are prone to magnetic interference and unable to give the feeling of immersion due to cable connections, such that natural interaction could not be achieved.

1.1.2 Human-computer interaction

Traditionally, a main task in human-computer interaction (HCI) focused on designs of machineries for computers. However, although some simple controlling tasks could be fulfilled, interacting with 3-D VEs through these machineries is not the most appropriate way. A more intuitive concept for interaction is needed.

Obviously, a more intuitive and direct interaction could be achieved if computers could capture human movements and then recognize, even understand, the meanings of these movements. For example, computers could estimate where the user is pointing, detect when the user claps hands, interpret why the user waves hands, even understand when the user is happy, etc. This concept would bring a revolution to human-computer interaction.

Early practice of this idea concentrated on the design of motion sensors, which are attached to human body parts to measure their motions and transfer the measurements to electromagnetic signals. There are some commercial products available in the market, such as MotionStar and CyberGlove. Nowadays, wireless techniques are also embedded in these motion sensors to get rid of the cable connections. These motion sensors are important for collecting human motion data. However, these invasive sensors are still quite cumbersome for real interactions because people are often reluctant to wear them, forcing researchers to explore other noninvasive ways for natural human-computer interaction.

1.1.3 Vision-based human-computer interaction

Instead of using electromagnetic sensors, vision-based interaction (VBI) aims to develop more natural, intuitive, and convenient interfaces with computers by using live video inputs. Computers are able to "see" and recognize a user's physical actions. Furthermore, a user's psychological status can be estimated.

Early research on vision-based motion capturing usually needed the help of color markers. In current state-of-the-art of vision-based interaction, research is focusing on tracking people directly, recognizing faces, understanding facial expressions, interpreting gestures and body motion, and so on.

Vision-based interaction is a challenging interdisciplinary research area, which involves computer vision and graphics, image processing, machine learning, bioinfomatics, and psychology. To make a working system, there are some requirements:

- *Robustness*: In the real-world, visual information could be very rich, noisy, and incomplete, due to changing illumination, clutter and dynamic backgrounds, occlusion, etc. VBI should be able to automatically recover from the failure. VBI systems should be user-independent.

- *Computational efficiency*: Generally, VBI often requires real-time systems. The vision and learning techniques used in VBI should be effective as well as cost efficient.

- *User's tolerance*: The malfunctions or mistakes of VBI should be tolerated. When a mistake is made, it should not incur much loss. Users can be asked to repeat some actions, instead of letting the computer make more wrong decisions.

- *Scalability*: The VBI system should be easily adapted to different scales of applications. Fox example, the core of VBI should be the same for desktop environments and CAVE environments.

1.1.4 Gesture interfaces

The use of hand gestures has become an important part of human-computer interaction in recent years [1, 2, 3, 4, 5, 6]. To use human hands as a natural interface, some glove-based devices have been employed to capture human hand motion by attaching sensors to measure the joint angles and spatial positions of hands directly. Unfortunately, such devices are expensive and cumbersome.

Since rich visual information provides a strong cue to infer the inner states of an object, vision-based techniques provide some promising alternatives to capture human hand motion. At the same time, vision systems could be very cost-efficient and noninvasive. These facts serve as the motivating forces for research in the modeling, analysis, animation, and recognition of hand gestures.

Vision-based gesture interface is a promising direction in HCI [3, 4, 5, 7]. To achieve a natural and intelligent interaction, there are several important research issues in gesture interface:

- *Gesture modeling*: Gesture is a complicated phenomenon. Meaningful information expressed by gestures is embedded in both the static and temporal characteristics of gestures. Gesture modeling is the basis of gesture interface. It involves gesture representation studying, hand shape modeling, kinematic modeling, temporal modeling, and semantic modeling.

- *Gesture tracking*: The hand is highly articulated, and the range of movement of the hand is very large. Hand motion consists of global hand motion and local finger motion. Gesture tracking involves hand localization and capturing articulated finger motion.

- *Gesture recognition*: Static gestures (or postures) could be used to represent static concepts, and temporal gestures represent dynamic movements. Recognizing postures and temporal gestures are challenging problems.

- *Realistic animation*: VBI has the ability to produce responses to human actions. One such response can be an avatar, which should be realistically animated.

1.1.5 Visual learning

To achieve intelligent interaction between computers and humans, computers should have the capacity to "learn". Since the large variation of visual inputs makes it nearly impossible to explicitly represent the rule of interaction, such knowledge must be obtained from a set of examples by machine learning techniques.

Due to the richness of visual inputs and the gap between low-level visual features and high-level concepts, it is often difficult to perform visual tracking, analysis, recognition, and retrieval. Various learning techniques are employed to deal with the large variety of visual content in many vision tasks such as face and gesture recognition, visual tracking, image and video databases, etc. Learning offers a flexible and tractable means to address these problems, since the training data implicitly represent the *a priori* knowledge of the domain that is hard to model explicitly.

1.2 Motivation

One of the core problems in vision-based human-computer interaction is human tracking and motion capturing. It is the first step of human motion recognition and understanding. Specifically, for vision-based gesture interfaces, we need to localize hands in video sequences robustly, and capture global hand poses and local finger articulations accurately. We need to develop effective methods for:

- *Efficient localization*: The efficiency of the localization system not only depends on deliberated implementation, but also on efficient techniques. Since many computationally intensive processes will be carried out after locating the object of interest in videos, localization should use computational resource as economically as possible.

- *Robust tracking*: Visual tracking involves many fundamental problems in computer vision such as matching and recognition. How to make tracking robust is the key to real systems for vision-based interaction. Integration of multiple visual cues could be a powerful way to achieve this goal.

- *Accurate motion capturing*: Human motion is quite complex. For example, the hand motion is highly articulated because the hand has roughly 27 degrees of freedom. Capturing finger articulations is a challenging problem.

What motivates us to develop a new learning paradigm for visual learning in gesture interface is that many learning approaches confront nearly the same difficulties in the practice of vision applications.

- *Few labeled samples*: Although supervised learning techniques are widely used in visual learning tasks, the insufficiency of labeled training data is often one of the problems in the practice of supervised learning. At the current time, it is hard to deal with the uncertainty in pure unsupervised learning due to the lack of supervised information.

- *Feature extraction and selection*: Due to the rich content of an image, feature extraction finds a compact representation of an object or an scene in a lower-dimensional space. Feature selection selects the features most relevant to the learning tasks. One of the difficulties we often confront is how to automatically achieve such representation.

- *Learning in high dimensionality*: Most visual leaning tasks are high dimensional, due to the richness of visual inputs. How to perform effective and efficient learning in high dimensional spaces is a challenging problem.

- *Incremental or on-line learning*: To achieve intelligent interaction, a machine should have the capacity of incremental on-line learning [8], which is a process to learn new knowledge based on already-learned knowledge.

We use several examples to illustrate these common problems in visual learning tasks. One example is invariant object recognition that requires recognition of the object from any view direction. Our task of view-independent hand posture recognition is even difficult because of the variation among different users. Pure supervised techniques need a huge labeled training database. However, to manually label a large data set will be very time-consuming and tedious.

Another example is the task of image retrieval which is to find the maximum possible "similar" images to the query images in the given database [9, 10, 11]. For these information retrieval problems, we are only given very few labeled training samples by queries.

A more interesting example comes from nonstationary color model adaptation. Many techniques are plagued by the large variation in skin tone, unknown lighting conditions, and dynamic scenes. If a color classifier is trained under a specific condition, it may not work well in other scenarios. The learning task is to transduce an old color model to a new environment.

There has been much discussion about enhancing the generalization by looking into the training algorithm, and many schemes have been proposed to avoid overfitting. However, there are fewer concerns about the training data set. The drawbacks of pure supervised and pure unsupervised learning paradigms motivate us to develop a new paradigm.

1.3 Organization

In this dissertation, three novel techniques for nonstationary color tracking, multiple cues integration, and articulated hand motion capturing will be presented, which serve as the basis for the development of vision-based gesture interfaces. This dissertation will also present a new learning paradigm, self-supervised learning, by taking into account both labeled and unlabeled training data. The development of two interesting gesture interface systems will be described. The dissertation is organized as follows:

- Chapter 2 gives a comprehensive overview of the state-of-the-art of the research in vision-based interface. The research on gesture representation is given in Section 2.2. Various aspects of hand modeling are discussed in Section 2.3. We also discuss the feature and data collection for gesture recognition in Sections 2.5.1 and 2.5.2, respectively. Section 2.4.2 discusses some techniques for tracking hand global motion, and Section 2.4.4 reviews and compares the model-based and appearance-based approaches to capturing articulated hand motion. Various methods for hand posture recognition and temporal gesture recognition are given in Sections 2.6 and 2.7. Some of the research directions are described in Section 2.9.

- Chapter 3 presents a nonparametric color model transduction method based on a self-organizing map. An adaptive self-organizing map technique is proposed in Section 3.2. A general technique for model transduction is given in Section 3.3. And the SOM-based transduction technique is discussed in Section 3.4. Segmentation and tracking results are presented in Section 3.5.

- Chapter 4 focuses on multiple cues integration for robust tracking. Based on the graphical model described in Section 4.3, Section 4.5 presents a co-inferencing approach that makes use of sequential Monte Carlo techniques in Section 4.4. Mathematical details of co-inferencing are given in Appendix A, and the implementation of co-inferencing tracking is given in Section 4.6. Section 4.7 describes many experimental results.

- Chapter 5 describes a 3-D model-based approach for capturing articulated hand motion. An introduction of hand motion and a hand model is given in Section 5.2. The methods of capturing global hand motion and local finger articulation are presented in Sections 5.3 and 5.4. Section 5.5 describes a two-step iteration approach to combine the global and local motion estimation. Experiments are given in Section 5.6.

- Chapter 6 presents linear D-EM and kernel D-EM algorithms in self-supervised learning. We review the state-of-the-art of learning from hybrid data set in Section 6.2. The EM algorithm is discussed in Section 6.4 and some problems of the EM-based approach are addressed in Section 6.5. The linear D-EM algorithm and kernel D-EM algorithm are proposed in Sections 6.6 and 6.7, respectively. Extensive experiments on invariant

hand posture recognition and content-based image retrieval are given in Section 6.8. Section 6.9 discusses the proposed self-supervised learning. Some fundamental questions of this paradigm are raised in Section 6.9.2. The proposed self-supervised learning paradigm is given in Section 6.9.3, which includes induction, transduction, and deduction.

- Chapter 7 describes the development of two interesting vision-based gesture interfaces. The design of a vision-based system for the interactive game of "Paper-Rock-Scissors" and the design of the "Visual Panel" system of remote display control are presented in Sections 7.1 and 7.2, respectively.

- Chapter 8 summarizes the dissertation by four typical self-supervised learning problems, i.e., transduction, co-transduction, model transduction and co-inferencing, which serve as clues to glue different parts together. Some interesting future research topics are given at the end.

1.4 Contributions

Original contributions presented in this dissertation span the areas of visual motion capturing and learning techniques for vision-based human-computer interaction. In particular, the following issues have been addressed.

- A novel nonparametric algorithm, SOM transduction, has been developed to conduct model transduction in a dynamic learning environment. It has been used for non-stationary color tracking tasks.

- A novel approach, the co-inferencing algorithm, has been proposed to integrate multiple cues for robust visual tracking. The co-inferencing technique could be extended to many other sensor fusion problems.

- An effective 3-D model-based approach of visual motion capturing has been developed to analyze global hand motion and local finger articulation. The originality lies in the employment of the finger motion model learned from motion data.

- A novel self-supervised algorithm, the D-EM algorithm, has been proposed by integrating discriminant analysis and EM. This algorithm employs both labeled and unlabeled train-

ing data. Both linear D-EM and nonlinear kernel D-EM have been investigated. D-EM has been successfully applied to invariant hand posture recognition tasks and content-based image retrieval tasks. Self-supervised learning techniques have potential applications for many other learning tasks.

- Two interesting prototypes of vision-based gesture interfaces have been developed by integrating algorithms presented in this dissertation.

This dissertation mainly concentrates on the issues in vision-based intelligent human-computer interaction. However, many techniques developed in this work are easily extended to many other tasks and areas. For example, the SOM transduction algorithm could be extended to other distribution adaptation tasks, and the co-inferencing algorithm is ready for many other sensor fusion tasks. The self-supervised learning is a new learning paradigm. It is not by any means confined to this specific area. Actually, this dissertation has described its application to the image retrieval task. The D-EM algorithm could be easily extended to many other information retrieval applications. Although this work does not completely develop the theory of self-supervised learning paradigm, it will motivate more investigation of this new research topic in the near future.

CHAPTER 2

VISION-BASED GESTURE INTERFACES: A REVIEW

2.1 Introduction

This chapter surveys recent studies of vision-based gesture recognition techniques. Section 2.2 discusses several human gesture representation paradigms in psycholinguistic and cognitive studies, since almost all high-level temporal gesture recognition tasks can be represented by those paradigms which serve as a cognitive model for many complicated temporal hand gestures. Since any recognition method needs feature extraction and data collection, Section 2.5.1 discusses the gesture features used in current studies, and Section 2.5.2 provides a brief overview of tracking techniques which serve as the data collection process for vision-based gesture recognition.

Since meaningful hand gestures can be classified as static hand postures and temporal gestures, Sections 2.6 and 2.7 discuss various techniques for hand posture recognition and temporal gesture recognition, respectively. Since sign language recognition is an important task, Section 2.8 discusses several studies related to it.

2.2 Gesture Representation

There have been many psycholinguistic studies of human gestures. Stokoe [12] represents gestures as four aspects: *hand shape*, *position*, *orientation*, and *movement*. Kendon [13] describes a philology of gesture, which consists of *gesticulation*, *language-like gestures*, *pantomimes*, *emblems*, and *sign language*. *Sign languages* are characterized by a specific set of

vocabulary and grammar. *Emblems* are informal gestural expressions in which the meaning depends on convention, culture, and lexicon.

According to different application scenarios, hand gestures can be classified into several categories such as *conversational gestures*, *controlling gestures*, *manipulative gestures*, and *communicative gestures* [4]. Sign language is an important case of *communicative gestures*. Since sign languages are highly structural, they are very suitable as test-beds for vision algorithms. At the same time, they can also be a good way to help the disabled to interact with computers. *Controlling gestures* are the focus of current research in vision-based interface (VBI). Virtual objects can be located by analyzing pointing gestures. Some display-control applications demonstrate the potential of pointing gestures in HCI. Another controlling gesture is the navigating gesture. Instead of using wands, the orientation of hands can be captured as a 3-D directional input to navigate the VEs. The manipulative gesture will serve as a natural way to interact with virtual objects. Teleoperation and virtual assembly are good examples of applications. Communicative gestures are subtle in human interaction, which involves psychological studies; however, vision-based motion capturing techniques can help those studies.

Communicative gestures can be decomposed into three motion phases: *preparation*, *stroke*, and *retraction* [13]. Psycholinguistic studies show that *stroke* may be distinguished from other gesture phases, since *stroke* contains the most information. This model is taken from Quek [14]. He also makes a distinction between *presentation* gestures and *repetitive* gestures.

Bobick [15] emphasizes the dynamical part of gestures. He represents gestures as *movement*, *activity*, and *action*. *Movements* are typically atomic and are the most primitive form of motion that can be interpreted semantically. *Activity* is a sequence of either movements or static configurations. Dynamic models may be used to recognize activities. *Actions* are the high-level entities that people typically use to describe what is happening. Time and context become fundamental, though how much one has to reason about context is unclear.

2.3 Hand Modeling

Human hand motion is highly articulate because the hand consists of many connected parts, leading to complex kinematics. At the same time, hand motion is also highly constrained, which makes it difficult to model. Usually, the hand can be modeled in several aspects such as

shape, kinematic structure, dynamics, and semantics. Hand models are not only used in hand animation applications, but also are employed to analyze hand motion. Different models are suitable for different HCI applications.

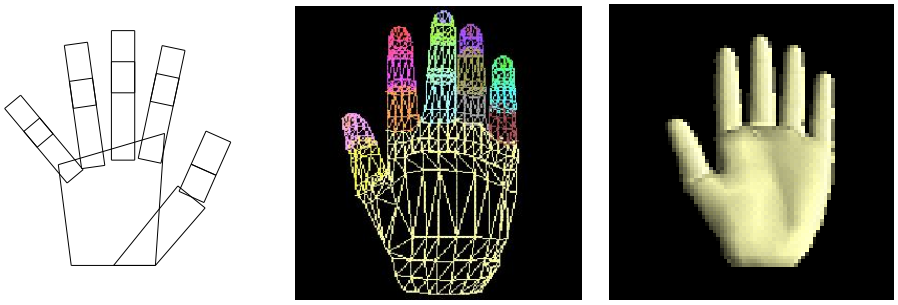
2.3.1 Modeling the shape

Hand shape models can be classified into several groups such as geometrical models, physical models, and statistical models.

Geometrical models are suitable for 3-D rendering and hand animation applications. Moreover, they could be employed to analyze hand motion using the approach of analysis-by-synthesis [16, 17, 18]. Both physical models and statistical models emphasize hand deformation. The difference is that physical models aim for an explicit representation of deformation, while statistical models characterize hand deformation implicitly by learning from a set of examples.

Spline-based geometrical surface models represent a surface with splines to approximate arbitrarily complicated geometrical surfaces. These spline-based surface models can be made as realistic as possible, but many parameters and control points need to be specified [17]. An alternative is to approximate the homogeneous body parts by simpler parameterized geometric shapes such as generalized cylinders or super-quadrics. The advantage of this method is that it can achieve equally good surface approximation with less complexity [16, 18]. Other than parametric models, free-form hand models are defined on a set of 3-D points [19]. Polygon meshes that are formed by those 3-D points approximate the hand shape, which is computationally efficient. For computational efficiency, cardboard models could be used for visual motion capturing. Each piece in a cardboard model is a two-dimensional (2-D) plane, but the joint angles could be adjusted. Examples of different hand models are shown in Figure 2.1. Cyber Scanner, MRI techniques, or other space digitizers may be used to obtain the range data directly [19]. Another way is to reconstruct the hand model from multiple images of different views.

Physical hand shape models emphasize the deformation of the hand shape under the action of various forces [20]. The motion of the model is governed by Newtonian dynamics. The internal forces are applied to hold the shape of the model, and the external forces are used to fit the model to the image data. Examples are the simplex mesh model [19] and the finite element method model [21].



(a) (b) (c)

Figure 2.1 Various hand models. (a) Cardboard model, (b) wireframe model, (c) polygon-mesh model.

Statistical hand shape models [19] learn the deformation of hand shape through a set of training examples that can be 2-D images or range images. Mean shape and modes of variation are found using principal component analysis (PCA). A hand shape is generated by adding a linear combination of some significant modes of variation to the mean shape.

2.3.2 Modeling the kinematic structure

Figure 2.2 shows the skeleton of a hand. Each finger consists of three joints whose names are indicated in the figure. Except for the thumb, there are 2 degrees of freedom (DOF) for *metacapophalangeal* (MCP) joints, and 1 DOF for *proximal interphalangeal* (PIP) joints and *distal interphalangeal* (DIP) joints. For simplicity, the thumb could be modeled by a 5 DOF kinematic chain, with 2 DOF for the *trapeziometacarpal* (TM) and MCP joint and 1 for *interphalangeal* (IP) joint. Considering global hand poses, human hand motion has roughly 27 DOF. The challenge of hand motion analysis lies in the fact that hand motion is highly articulated. Each finger can be modeled by a kinematic chain, in which the palm is its base reference frame and the fingertip is the end-effector. We can write:

$$\mathbf{x}^b = \mathbf{H}_0^b(\ \theta_{MCP\_AA})\mathbf{H}_1^0(\ \theta_{MCP})\mathbf{H}_2^1(\ \theta_{PIP})\mathbf{H}_3^2(\ \theta_{DIP})\mathbf{x}^3 \tag{2.1}$$

where $\mathbf{x}^3$ is the fingertip in DIP frame, while $\mathbf{x}^b$ is fingertip in the base frame. $\mathbf{H}_i^j$ is the coordinate transformation which transforms the i frame to the j frame.

When fixing the joint length, hand kinematics can be characterized by its joint angles. The inverse kinematics problem is often involved to calculate joint angles when analyzing finger motion. Generally, gradient-based methods can be used to solve this problem by deriving

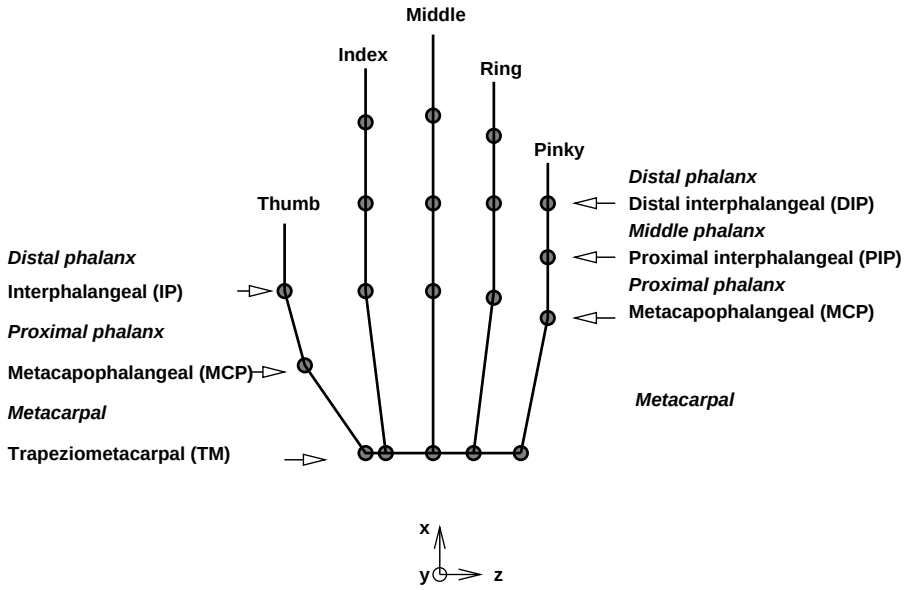


Figure 2.2 Hand skeleton structure. Generally, we can assume 2 DOF for the MCP and TM joint, and 1 DOF for all the other joints. Thus, the hand has roughly 21 DOF for its local finger motion.

the kinematical Jacobian [22]. There are other alternatives in the literature such as genetic algorithms [23]. However, such an inverse kinematics problem is ill-posed, such that a unique solution cannot be guaranteed, which makes the analysis formidable.

Fortunately, natural hand motion is also highly constrained. One set of constraints, usually referred to as static constraints, consists of the limits of the range of finger motions as a result of hand anatomy, such as $0^0 \leq \theta_{MCP} \leq 90^0$. These constraints limit hand articulation within a boundary. Another type of constraint describes the correlations among different joints, and thus reduces the dimensionality of hand articulation. For example, the motions of the DIP joint and PIP joint are generally not independent and they could be described as $\theta_{DIP} = (2/3)\theta_{PIP}$ from the study of biomechanics [17, 24]. Although this constraint could be intentionally made invalid, it is a good approximation of natural finger motion.

Unfortunately, not all such constraints could be quantified in closed forms. There are few studies of finger motion constraints in the literature. A preliminary investigation could be found in [25], in which learning techniques are employed to model the hand configurations space directly by collecting a large set of hand motion data [25]. The computational complexity of finger motion analysis could be reduced significantly when considering such motion constraints.

2.3.3 Modeling the dynamics

To capture complex hand motion and recognize continuous hand gestures, the dynamics and semantics of hand motion should also be modeled.

Kalman filtering and extended Kalman filtering (EKF) techniques are widely adopted to model the dynamics [18]. EKF works well for some tracking tasks. However, it is based on a small motion assumption that often fails to hold for hand motion.

Simple hand gestures can be modeled by a finite state machine [26], but it is insufficient to represent complex hand dynamics. Rule-based approaches can be applied to model complex hand movements [14]. However, many heuristics are needed to construct the rules. Considering the similarities between sign languages and spoken languages, the hidden Markov model (HMM) and its variants are also used to model the hand dynamics [27, 28]. As a generalization of HMM, dynamic Bayesian net [29] is another promising approach to model the hand dynamics. These methods are essentially learning methods that learn the intrinsic dynamics from a set of training data. The knowledge of dynamics and semantics is not explicitly expressed in these methods but implicitly stored in the structures of the learning models.

The learning results of these methods depend on the training data set, structures of learning models, and training methods. One of the common problems of the learning approaches is that generalization of the learning results largely depends on the training data. However, obtaining the training samples is not a trivial problem. Currently, the research of learning dynamics (behaviors, semantics) of human motion has drawn much attention from researchers in HCI, computer vision, computer graphics, and psychology.

2.4 Capturing Human Hand Motion

Hand motion capturing is finding the global and local motion of hand movements. Several different model-based approaches will be discussed in this section.

2.4.1 Formulating hand motion

Highly articulated human hand motion consists of the global hand motion and local finger motion, which can be expressed as $\mathbf{M} = [\mathbf{M}_G, \mathbf{M}_L]$, where $\mathbf{M}$ is the hand motion, $\mathbf{M}_G$ is the global motion, and $\mathbf{M}_L$ is the local motion. Global hand motion that presents large rotation

and translation can be written as $\mathbf{M}_G = [\mathbf{R}, \mathbf{t}]$, where $\mathbf{R}$ and $\mathbf{t}$ are rotation and translation, respectively. One important issue is how to reliably track the global motion in image sequences.

Local hand motion is articulated, and self-occlusion makes the detection and tracking local hand motion challenging. Local hand motion can be parameterized with the set of joint angles (or *hand state*) $\mathbf{M}_L = [\Theta]$ where Θ is the joint angle set. Consequently, hand motion can be expressed as $\mathbf{M} = [\mathbf{R}, \mathbf{t}, \Theta]$.

One possible way to analyze hand motion is the appearance-based approach, which emphasizes the analysis of hand shapes in images [3]. However, local hand motion is very hard to estimate by this means. Another possible way is the model-based approach [16, 17, 18, 19, 20, 22, 23, 24]. With a single calibrated camera, local hand motion parameters can be estimated by fitting the 3-D model to the observation images. Multiple camera settings are helpful to deal with occlusion [20, 22, 24]. The use of a 3-D model can largely alleviate the problem of depth ambiguity since the structure of the hand is included in the model.

2.4.2 Localizing hands in video sequences

Hand localization locates hand regions in image sequences. Skin color offers an effective and efficient way to fulfill this goal. The core of color tracking is color-based segmentation. According to the representation of color distribution in certain color spaces, current techniques of color tracking can be classified into two general approaches: nonparametric [2, 30, 31] and parametric [32, 33]. Figure 2.3 gives an example of segmentation-based hand localization, in which the input image is segmented by color, and the hand blob is localized by grouping skin-color pixels.

(a) (b) (c)

Figure 2.3 Hand localization. (a) Input image, (b) segmentation result, (c) hand blob located by analyzing the segmented image pixels.

One of the nonparametric approaches is based on color histograms [2, 30]. Because color space is quantized by the structure of the histogram, this technique shares the same problem with nonparametric density estimation, in which the level of quantization will affect the estimation. How to select a good quantization level of the color histogram is not trivial. Although nonuniform quantization would perform better than uniform quantization, it is much more complicated. Another nonparametric approach is proposed in [31] based on the self-organizing map, an unsupervised clustering algorithm to approximate color distribution. Generally, these nonparametric approaches work effectively when the quantization level is properly set and there are sufficient data.

Parametric approaches model the color density in parametric forms such as Gaussian distribution or Gaussian mixture models [32, 33]. Expectation-maximization (EM) offers a way to fit probabilistic models to the observation data. The difficulty of *model order selection* could be handled by heuristics [32] or cross-validation.

However, when we try to apply these techniques to track the human hand and face in some virtual environment (VE) applications, this problem is still made challenging by some special difficulties such as large variation in skin tone, unknown lighting conditions, and dynamic scenes.

In order to achieve user-independence, the tracking algorithm should be able to deal with the large variation in skin color for different people. One possible solution is to make a generic statistical model of skin color by collecting a huge training data set [30] so that the generic color model works for every user. However, collecting and labeling such a huge database is not trivial.

Even though such a good generic color model can be obtained, we have to face another difficulty in color tracking: generic color models would be incapable to handle changing lighting conditions unless some invariants could be found. Many color tracking techniques assume controlled lighting. However, in many cases, the interested object may be shadowed by other objects or by the object itself so that the color looks very different. What is more, we cannot assume constant lighting sources, since the lighting directions, intensities, and tones might change. In some VE applications, since the graphics rendered in the display keep changing, the reflective lights would change the apparent color of objects. This color constancy problem is not trivial in color tracking.

Because of dynamic scenes and changing lighting conditions, the color distribution over time is nonstationary, since the statistics of color distribution will change with time. If a color classifier is trained under a specific condition, it may not work well in other scenarios.

There have been some researchers who have looked into the nonstationary color distribution problem in color tracking. Several methods have been proposed to approach this problem. A scheme of color model adaptation was addressed in [32], in which a Gaussian mixture model was used to represent color distribution, and a linear extrapolation was employed to adjust the parameters of the model by a set of labeled training data drawn from the new frame. However, since the new image is not segmented, this labeled data set is not reliable.

In [31], the scheme of *transduction of SOM* was proposed to update the weights and structure of the trained SOM to capture the new color distribution, according to a set of new training data, which consists of both labeled and unlabeled samples. Since the transduction of SOM combines unsupervised and supervised updating, a large amount of labeled training data is not required.

To lead to a robust and efficient localization, besides the color cue, hand shape and motion could also be employed for localization. One important research problem is the integration or fusion of multiple cues [27, 34].

Besides color, the hand can be localized by other cues, such motion and shape. A scheme of tracking the contours of hands is given in [35, 36]. Although color-based localization is the most computationally efficient, integrating other cues would enhance the robustness of the localization.

2.4.3 Selecting image features

To estimate the parameters of the model, some image features should be extracted and tracked to serve as the observation of the estimators. Hand image features can be geometric features such as points, lines, contours, and silhouettes [17]. The fingertip is one of the frequently used features, because the positions of fingertips are almost sufficient to recognize some gestures due to the highly constrained hand motion [24]. Color markers are often used to help track the 3-D positions of fingertips [16, 24]. Some researchers estimate the positions and orientations of fingertips by fitting a 3-D cylinder to the images [16]. Line fitting is also a frequently used technique to detect the fingertips [22].

2.4.4 Capturing hand motion in full DOF

To capture articulated hand motion in full DOF, both global hand motion and local finger motion should be determined from video sequences. It is a challenging problem to analyze and capture hand motion, since the hand is highly articulated. Different methods have been used to approach this problem. One possible method is the appearance-based approach, in which 2-D deformable hand shape templates are used to track a moving hand in 2-D. However, this method is insufficient to recover full articulations, since it is difficult to infer finger joint angles based on appearances only.

Another possible way is the 3-D model-based approach, which takes the advantages of *a priori* knowledge built in the 3-D models. This approach aligns a 3-D model to images or even range data by estimating the parameters of the model. In 3-D model-based methods, image features could be looked at as the image evidence or image observation of a 3-D model that is projected to the image plane. A 3-D model with different parameters will produce different image evidence. Model-based methods recover the joint angles by minimizing the discrepancy between the image feature observations and projected 3-D model hypotheses [16, 17, 18, 19, 22, 23, 24, 37], which is a challenging optimization problem. Two important tasks in the model-based approach are determining the match and searching the hand joint angles space. Some examples are shown in Figure 2.4, in which the parameters of a cardboard hand model are adjusted to match three input images. Generally, due to the huge search space of hand articulation, the optimization involved is difficult and computationally intensive.

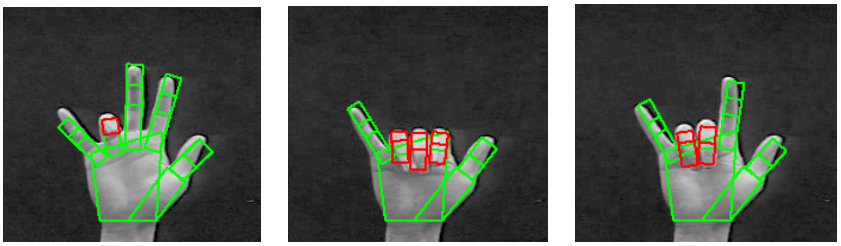


Figure 2.4 Capturing articulate hand motion using a cardboard hand model. Hand pose and finger joint angles could be recovered by fitting the model to the images. The fitting minimizes the discrepancy between image feature observations and projected models.

Many methods tend to estimate the global and local hand motion simultaneously. In [22], the hand was modeled as an articulated stick figure, and point and line image features were used for the registration. Hand motion capturing was formulated as a constrained nonlinear

programming problem. The drawback of this approach is that the optimization is often trapped in local minima. Another idea is to model the surface of the hand [16, 17, 18], and then hand configurations can be estimated using the analysis-by-synthesis approach, in which candidate 3-D models are projected to the image plane and the best match is found with respect to some similarity measurements. If the surface model is very fine, an accurate estimation can be obtained. However, those hand models are user-dependent. Rough models can only give approximate estimations [18].

To ease the optimization, a decomposition method can be adopted to analyze articulate hand motion by decoupling hand motion to its global motion and local finger motion. Global motion is parameterized as the pose of the palm, and local motion is parameterized as the set of joint angles. A two-step iterative algorithm could be used to find an accurate estimation [23]. Given an initial estimation, hand pose is estimated using least median of squares with joint angles fixed. Then the joint angles are recovered by a genetic algorithm with the global hand pose fixed. Those two steps are alternately iterated until the solution converges [23].

2.5 Data Preparation for Recognition

Selecting good features is crucial to gesture recognition because hand gestures are very rich in shape variation, motion, and textures.

2.5.1 Features for gesture recognition

For static hand posture recognition, although it is possible to recognize hand posture by extracting some geometric features such as fingertips, finger directions, and hand contours, such features are not always available and reliable due to self-occlusion and lighting conditions. There are also many other nongeometric features such as color, silhouette, and textures. However, they are inadequate in recognition. Since it is not easy to specify features explicitly, the whole image or transformed image is taken as the input, and features are selected implicitly and automatically by the recognizer.

Cui and Weng [38] investigate the difference between the *most discriminating features (MDF)* and the *most expressive features (MEF)*. MEFs are extracted by K-L projection. However, MEFs may not be the best for classification because the features that describe some

major variations in the class are typically irrelevant to how the subclasses are divided. MDFs are selected by multiclass, multivariate discriminate analysis and have a significantly higher capability to catch major differences between classes. Their experiments also showed that MDFs are superior to the MEFs in automatic feature selection for classification.

Recognizing temporal gestures not only needs spatial features, but also requires temporal features. It is possible to recognize some gestures by 2-D locations of hands. However, it is not general and view-dependent. The most fundamental feature is the 2-D location of the interested blob. Wren et al. [39] use a multiclass statistical model of color and shape to obtain a 2-D representation of the head and the hand in a wide range of viewing conditions in their tracking system *Pfinder*.

In order to achieve spatial invariant recognition, 3-D features are necessary. Campbell et al. [40] investigated the 3-D invariant features by comparing the recognition performance on ten different feature vectors derived from a single set of 18 *T'ai Chi* gestures which are used in the *Staying Alive* application developed by Becker and Pentland [41]. A hidden Markov model (HMM) is used as the recognizer. They reported that $(dr, d\theta, dz)$ had the best overall recognition rates. At the same time, their experiments highlight the fact that choosing the right set of features can be crucial to the performance.

Features for temporally invariant gesture recognition are hard to specify since they depend on the temporal representation of gestures. However, the features can be handled implicitly in some recognition approaches such as finite state machine and HMM, which will be discussed in Section 2.7.

2.5.2 Data collection for recognition

To collect data for temporal gesture recognition is not a trivial task. The hand has to be localized in the image sequences and segmented from the background. Two-dimensional tracking supplies the localized information such as hand bounding boxes and centroid of hand blobs. Simple 2-D motion trajectories can be extracted from the image sequences. In some cases, these 2-D features are sufficient for gesture recognition. There have been many 2-D tracking algorithms such as color tracking, motion tracking, template matching, blob tracking, and multiple cues integrating.

Although 2-D tracking gives the position information of the hand, some recognition applications need still more features such as hand orientation and hand shape. Three-dimensional tracking approaches try to locate the hand in 3-D space by giving the 3-D position and orientation of the hand. However, since the hand cannot be treated as a rigid object, it is very hard to estimate the hand orientation. Three-dimensional position of the hand can be achieved by stereo camera or model-based approaches.

Since the hand is highly articulated and its shape depends on the viewpoint, hand shape is hard to describe. Several studies try to recover the *state of the hand* which is represented by the set of joint angles, which is full DOF tracking. If the hand configuration can be estimated, recognizing finger spelling may be easier. However, how to estimate the configuration of articulated objects needs more study.

2.6 Static Hand Posture Recognition

Although hand gestures are complicated to model since the meanings of hand gestures depend on people and cultures, a specific hand gesture vocabulary can be always predefined in many applications, such as Virtual Environment (VE) applications, so that ambiguity can be limited. Generally, these hand gestures can be either static hand postures or temporal hand gestures. Hand postures express some concepts by hand configurations and hand shapes, while temporal hand gestures represent some actions by hand movements. Sometimes, hand postures act as special transition states in temporal gestures and supply a cue to segment and recognize temporal hand gestures. Some research results show that static hand signs and temporal hand gestures seldom present simultaneously, which suggests that we study static hand gestures and temporal gestures separately.

In contrast to sign languages, the gesture vocabulary in VE applications is structured and disambiguated. In such scenarios, some simple controlling, commanding, and manipulative gestures are defined to fulfill natural interaction such as pointing, navigating, moving, rotating, stopping, starting, selecting, etc. These gesture commands can be simple in the sense of motion; however, many different hand postures are used to differentiate and switch among those commanding modes. For example, it makes sense to estimate a gesture's pointing direction

only if we know it is a pointing gesture. This problem is an empirical problem in most VE applications.

Although this problem can be formulated as a classification problem of different predefined static hand postures, there are still many difficulties. The first is view-independent hand posture recognition [42, 43], which means hand postures must be recognized from any view direction. This is a natural requirement in many VE applications. In most cases, since users do not know where the cameras are, the naturalness and immersiveness will be ruined if users are obliged to issue commands to an unknown direction. Another difficulty is that the human hand is highly articulated and deformable; the large variation in hand postures should be handled to make a user-independent system.

Since hand postures can express some concepts as well as act as special transition states in temporal gestures, recognizing or estimating hand postures or human postures is one of the main topics in gesture recognition. Some work has been done in this area.

2.6.1 3-D Model-based approaches

One approach is the 3-D model-based approach, in which the hand configuration is estimated by taking advantage of 3-D hand models [16, 18, 19, 22, 23, 24, 44, 45]. Since hand configurations are independent of view directions, these methods could directly achieve view-independent recognition. Different models take different image features to construct feature-model correspondences. Joint angles can be estimated by minimizing a projected surface model and some image evidence such as silhouettes in the light of "analysis-by-synthesis" [16, 17, 24, 44]. However, this approach needs good surface models and the process of projection-and-comparison is expensive. Alternatively, point and line features are employed in kinematical hand models to recover joint angles [18, 22, 23]. Hand postures could be estimated accurately if the correspondences between the 3-D model and the observed image features are well established. Physical models and statistical models [19] were also employed to estimate hand configurations. However, the ill-posed problem of estimating hand configuration is not trivial. Many current methods require reliable feature detection, which is plagued by self-occlusion. Another drawback is that it is not trivial to achieve user-independence, since 3-D models should be calibrated for each user; otherwise the accuracy will be sacrificed.

2.6.2 Appearance-based approaches

Although accurate estimation of hand configuration is important in some applications such as manipulating virtual objects or multi-DOF input devices, a classification of hand postures is often enough in many other applications such as commands switching. Since the appearances are much different among different hand postures and these differences are not large among different people, an alternative approach is an appearance-based approach [38, 42, 46, 47, 48, 49], which aims to characterize the mapping from the image feature space to the possible hand configuration space directly from a set of training data. This approach often involves learning techniques. Images for different hand postures are shown in Figure 2.5.

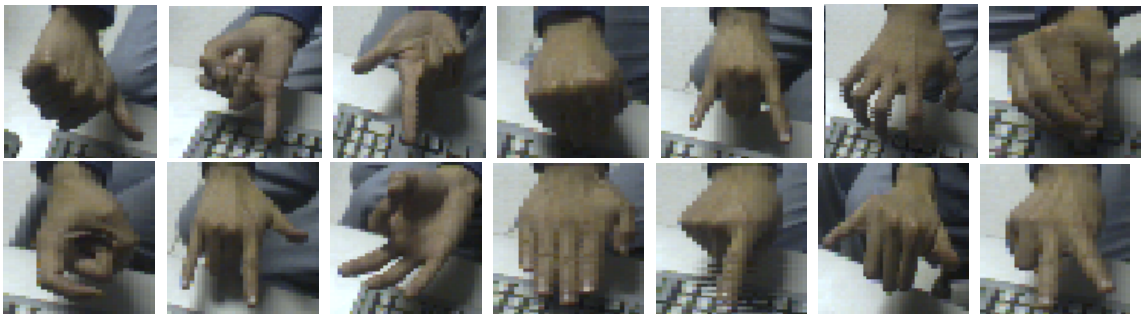


Figure 2.5 Recognizing different hand postures.

Cui and Weng [38] use the most discriminating features to classify hand signs by partitioning the MDF space. A manifold interpolation scheme is introduced to generalize to other variations from a limited number of learned samples. Their algorithm can handle complex backgrounds.

Triesch and von de Malsburg [49] employ the *elastic graph matching* technique to classify hand postures against complex backgrounds. Hand postures are represented by labeled graphs with an underlying two-dimensional topology. Attached to the nodes are *jets*, which are a sort of local image description based on Gabor filters. The recognition rate against complex background is 86.2%. This approach can achieve scale-invariant and user-independent recognition, and it does not need hand segmentation. Since using one graph for one hand posture is insufficient, this approach is not view-independent.

Quek and Zhao [47] introduced an inductive learning system which is able to derive rules of disjunctive normal form formulate. Each DNF describes a hand pose, and each conjunct within the DNF constitutes a single rule. Twenty-eight features such as the area of the bounding box,

the compactness of the hand, and the normalized moments, served as the input feature vector for their learning algorithm. They obtained a 94% recognition rate.

Nolker and Ritter [45] detected the 2-D location of fingertips by the *local linear mapping* (LLN) neural network, and those 2-D locations are mapped to 3-D position by the *parametric self-organizing map* (PSOM) neural network, since PSOM has the ability to perform an associative completion of fragmentary input. By this means, their approach can recognize hand pose under different views.

Although it is easier for the appearance-based approach to achieve user-independence than the model-based approach, there are two major difficulties of this approach: automatic feature selection and training data collection. Although there has been much discussion about feature extraction [45, 47, 49] and selection [38, 46], little has been addressed on the training data. The generalization of many current methods largely depends on their training data sets. In general, good generalization requires a large and representative labeled training data set. However, to manually label a large data set will be very time-consuming and tedious. Although unsupervised schemes have been proposed to cluster the appearances of 3-D objects [50], it is hard for the pure unsupervised approaches to achieve accurate classification without supervision. In [42], a hybrid learning approach was proposed to employ a large set of unlabeled images in training.

2.7 Temporal Gesture Recognition

There are some similarities between temporal gestures and speech so that some techniques such as HMM can be applied to gesture. However, temporal gesture is more complicated than speech. Some low-level movements can be recognized using dynamic models. Some gesture semantics can be exploited to recognize high-level activities. Example-based learning methods can also be used. There are also many other techniques developed in recent years.

2.7.1 Recognizing low-level motion

Modeling the low-level dynamics of human motion is important not only for human tracking, but also for human motion recognition. It serves as a quantitative representation of simple movements so that those simple movements can be recognized in a reduced space by the trajectories of motion parameters. However, those low-level dynamics models are not sufficient

to represent more complicated human motions. Some low-level motions can be represented by simple dynamic processes, in which the Kalman filter technique is often employed to estimate, interpolate, and predict the motion parameters. However, this simple dynamic model is not sufficient to model most cases of human motion, and the Gaussian assumption of the Kalman filtering is usually invalid.

Black and Jepson [51] extended the CONDENSATION algorithm to recognize temporal trajectories. Since a sampling technique is used to represent the probability density in the CONDENSATION algorithm, their approach avoids some difficulties of Kalman filtering. Gesture recognition is achieved by matching of input motion trajectories and model trajectories using *dynamic time warping* (DTW).

Pentland and Liu [52] try to represent human behavior by a complex, multistate model. They used several alternative models to represent human dynamics, one for each class of response. Model switching is based on the observation of the state of the dynamics. This approach produces a generalized maximum likelihood estimate of the current and future values of the state variables. Recognition is achieved by determining which model best fits the observation.

Rittscher and Blake [53] push the technique of combining the idea of model switching and CONDENSATION. They use mixed discrete/continuous states to couple perception with classification, in which the continuous variable describes the motion parameters and the discrete variable labels the class of the motion. An ARMA model is used to represent the dynamics. This approach can achieve automatic temporal sequence segmentation.

There is also some work dealing with specific gestures. Cohen et al. [54] use a dynamic model to represent circle and line gestures to generate and recognize basic oscillatory gestures such as crane control gestures.

2.7.2 Recognizing high-level motion

Many applications need to recognize more complex gestures which include semantic meaning in the movements. Modeling the dynamics alone is not sufficient in such tasks.

The *finite state machine* is a commonly employed technique to handle this situation. Davis and Shah [16] use this technique to recognize simple hand gestures. Jo, Kuno, and Shirai [55] take this approach to recognize manipulative hand gestures such as grasping, holding, and extending. The task knowledge is represented by a state transition diagram, in which each state

indicates possible gesture states at the next moment. By using a rest state, all unintentional actions can be ignored. Pavlović and Berry [56] also take this approach.

Another approach is rule-based modeling. Quek [14] uses extended variable-valued logic and a rule-based induction algorithm to build an inductive learning system to recognize 3-D gestures. Cutler and Turk [57] build a set of simple rules to recognize gestures such as waving, jumping, marching, etc.

Pinhanez and Bobick [58] develop a new representation for temporal gestures, a 3-valued domain {past, now, fut}(PNF) network. The occurrence of an action is computed by minimizing the domain of its PNF-network, under constraints imposed by the current state of the sensors and the previous states of the network.

Other promising approaches to modeling the semantics of temporal gestures are the *Bayesian networks* and the *dynamic Bayesian networks*. Pavlović [29] has pushed these ideas forward recently.

2.7.3 Gesture recognition by HMM

Pentland and Liu [52] use HMM to model the state transitions among a set of dynamic models. Bregler [59] takes the same approach. HMM has the capacity not only to model the low-level dynamics, but also the semantics in some gestures. Stoll and Ohya [60] employ HMM to model semantically meaningful human movements, in which one HMM is learned for each motion class. The data used for modeling the human motions is an approximate pose derived from an image sequence. Nam and Wohn [61] present a HMM-based method to recognize some controlling gestures. Their approach takes into account not only hand movement, but also hand postures and palm orientations.

There are also many variations of HMM. Yang et al. [62] model the gesture by employing a multidimensional HMM, which contains more than one observation symbol at each time. Their approach is able to model multipath gestures and provides a means to integrate multiple modalities to increase the recognition rate.

Since the output probability of feature vectors of each state in HMM is unique, HMM can handle only piecewise stationary processes which are not adequate in gesture modeling. Kobayashi and Haruyama [63] introduce the *partly-hidden Markov model* (PHMM) for temporal matching. Darrell and Pentland [64] introduce a hidden-state reinforcement learning paradigm

based on the *partially observable Markov decision process* to gesture recognition by which an active camera is guided.

When the Markov condition is violated, conventional HMMs fail. HMMs are ill-suited to systems that have compositional states. Brand et al. [65] presented an algorithm for coupling and training HMMs to model interactions between processes that may have different state structures and degrees of influence on each other. These problems often occur in vision, speech, or both. The coupled HMMs are well suited to applications requiring sensor fusion across modalities.

Wilson and Bobick [28] extended the standard HMM method to include a global parametric variation in the output probabilities of the HMM to handle parameterized movements such as musical conducting and driving by the EM algorithm. They presented results on two different movements, i.e., a size gesture and a point gesture, and show robustness with respect to noise in the input features.

2.7.4 Other techniques

There are also many statistical learning techniques applied to gesture recognition. As we described before, Cui and Weng [66] use the multiclass, multidimensional discriminant analysis to automatically select the most discriminating features for gesture recognition. Polana and Nelson [67] attempt to recognize motion by low-level statistical features of image motion information. A simple nearest centroid algorithm serves as the classifier. Their experiments show their approach is suitable for repetitive gesture recognition. Watanabe and Yachida [68] introduce an eigenspace which is constructed from multiple input image sequences to recognize gestures. Since this eigenspace represents the approximate 3-D information for gestures, their approach can handle self-occlusion.

Bobick and Ivanov [15] model the low-level temporal behaviors by HMM techniques. The outputs of HMM serve as the input stream of a stochastic context-free grammar parsing system. The grammar and parser provide longer range temporal constraints. The uncertainty of low-level movement detection is disambiguated in the high-level parser, which includes a priori knowledge about the structure of temporal actions.

Yang and Ahuja [69] use *time-delay neural networks* (TDNN) to classify motion patterns. TDNN is trained with a database of more than ASL signs. The input of the TDNN is the motion trajectories extracted by multiscale motion segmentation.

2.8 Sign Language

Unlike general gestures, sign languages are highly structured so that they provide an appealing test bed for understanding more general principles. However, because there are no clear boundaries between individual signs, recognition of sign languages is still very difficult. Speech recognition and sign language recognition are parallels. Both are time-varying processes, which show statistical variations, making HMMs a plausible choice for modeling the processes. And both must devise ways to cope with context and coarticulation effects. HMMs provide a framework for capturing the statistical variations in both position and duration of the movement. In addition, they can segment the gesture stream implicitly.

There are two kinds of gestures to be recognized: one is isolated gesture, and the other is continuous gesture. The presence of silence makes the boundaries of isolated gestures easy to spot. Each sign can be extracted and presented to the trained HMMs individually. Continuous sign recognition, on the other hand, is much harder since there is no silence between the signs. Here HMMs offer the compelling advantage of being able to segment the streams of signs automatically with the Viterbi algorithm. Coarticulation is difficult to handle in continuous recognition, since it results in the insertion of an extra movement between the two signs.

Starner et al. [27] employ HMM to recognize American Sign Language (ASL). They assume that detailed information about hand shape is not necessary for humans to interpret sign language, so a coarse tracking system is used in their studies.

There are several possible approaches to deal with the coarticulation problem. One is to use context-dependent HMMs, and the other is modeling the coarticulation. The idea of context-dependent HMMs is to train bi-sign or even tri-sign context-dependent HMMs. However, this method cannot work well. Vogler and Metaxas [70, 20] study the coarticulation in sign language recognition. They propose an unsupervised clustering scheme to obtain the necessary classes of "phonemes" for modeling the movements between signs. Recently, they use phonemes instead of whole signs as the basic units so that the ASL signs can be broken into phonemes such as

movements and holds, and HMMs are trained to recognize the phonemes [70]. Since the number of phonemes is limited, it is possible to use HMMs to recognize large-scale vocabularies.

Liang and Ouhyoung [71] also take the HMM approach to the recognition of continuous Taiwanese Sign Language with a vocabulary of 250 signs. The temporal segmentation is performed explicitly based on the discontinuity of the movements according to four gesture parameters such as posture, position, orientation, and motion.

2.9 Research Directions

Although much progress has been made in recent years, there are still many issues related to gesture analysis and recognition that need to be adequately addressed in the future.

2.9.1 Robust hand localization

Although the idea of localizing the hand by tracking skin color is straightforward, in practice, there are some challenging problems with color tracking. Many color tracking techniques assume controlled lighting. However, due to the dynamic scenes and changing lighting conditions, the color distribution over time is nonstationary. If a color classifier is trained under a specific condition, it may not work well in other scenarios. Besides the large variation in skin colors, in some VE applications, since the graphics rendered in the display keep changing, the reflected lights would probably change the skin color as well. This color consistency problem is not trivial in tracking skin color.

Recently, some researchers have begun to look into the nonstationary color distribution problem in color tracking. Several color model updating methods have been proposed to solve this problem [31, 32, 72]. However, handling the nonstationary color is still an open research problem. In the mean time, to achieve a robust hand localization system, multiple cues should be integrated. Better approaches for integration should be studied in the future.

2.9.2 Modeling motion constraints

Although the hand is highly articulated, natural finger motion is also highly constrained. These constraints largely reduce the possible hand configuration space. Consequently, the search space would be significantly reduced in hand posture estimation, and the articulate

motion capturing would be more efficient. Unfortunately, most such constraints are impossible to represent explicitly, partly due to the large variation in finger motion.

However, to achieve robust and efficient estimation of hand configuration and realistic hand animation, such constraints have to be modeled. Instead of explicit modeling, learning techniques could be taken to characterize the hand configuration space. A more profound investigation should be conducted.

2.9.3 Motion editing for animation

Realistic articulated hand animation should be considered in the future. The human animation produced by many current animation systems looks very unrealistic and still looks like robots, due to the fact that the current motion model is too simplified and largely dependent on kinematics. To achieve realistic animation, the natural motion constraints should be integrated with animation systems. Another research direction is to achieve personalized animation, in which different styles of motion can be produced with low cost. Schemes of avoiding the violation of body constraints and collision detection should be built into animation systems.

2.9.4 Recognizing temporal patterns

Although HMM is widely used in speech recognition, and many researchers are applying HMM to temporal gesture recognition, current examples of gesture recognition by HMM still have very limited vocabularies. Compared to HMM in speech recognition, data collection for HMM training in temporal gesture recognition is very difficult, which is part of the reason that large vocabularies are prevented. A crucial issue of training data collection is motion capturing. Due to its lower cost and noninvasive nature, vision-based motion capturing would be one of the ideal approaches to collect motion training data. However, there are many challenging and unsolved problems in vision-based motion capturing techniques. Another issue is gesture coarticulation, which makes the extraction and segmentation of gesture commands even harder in continuous hand movements.

In a word, a good representation of temporal gestures needs to be found in future research. It could be a very different representation from speech signals. Motion interpretation is a quite ill-posed problem, in which cognitive science and psychological studies may be combined. In

the near future, it is very possible to develop task-specific gesture systems, but we are still far from a general purpose temporal gesture recognition and understanding system.

2.9.5 Other open questions

To achieve an immersive interaction, multimodality by integrating hand gestures and speech should also be adequately addressed in the future. Recent research shows that there is a complementarity among different modalities such as hand gestures and speech [73]. More profound research should be conducted.

Current research focuses on single hand gestures for simplicity. However, two-handed gestures should also be studied in the future, since they are more expressive and allow more natural interaction.

CHAPTER 3

NONSTATIONARY COLOR TRACKING

3.1 Introduction

In most vision-based interaction systems, localizing and tracking targets in video sequences are two important and related issues, since such steps will provide inputs to other processing steps such as target recognition and action recognition. Generally, target localization could be achieved by estimating the bounding box for the target in image sequences. Visual tracking includes 2-D tracking which estimates 2-D motion parameters, 3-D tracking which gives the positions and orientations of the target in 3-D space, and high DOF tracking which tracks the deformation or articulation of the target. Some difficulties for visual tracking lie in complex backgrounds, unknown lighting conditions, and complex target movements. When multiple objects must be handled simultaneously, the problem becomes even more challenging since different objects would cause occlusion. The robustness, accuracy, and speed are important to evaluate tracking algorithms.

Different image features of the object supply different cues for tracking algorithms. Edge-based approaches match edges in the images of the target against other images, and region-based approaches use image templates for matching. Under the small motion assumption that assumes there is little difference between two consecutive image frames, these approaches could achieve accurate results. However, when this assumption does not hold, which could be very likely in practice, these tracking algorithms will lose track, and the tracking recovery has to depend on some other remedies. At the same time, edge-based and region-based tracking methods generally need more computational resources, which makes real-time systems difficult to achieve.

An alternative is the blob-based approach, which does not use local image information such as edge and region, but represents the target by its color and motion such that the target could be segmented from the background for localization and thus tracking. For example, when we need to localize the human hand in video sequences, it would be very difficult to represent the hand only based on edges and image appearances because the hand is highly articulated due to finger movements, and there are very large variations in hand appearances from different viewpoints. When we notice that the skin color tones of different fingers are uniform, color-based segmentation approaches will afford efficient and robust implementation of real-time localization and tracking of the hand. Skin color is a strong cue in human tracking. Combining these two approaches by integrating multiple visual cues would result in more robust tracking results [74, 75].

Meanwhile, efficient segmentation is also desirable for tracking bootstrapping, in the cases when small motion assumption does not hold, and in tracking reinitialization when it loses track. Recently, some successful tracking systems have been built based on color segmentation [2, 31, 64, 76, 77]. A simple approach is to collect some skin color pixel samples from the user and train a color classifier, such that skin color regions could be segmented from the background by classifying and grouping similar color pixels. To approach the problem of the large variation of skin color for different people, one of the solutions is to tune the color classifier through a very large training data set collected from many people [30]. Unfortunately, in practice, there are still some complications. One of them is that color distributions may change with lighting conditions, in which case a fixed skin color model may not work well at the time. Another difficulty is that collecting such a large labeled training data set is not trivial. We will discuss such issues in more details in the sections.

In this paper, we will study a new representation for color distributions, the SASOM model, for the tasks of adaptive color-based segmentation and nonstationary color tracking. An interesting aspect of such a SASOM neural network is that its structure could be learned through training. An analysis of the stationary status of a self-organizing map neural network will also be given in that section. Based on the SASOM color model, we will present the SASOM transduction algorithm. In contrast to the methods of constructing a specific skin color model, our proposed approach tries to adapt the models to nonstationary color distributions by trans-

ducing a learned color model through image sequences. Section 3.5 will report some of our experiments on the proposed SASOM color model.

3.2 Nonparametric Density Representations

In contrast to parametric representation of density, nonparametric approaches does not assume distribution models and so does not require parameter estimation. Consequently, it is more flexible, since it is able to model complex data distributions. However, the sacrifice of analyzability is the cost of flexibility. At the same time, nonparametric techniques often require a large number of samples.

3.2.1 Histogram and vector quantization

Histogramming is a simple and widely used technique for nonparametric modeling of density. The density of a particular point is represented by the number of samples falling into the volume around this point. The Parzen Window method [78] can be looked at as a generalization of histogramming.

Although histogramming is able to model any density, it is plagued by the *curse of dimensionality*. In a d-dimensional space, the order of bins is $O(n^d)$, where n is the quantization level for each dimension. For an example, we use 255 for each component in color histogram. Then we have 255^2 bins for HS histogramming, and 255^3 for HSV histogramming. A huge data set is in turn needed to obtain such histograms.

Vector quantization (VQ) is a technique to overcome this difficulty by tessellating the data space. Each cell of the tessellation is represented by a code vector. Clustering techniques are often used to achieve such vector quantization, and in turn the representation of data density similar to histogramming.

One of the major factors in histogramming and VQ is the quantization level, or resolution. If the quantization level is large (low resolution), density estimation will be rough but smooth. On the other hand, if the quantization level is small (high resolution), density estimation will be accurate but noisy. It is important to find a good quantization level, but it is difficult to determine it in advance. Many methods have been proposed to deal with this difficulty, such as multiresolution approaches [79, 80] and adaptive approaches.

3.2.2 Self-organizing map

Self-organizing map (SOM) [81] is mainly used for visualizing and interpreting large high-dimensional data sets by mapping them to low-dimensional space based on a competitive learning scheme. SOM consists of an input layer and an output layer. Figure 3.1 shows the structure of 1-D SOM. The number of nodes in the input layer is the same as the dimension of the input vector, while the structure of the output layer can be 1-D or 2-D connected nodes that are connected to each input node with some weights. Through competition, the index of the wining node is taken as the output of SOM. The Hebbian learning rule adjusts the weights of the winning node and its neighborhood nodes. SOM is highly related to vector quantization (VQ) and k-mean clustering. One good characteristic of SOM is its partial data density preservation if properly trained [82, 83, 84, 85].

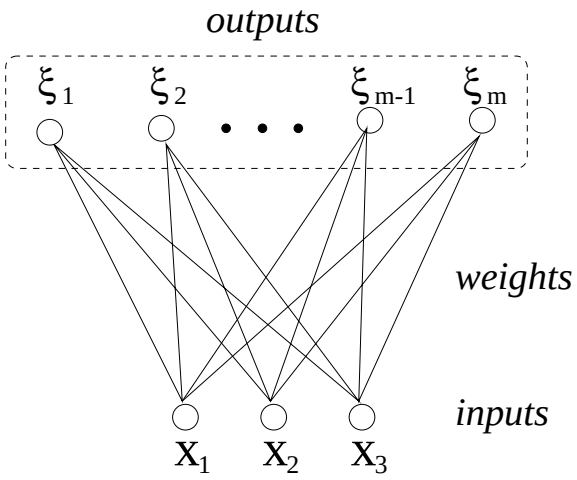


Figure 3.1 1-D SOM structure.

3.2.3 Adaptive self-organizing map

One of the problems of many clustering algorithms is that the number of clusters should be specified in advance. The success of the clustering algorithm depends on the specified number of clusters. It is the same case in the basic SOM algorithm. The more output neurons, the higher the resolution, since output neurons correspond to clusters. Different numbers of clusters lead to different results of tessellation of the pattern space. If fewer neurons are used, data of

lower density will be dominated by the patterns of higher density. On the other hand, if more nodes are used, the ordered mapping is hard to obtain.

One possible approach to this problem is cross-validation. Although the structure of the SOM, such as the number of output neurons, is fixed each time, a good structure can be determined after validating several different structures. However, this approach does not offer flexibility to find an appropriate structure of SOM, and it is not fast. An alternative is to embed some heuristics to dynamically change the structure of SOM in training [86, 87, 88, 89]. Our algorithm can automatically find an appropriate number of clusters by the schemes of growing, pruning, and merging [43, 90].

Growing Scheme: Our algorithm is also a competitive learning scheme which deals with the problem of how to find the competition winner. In the SOM algorithm, the output of a node is the distance between the input vector and the weight vector of the node. The distance measurement can be defined as

$$\mathcal{D}(\mathbf{x} - \mathbf{w}_i) = ||\mathbf{x} - \mathbf{w}_i|| \tag{3.1}$$

where $\mathcal{D}$ is a distance measurement between the input vector $\mathbf{x}$ and the weight vector $\mathbf{w}_i$ of the ith neuron of SOM. We call it the output value of an output neuron. The measurement here is the Euclidean distance; however, other distance measurements could also be employed.

In standard SOM, the neuron with the smallest output value is taken as the winner c.

$$c = arg \ \min_i \mathcal{D}(\mathbf{x} - \mathbf{w}_i) \tag{3.2}$$

In some cases, however, when the output values of all neurons become nearly the same, determining the winner by finding the one with the smallest value is not suitable. In this situation, the input vector may be too far from every weight vector or around the center of the convex hull of the weight vectors. If the current input sample is drawn from the data cluster represented by one of these output neurons, the weight vector of that neuron will be misplaced unnecessarily in training by adjusting the weight. So, it is not a robust way to take the neuron with the smallest output value as the winner. In this situation, a new neuron could be generated by inserting it to the current structure and taking the input vector as its initial weight, which is illustrated in Figure 3.2.

By comparing the mean value and the median value of the output values of all neurons, we make a rule to detect the situation in which a new neuron should be created. The competition

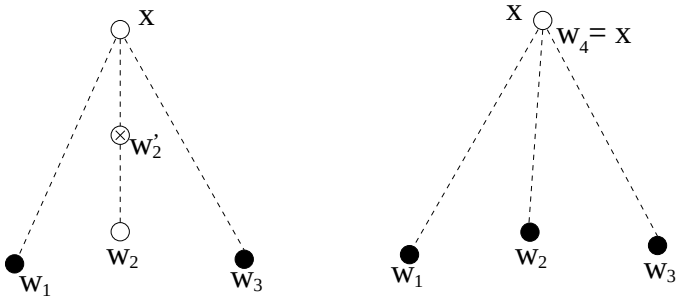


Figure 3.2 Growing scheme of SASOM. $\mathbf{w}_i$ is the weight vector, and $\mathbf{x}$ is an input vector. Left: when the input vector is too far from every weight vector so that the output value of all neurons are nearly the same, if current input $\mathbf{x}$ is in the data cluster represented by one of these neurons, say $\mathbf{w}_2$, the weight vector of that neuron will be misplaced unnecessarily to $\mathbf{w}_2'$. Right: in this situation, a new neuron is created and its weight will be set by $\mathbf{w}_4 = x$.

can be described as

$$v_i = \mathcal{D}(\mathbf{x} - \mathbf{w}_i) \ \forall i \in \{1, \ldots, M\} \tag{3.3}$$

where v_i is the output value of the ith neuron with weight vector $\mathbf{w}_i$, and M is the number of neurons. The competition winner can be found by:

$$c = \begin{cases} NULL, & \text{if } \texttt{mean}(\mathbf{v}) \approx \texttt{median}(\mathbf{v}) \\ arg\ min_i\, v_i, & \text{otherwise} \end{cases} \tag{3.4}$$

where $\mathbf{v} = \{v_1, v_2, \ldots, v_M\}$ and M is the current number of neurons.

Pruning Scheme: In the training process, when a neuron is rarely a winner, it means that the data cluster represented by this neuron has very low density or might be taken as noise. So, such a neuron can be pruned. In practice, a threshold is set to determine such neurons.

Merging Scheme: In the training process, the distances between any two weight vectors of each two neurons are calculated. If any two weight vectors are near enough, we can merge these two neurons by assigning the average of the two weights to the new one.

The algorithm of SASOM is summarized in Figure 3.3.

We perform color-based image segmentation based on the SASOM color model. In our segmentation algorithm, the training data set is collected from one color image, and each data vector is a weighted HSI vector, i.e., $\mathbf{x} = \{\ \alpha H,\ \beta S,\ \gamma I\}$, where we set $\alpha = \ \beta = 1$ and $\gamma = 0.1$. Pixels with large and small intensities are not included in the training data set because hue and saturation become unstable in this range. Once trained, the SASOM is used to label each pixel by its HSI value. The pixel label is the index of the node in the SASOM.

- Initially set the number of neurons M to 2, and randomly initialize the weights $\mathbf{w}_i = \mathbf{w}_i(0), i = \{1, 2\}$, where $\mathbf{w}_i(k)$ represents the weight vector of the ith node at the kth iteration.
- Draw an input $\mathbf{x}$ from the training sample set randomly to the SASOM.
- Find the winner among the neurons using Equation 3.4.
- If(winner!=NULL), adjust the weights of the winner neuron c and its two neighborhood neurons $c-1$ and $c+1$.

$$\begin{aligned}\mathbf{w}_c(k+1) &= \mathbf{w}_c(k) + \ \eta(k)(\mathbf{x} - \mathbf{w}_c(k)) \\ \mathbf{w}_{c-1}(k+1) &= \mathbf{w}_{c-1}(k) + \ \eta(k)\ \alpha(k)(\mathbf{x} - \mathbf{w}_{c-1}(k)) \\ \mathbf{w}_{c+1}(k+1) &= \mathbf{w}_{c+1}(k) + \ \eta(k)\ \alpha(k)(\mathbf{x} - \mathbf{w}_{c+1}(k))\end{aligned}$$

  where $\eta(k)$ is the step size of learning, $\alpha(k)$ is a neighborhood function, k is the counter of iteration.
- If there is no winner, grow a new neuron m according to the growing scheme. $\mathbf{w}_m(k+1) = \mathbf{x}$ and set $M = M + 1$.
- If a neuron is rarely win, delete it according to pruning scheme, $M = M - 1$.
- Calculate the distance between each two neurons and perform merging scheme.

Figure 3.3 The training algorithm of the structure adaptive self-organizing map.

3.3 Model Transduction and Color Model Adaptation

It is a good practice to learn a generic color classifier for color-based segmentation by collecting a large labeled data set [30]. If some color invariants could be found, learning such a color classifier would suggest a direct and robust way of color tracking. However, when we consider the nonstationary color distribution over time, we do not generally expect to find such invariants.

The approach taken in [30] is an *inductive learning* approach, by which the color classifier learned should be able to classify any pixel in any image. Generally, this color classifier would be highly nonlinear, and a huge labeled training data set is required to achieve good generalization. In fact, learning such a highly nonlinear color classifier for all lighting conditions and all images may not be necessary because the requirement of the generalization could be relaxed to a subset of the data space. In color tracking, a color classifier M_t at time frame t could only be used to classify pixel $\mathbf{x}_j$ in the current specific image feature data set I_t so that this specific classifier

M_t could be simpler. When there is a new image I_{t+1} at time $t+1$, this specific classifier M_t should be *transduced* to a new classifier M_{t+1} which works just for the new image I_{t+1} instead of I_t. The classification can be described as

$$y_i = arg \max_{j=1,\ldots,C} p(y_j|\mathbf{x}_i, M_t, I_{t+1} : \forall \mathbf{x}_i \in I_{t+1}) \tag{3.5}$$

where y_i is the label of $\mathbf{x}_i$, and C is the number of classes. In this sense, we do not care about the performance of the classifier M_{t+1} outside I_{t+1}. We call the problem of transducing the classifier M_t to M_{t+1} given I_{t+1} *model transduction*. Figure 3.4 shows the transduction of color classifiers.

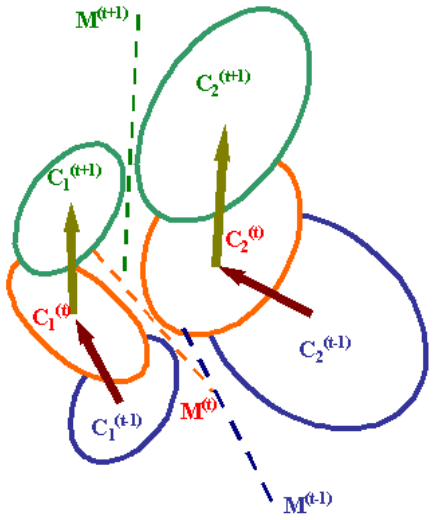


Figure 3.4 An illustration of transduction of classifiers.

This model transduction may not always be feasible unless we know the joint distribution of I_t and I_{t+1}. Unfortunately, such joint probability is generally unknown since we may not have enough *a priori* knowledge about the transition in a color space over time. One approach is to assume a transition model, like the case in motion tracking by Kalman filter or Condensation [35], so that we can explicitly model $p(I_{t+1}|I_t)$. One of the difficulties of this approach is that a fixed transition model is unable to capture much dynamics. Although the issue of motion model switching by learning transition models has been addressed in [35], their scheme is not general. Another difficulty is that it may not be easy to identify parameters of the transition models due to the insufficient labeled training data. The approach used in [32] assumes

a linear transition model. However, the transition (updating) of color models is plagued since the newest image has not been segmented yet.

However, our assumption is different from the transition model assumption. We assume that the classifier M_t at time t can give "confident" labels to several samples in I_{t+1}, so that the data in I_{t+1} can be divided into two parts: labeled data set $\mathcal{L} = \{(\mathbf{x}_j, y_j), j = 1, \ldots, N\}$, and unlabeled set $\mathcal{U} = \{\mathbf{x}_j, j = 1, \ldots, M\}$, where N and M are the size of the labeled set and unlabeled set, respectively, $\mathbf{x}_j$ is the color feature vector, and y_j is its label (such as skin or non-skin). Here, $\mathcal{L}$ and $\mathcal{U}$ are from the same distribution. Consequently, the transductive classification can be written as

$$y_i = arg \max_{j=1,\ldots,C} p(y_j|\mathbf{x}_i, \mathcal{L}, \mathcal{U} : \forall \mathbf{x}_i \in \mathcal{U}) \tag{3.6}$$

In this formulation, the specific classifier M_t is transduced to another classifier M_{t+1} by combining a large unlabeled data set from I_{t+1}.

At time t, we have a classifier model $M^t = p^t(\mathbf{x}, y)$, which is learned from a fully labeled data set $(\mathcal{X}^t, \mathcal{Y}^t) = \{\mathbf{x}_k^t, y_k^t\}$, where $k = 1, \ldots, N^t$. The density $p^t(\mathbf{x})$ and the conditional density $p^t(y|\mathbf{x})$ are known. And at time $t+1$, we just have a set of unlabeled data $\mathcal{X}^{t+1} = \{\mathbf{x}_k^{t+1}\}$, where $k = 1, \ldots, N^{t+1}$. At this point, we only know $p^{t+1}(\mathbf{x})$. The conditional density $p^{t+1}(y|\mathbf{x})$ is given by transduction. And a new classifier model $M^{t+1} = p^t(\mathbf{x}, y)$ should be learned from M^t and $\mathcal{X}^{t+1}$. We have

$$p(M^{t+1}|\mathcal{X}^{t+1}, M^t) = \frac{p(\mathcal{X}^{t+1}|M^{t+1})p(M^{t+1}|M^t)}{p(\mathcal{X}^{t+1}|M^t)} \tag{3.7}$$

$$\propto p(\mathcal{X}^{t+1}|M^{t+1})p(M^{t+1}|M^t) \tag{3.8}$$

The term of $p(\mathcal{X}^{t+1}|M^{t+1})$ can be calculated by the density estimation techniques. SOM and LVQ can be easily applied. However, the term of $p(M^{t+1}|M^t)$ is difficult to deal with because it involves the dynamics from M^t to M^{t+1}, if we do not have an explicit model for such transition. Without using any transition model, we have to at least make some assumptions about the transition; otherwise, the transduction does not make sense, since such transition can be arbitrary.

Here, we assume that M^t can confidently give some true observations O^{t+1} (or classify some samples which consist of their true labels) at time $t+1$. This assumption is reasonable, since it means that the difference between M^t and M^{t+1} should be small and the transition should

not be large. So, we have

$$p(M^{t+1}|M^t) \propto p(M^{t+1}|O^{t+1}, \mathcal{X}^{t+1}) \tag{3.9}$$

$$= p(M^{t+1}|\mathcal{X}^{t+1}, O_y^{t+1}) \tag{3.10}$$

which means we should estimate $p(M|\mathcal{X}, O_y)$.

Since the labels $\mathcal{Y}$ for $\mathcal{X}$ are hidden variables, basically, an EM-like iteration can be taken to fulfill such a transition.

- *Step 1*: At the kth iteration, find $\mathcal{Y}(k) = argmax_Y\, p(\mathcal{Y}|M(k-1), \mathcal{X})$
- *Step 2*: Learn $M(k) = p(\mathcal{X}, \mathcal{Y}(k))$

3.4 SOM Transduction Algorithm

Our solution to this problem is called *transduction of SOM*, which is to update the weights and structure of the trained SOM according to a set of new training data so that the transduced SOM captures the new distribution. The new training data set in the transduction consists of both labeled and unlabeled samples. The algorithm is described below.

- $\mathcal{W}^{(n-1)} = \{\mathbf{w}_i^{(n-1)}, i = 1, \cdots, C^{(n-1)}\}$ are the weights of SOM at time frame $n-1$. The training data set $\mathcal{X}^{(n)} = \{\mathbf{x}_k^{(n)}, k = 1, \cdots, N\}$ is drawn randomly from the image at time frame n. We use $\mathcal{W}^{(n)}$ to represent SOM at time frame n.
- The training data set $\mathcal{X}^{(n)}$ is classified by the SOM $\mathcal{W}^{(n-1)}$ and is partitioned into two parts: a labeled data set $\mathcal{X}_l^{(n)}$ and an unlabeled data set $\mathcal{X}_u^{(n)}$. If a sample $\mathbf{x}_k^{(n)}$ is confidently classified by $\mathcal{W}^{(n-1)}$, then put this sample to the set $\mathcal{X}_l^{(n)}$ and label it with the index of the winning neuron of $\mathcal{W}^{(n-1)}$; otherwise, put it to $\mathcal{X}_u^{(n)}$ and let it unlabeled.
- **Unsupervised updating**: The algorithm described in Section 3.2.2 is employed to update $\mathcal{W}^{(n-1)}$ by the unlabeled data set $\mathcal{X}_u^{(n)}$.
- **Supervised updating**: The labeled data set $\mathcal{X}_l^{(n)}$ is used in this step. $(\mathbf{x}_k, l_k)$ is drawn from $\mathcal{X}_l^{(n)}$, where l_k is the label for $\mathbf{x}_k$. The winning neuron for the input $\mathbf{x}_k$ is c.

$$\mathbf{w}_c^{(n)} = \begin{cases} \mathbf{w}_c^{(n-1)} + \alpha(\mathbf{x}_k - \mathbf{w}_c^{(n-1)}), & if\ c = l_k \\ \mathbf{w}_c^{(n-1)} - \alpha(\mathbf{x}_k - \mathbf{w}_c^{(n-1)}), & if\ c \neq l_k \end{cases}$$

After several iterations, the SOM at time frame $n-1$ is transduced to n.

3.5 Experiments Based on SASOM

Our color segmentation algorithm based on SASOM has been tested with a large variety of pictures. And our localization system that integrates this color segmentation algorithm has run under a wide range of operating conditions. Such a real-time system has been employed in vision-based gesture analysis [23]. Extensive experiments show that our color segmentation algorithm is fast, automatic, and accurate, and that the proposed localization system is robust, real-time, and reliable. This color segmentation algorithm can also be applied to other segmentation tasks.

3.5.1 Performance of image segmentation

One parameter we should specify in SASOM is the maximum number of neurons. If the scene is simple, we set the maximum number to 2 or 3. If the scene is complex, we set it to 10 or more. In between, we use 6.

Figure 3.5 Some results of color-based image segmentation using structural adaptive self-organizing map. Left column: source color images; middle column: segmented images; right column: interested color regions.

Figure 3.5 shows some segmentation results. In the left column are source color images, in the middle column are segmented images, and in the right column are separated color regions. The colors of segmented color regions are the average colors of these regions. Each pixel in the

source images is assigned a label by the SASOM algorithm, and this label is used as a mask to separate the corresponding color region. Our segmentation algorithm works well through these experiments. When the background has less color distraction, this algorithm finds exact color regions. Since texture is not used in the segmentation, segmentation results will be noisy when there is color distracter texture in the background. Hand and face images are taken from a cheap camera in the indoor environment in our labs. Our algorithm can also successfully segment hand and face regions.

3.5.2 Performance of hand tracking

A typical hand-tracking scenario is controlling the display or simulating a 3-D mouse in desktop environments. A camera mounted at the top of the desktop computer looks below at the keyboard area to give an image sequence of moving a hand. Another typical application is to track a human face. Our localization system is able to simultaneously localize multiple objects, which is useful in tracking a moving human.

Since our localization system is essentially based on a global segmentation algorithm, it does not largely rely on the tracking results of previous frames. Even if for some reason the tracker gets lost in some frames, it can recover by itself without interfering with the subjects. In this sense, the tracking algorithm is very robust.

Our proposed system can handle changing lighting condition to some extent because of the transduction of the SASOM color classifier. At the same time, since the hue and saturation are given more weight than intensity, our system is insensitive to changes in lighting intensity such as when objects are shadowed or the intensity of the light source changes. However, there are still some problems. Insufficient lighting, too strong lighting, or very dark or bright backgrounds may cause trouble for the color segmentation algorithm, since hue and saturation become unstable and the system does not give more weight to intensity. If the lighting condition changes dramatically, the color segmentation algorithm may fail since the transduction cannot be guaranteed.

Some hand tracking results in our experiments are given in Figure 3.6. In this experiment, a hand is moving around with the interference of a moving book. The book is also shading the

light so that the color of the skin is changing. The blue boxes are the bounding boxes of the interested color region.[1]

Figure 3.6 Results of hand tracking with 18 frames taken from image sequences. A moving hand with interference from a book is localized. The blue boxes are the bounding box of the interested color region.

Our tracking system is very robust and efficient from this experiment in which the background of the scene is cluttered. Since a book is interfering with the hand by shading the light, such a system can still find a correct bounding box. Sometimes, due to the sudden change of lighting conditions, the tracker may be lost. However, it can quickly recover to continue working. Different skin tones do not affect our system. The first image with the interested

[1] A demo sequence can be downloaded at `http://www.ifp.uiuc.edu/~yingwu`.

color region is used to initially train the SASOM so that it can work with nearly any users, which has been tested in our extensive experiments.

3.6 Discussion

Computer vision techniques supply good ways to improve human-computer interaction by visually understanding the human movements, which requires a robust and accurate tracking of parts of the human body, such as hands and face. However, cluttered backgrounds, unknown lighting conditions, and multiple moving objects make the tracking tasks challenging. This chapter mainly concentrated on color-based image segmentation and color-based target tracking.

We presented a new representation of color models based on the proposed structure adaptive self-organizing (SASOM) neural network, in which the structure of the SOM could be trained. This SASOM representation could afford efficient image segmentation through a competitive process of the neurons in SASOM. Then we also investigated the color-based tracking task. A challenge of such a task lies in the fact that the lighting conditions and the background may not be static, such that the color distributions in the image sequence are not stationary. In order to capture the nonstationary color distribution, the SASOM could be transduced by combing supervised and unsupervised learning paradigms, which we called SASOM transduction. Based on the SASOM model, we achieved a robust real-time tracking system that has been widely used in our further research.

Besides the SASOM color model, we also looked into a parametric approach by using the Gaussian mixture model. Since the nonstationary color tracking could be formulated as a model transduction problem, our study focused on the problem of learning a new Gaussian mixture model based on an old one and a set of training data, e.g., color pixel data. Instead of assuming a transition model, we assume that some unlabeled pixels in a new image frame can be "confidently" labeled by a "weak classifier" according to a preset confidence level.

We noticed that the SASOM transduction is not mature, and it needs more efforts to find a better way to combine supervised and unsupervised learning schemes. Since the process of competition among all neurons is essentially parallel, the tracking system can be made much faster by parallel implementation of the competition process. Currently, our localization system

offers a bounding box of the interested objects. Shape analysis of localized objects will be extended to estimate the 3-D motion of the target.

CHAPTER 4

MULTIPLE CUES INTEGRATION FOR ROBUST TRACKING

4.1 Introduction

With the rapid enhancement of computational power provided by computer hardware, it becomes easier to afford some visual capacities for computers. Recent years has witnessed an expeditious development of the research and applications of visual surveillance and vision-based interfaces. Visual tracking is an important part of such topics. The vision systems should be able to localize moving targets from video sequences. A large amount of research effort has been devoted to visual tracking and analysis of human motion [3, 6, 7]. More natural and immersive human computer interactions could be developed by visually recognizing and interpreting human actions. Therefore, to afford such interactions, human motion should be tracked visually to provide sensory inputs for recognition.

One of the purposes of visual tracking is to infer the *states* of the targets from image sequences. Besides 2-D positions, visual tracking could also be expected to recover other states, such as poses, articulation or deformations, which depend on different applications.

Although the tracking problem can be well formulated in control and signal processing research, for visual tracking, it involves much more fundamental research problems such as object representation, image measurement and matching. Since the states of the target are hidden and can only be inferred from some observable image features, one of the difficulties for visual tracking lies in the fact that it is crucial but difficult to measure state hypotheses against image evidence.

Bottom-up and *top-down* approaches are two kinds of methodologies to approach the visual tracking problem. *Bottom-up* approaches generally tend to construct object states by analyzing the content of images. Basically, many segmentation-based methods can be categorized as bottom-up approaches. For example, blob tracking techniques group similar image pixels into blobs to estimate the positions and shapes of the target. In contrast, top-down approaches generate candidate hypotheses of the target state from previous time frames based on a parametric representation of the target. Tracking is achieved by measuring and verifying these hypotheses against image observations. Many model-based and template-matching methods can be categorized as top-down approaches. Certainly, bottom-up approaches and top-down approaches could be combined.

Bottom-up methods might be computationally efficient, yet the robustness is largely limited by the ability of image analysis, because the processing of grouping or tracing image pixels could be overwhelmed by image clutters. On the other hand, top-down approaches depend less on image analysis because the target hypotheses serve as strong constraints for analyzing images. But the performances of the top-down approaches are largely determined by the methods of generating and verifying hypotheses. To achieve robust tracking, a large number of hypotheses may be maintained so that more computation would be involved for measuring them. The combination of these two methodologies could keep the robustness but reduce the computation.

Visual tracking techniques generally have four elements: *target representation*, *observation representation*, *hypotheses generating*, and *hypotheses measurement*, which roughly characterize tracking performances and limitations.

To discriminate the target from other objects, the target representation — which could include different modalities such as shape, geometry, motion, and appearance — characterizes the target in a *state space* either explicitly or implicitly. Although how to find the target representation is a fundamental problem in computer vision, visual tracking research generally employs concise representations to facilitate computational efficiency. For example, parameterized shapes [74, 91] and color distributions [32, 72, 92, 93] are often employed as target representations. To provide a more constrained description of the target, some methods employ both shape and color [33, 34, 74, 77, 94, 95]. Obviously, unique characterization of the target would be quite helpful to visual tracking, but in general it would involve high dimensionality. To add uniqueness in the target representation, many methods even employ the target's

appearance, such as image templates [96, 97, 98] or eigenspace representation [99], as the target representation. For example, if you know a person, it would be a bit easier to track this person in a crowd. Sometimes motion could also be taken into account in target representations, since different objects can be discriminated by the differences of their motions. On the other hand, if two objects share the same representation, it would be difficult to correctly track either of them when they are close in the *state space*, if there are no priors from the dynamics of the targets' movements.

Closely related to the target representation, the observation representation defines the image evidence of the object representation, i.e., the image features observed in the images. For example, if the target is represented by its contour shape, we expect to observe edges of the contour in the image. If the target is characterized by its color appearance, certain color distribution patterns in the images could be used as the observation of the target.

The hypotheses measurement evaluates the matching between target state hypotheses and their image observations. Sometime, we will measure with how much probability the state hypotheses will generate such image observations. Similarly, the question we often ask is that, given a certain image observation, which hypothesis will be most likely to produce such an image observation. In many cases, we cannot obtain exact probability measurements, but we can approximate them by pseudo-probabilities. For example, the template-matching tracking method often takes SSD as the measurement. The fewer the SSD measurement, the higher the probability of the hypothesis. The evaluation would be quite challenging when measuring a shape hypothesis against a cluttered background. Although some analytical results were reported in [35], many current tracking methods take ad hoc measurements.

The hypotheses generating is to produce new state hypotheses based on old estimation of target's representation and old observation, which implies the evolution of the dynamic process. The target's dynamics could be embedded in such a predicting process. At a certain time instant, the target state is a random variable. The *a posteriori* probability distribution of the target state, given the history of observations, changes with time. Therefore, the tracking problem can be viewed as a problem of conditional probability density propagation. The estimation of the target state at a certain time instance could be approximated by estimating the conditional probability density of it. The hypotheses generating basically describes the evolution of such posterior density or some of its statistics. The Kalman filtering technique gives a classical

example of hypotheses generating under Gaussian assumptions, due to which the density could be characterized by its mean and covariance. In this case, hypotheses generating characterizes the search range and confidence level of the tracking. If the Gaussian assumption does not hold, which is often the case in tracking against cluttered backgrounds, we could represent the posterior density of the target state by a nonparametric form. In this circumstance, the hypotheses generating could be viewed as a process of the evolution of a set of hypotheses or state samples or particles, which facilitates a Monte Carlo approach to tracking. The CONDENSATION [35] algorithm is one such example.

Since image sequences contain very rich visual information, using single object representation would not be robust when the target is in a clutter. For example, if we only use shape models to represent the target and measure it against edges in images, the tracking could be quite unstable when the target moves to a clutter. The false edges incurred by the clutter would very likely distract the tracker.

To approach the problem of robust tracking and integrating multiple cues, this chapter formulates it as the inference problem of a factorized graphical model. Due to the complexity of such a graphical model, a *variational method* is taken to approximate the Bayesian inference. Different modalities in the model present a *co-inference* phenomenon [75]. Based on the analysis of the factorized model, this paper presents an efficient Monte Carlo tracking algorithm to integrate multiple visual cues, in which the co-inference of different modalities is achieved by the EM iterations.

In this chapter, we will give a brief overview of the research of multiple cues integration for robust tracking in Section 4.2. Then the factorized graphical model used in our tracking formulation will be presented in Section 4.3, and the co-inference phenomenon will be analyzed and explained in this section as well. Section 4.4 describes the techniques in sequential Monte Carlo approaches for tracking problems. The importance sampling technique will be described in more details. Our proposed approach and our implementation of co-inference based on transductive importance sampling will be presented in Section 4.5, and the details of our tracking implementation and experiments are described in Section 4.7. Section 4.8 discusses the proposed approach and points out some possible directions for future research.

4.2 Multiple Cues Integration

Based on the four elements of visual tracking described in Section 4.1, we often notice that a state hypothesis of a richer target representation would have better opportunities to be verified according to various aspects of image observations. For example, combining color distribution of the target could largely enhance the robustness of contour tracking in a heavily cluttered background, and integrating shape and color representations could incur better tracking against color distracters. On the other hand, if the target and the clutter are indistinguishable in terms of their representations, the tracking has to be almost determined by the prior knowledge about the dynamics at least in a short period of time. If such prior dynamics is not available, or we assume a random walk for the target, we could say that the target is *untrackable* in terms of 2-D. This aperture problem motivates the research of tracking and integrating multiple visual cues.

On the other hand, we often observe that an accurate dynamics model is quite important for a successful tracking for many existing visual tracking algorithms because it generally provides the tracker a good prediction of the target. As an approximation, we often assume constant velocity models or constant acceleration models for the target's dynamics for a short period of time. In many cases, the parameters of such dynamic models are set in advance. Tracking will be at risk when the dynamic model is not properly set. An interesting question is whether there is a way to enhance the robustness even if the dynamic model is not accurate enough.

According to our investigation, the answer is yes, and the solution is to use a richer target representation and integrate multiple visual cues, which will be presented in the following sections of this chapter. Some intuitions are given here. Suppose we represent the target by two modalities: shape and color appearance. The two modalities have their own dynamics models, which means that the target is deformable, and the lighting could change, but it is difficult to know in advance about how the shape will deform and how the lighting will change. Therefore, we can only assume very rough dynamics models for approximation. However, such rough dynamics models will be sometimes violated such that the predictions based on the dynamics models could largely deviate, causing tracking to fail. The reason for this is that the two modalities are treated separately. Our main idea is to let the two modalities interact with each other. For example, if the target's shape changes very little but the lighting changes a lot,

the estimation of the target's color appearance could be fulfilled by taking advantage of shape estimation, instead of relying on the dynamics models of color appearance. On the other hand, if the lighting changes very slowly but the target deforms a lot, the deformation could be more robustly localized by looking into its color appearance, instead of taking a strong prior from the prediction from the dynamics model of the deformation. Naturally, we shall ask, What if both deformation and lighting changes are quite large? As described before, the problem becomes a sort of untrackable problem, but we can still approach it by taking the estimation that maximizes the joint probability of both modalities. Exact formulation and analysis will be given later in this paper.

Multiple cues integration could be done in terms of object representation and observation representation. Some approaches perform multiple observation measurements and accumulate the measurements for each hypothesis [34, 94]. Although robust to some extent, many methods of combining the measurements from different sources are often ad hoc. To integrate shape and color, many tracking algorithms assume fixed color distribution [34, 74, 77] for the target to enable efficient color segmentation. However, such an assumption is often invalid in practice. In the literature, nonstationary color tracking methods [32, 72] were also reported. Instead of assuming fixed color representation, some methods also include color modality in the target representation [33, 59, 95], in which a multivariable Gaussian was used to represent both color and motion parameters. To enable robust tracking and the capacity of reinitialization, some researchers investigated the combination and switch among different trackers that track different modalities [77, 100]. Generally, different modalities are updated sequentially in these methods. On the other hand, tracking both shape and color simultaneously would be a formidable problem, since it increases the dimensionality of the state space of the target. This paper tries to investigate the relationship among different modalities in visual tracking, and to identify an efficient way to facilitate robust and simultaneous tracking of these modalities.

4.3 Graphical Models for Tracking

In this section, we shall formulate the visual tracking problem in a probabilistic framework. The integration of multiple cues could be characterized by a factorized graphical model, in which we make use of the variational analysis to approximate the inference task.

In a dynamic system, the states of the target and image observations are represented by $\mathbf{X}_t$ and $\mathbf{Z}_t$, respectively. The history of states and measurements are denoted by $\underline{X}_t = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$ and $\underline{Z}_t = (\mathbf{Z}_1, \ldots, \mathbf{Z}_t)$. The tracking problem could be formulated as an inference problem with the prior $p(\mathbf{X}_{t+1}|\underline{Z}_t)$, which is a prediction density. We have

$$\begin{aligned} p(\mathbf{X}_{t+1}|\underline{Z}_{t+1}) &\propto p(\mathbf{Z}_{t+1}|\mathbf{X}_{t+1})p(\mathbf{X}_{t+1}|\underline{Z}_t) \\ p(\mathbf{X}_{t+1}|\underline{Z}_t) &= \int p(\mathbf{X}_{t+1}|\mathbf{X}_t)p(\mathbf{X}_t|\underline{Z}_t)d\mathbf{X}_t \end{aligned}$$

where $p(\mathbf{Z}_{t+1}|\mathbf{X}_{t+1})$ represents the *measurement* or *observation* likelihood, and $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$ is the dynamic model.

The probabilistic formulation of the tracking problem could be represented by graphical models in Figure 4.1, which is similar to the hidden Markov model [101]. At time t, the observation $\mathbf{Z}_t$ is independent of previous states $\underline{X}_{t-1}$ and previous observations $\underline{Z}_{t-1}$, given current state $\mathbf{X}_t$, i.e., $p(\mathbf{Z}_t|\underline{X}_t, \underline{Z}_{t-1}) = p(\mathbf{Z}_t|\mathbf{X}_t)$, and the states have Markov property, i.e., $p(\mathbf{X}_t|\underline{X}_{t-1}) = p(\mathbf{X}_t|\mathbf{X}_{t-1})$.

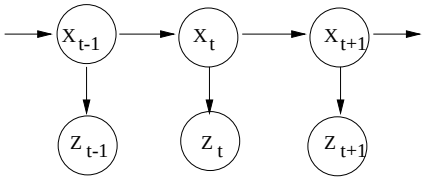


Figure 4.1 The tracking problem could be represented by a graphical model, similar to the hidden Markov model.

The tracking problem can be approached by the inference techniques in the graphical model. Consequently, when the dimensionality of the hidden states increases, the inference and learning would become difficult due to the exponential increase of required computational resources. However, a distributed state representation could largely ease this difficulty by decoupling the dynamics. Fox example, target states could be decomposed into shape states and color states with the architecture shown in Figure 4.2(a).

Furthermore, the observation could also be separated into $\mathbf{Z}_t^s$ and $\mathbf{Z}_t^c$ for shape and color respectively in Figure 4.2(b). Each observation depends on both color and shape states.

Due to the complex structure of the factorized network, the exact inference would be formidable. One approach to this problem is statistical sampling-based methods, such as Gibbs sampling. Another approach is to approximate the posterior probability $p(\underline{X}_t|\underline{Z}_t)$ of the hidden

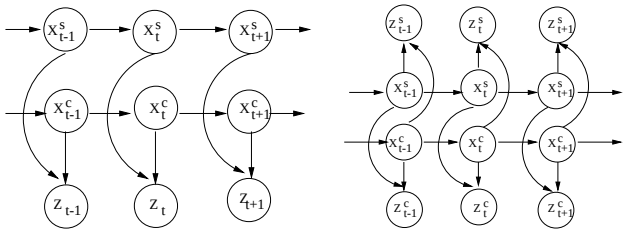


Figure 4.2 Factorized graphical models: (a) The states of the target could be decomposed into shape states $\mathbf{X}_t^s$ and color states $\mathbf{X}_t^c$ in a factorized graphical model. (b) The observation could also be separated into $\mathbf{Z}_t^s$ and $\mathbf{Z}_t^c$.

states by a tractable distribution $Q(\underline{X}_t)$. A lower bound on the log likelihood $\log P(\underline{Z}_t)$ can be achieved by such an approximation [102, 103, 104, 105]:

$$\log P(\underline{Z}_t) \geq \sum_{X_t} Q(\underline{X}_t) \log \frac{P(\underline{X}_t, \underline{Z}_t)}{Q(\underline{X}_t)} \tag{4.1}$$

$$KL(Q||P) = \sum_{X_t} Q(\underline{X}_t) \log \frac{Q(\underline{X}_t)}{P(\underline{X}_t|\underline{Z}_t)} \tag{4.2}$$

Generally, we can choose $Q(\cdot)$ to have a simpler structure by eliminating some of the dependences in $P(\cdot)$, while minimizing the Kullback-Leibler divergence between $P(\cdot)$ and $Q(\cdot)$ in Equation (4.2). It can be achieved by a *structured variational inference*. The basic idea is to uncouple the Markov chains and replace the true observation probability of each hidden state by a distinct variational parameter, which can be varied for the minimization. Supposing the target state includes M modalities, we could write

$$\begin{aligned} Q(\underline{X}_t|\theta) &= \frac{1}{Z_Q} \prod_{m=1}^{M} Q(\mathbf{X}_1^m|\theta) \prod_{t=2}^{T} Q(\mathbf{X}_t^m|\mathbf{X}_{t-1}^m, \theta) \\ &= \frac{1}{Z_Q} \prod_{m=1}^{M} h_{x_1}^m \pi^m \prod_{t=2}^{T} h_{x_t}^m p(\mathbf{X}_t^m|\mathbf{X}_{t-1}^m) \end{aligned}$$

where M is the number of modalities or factorized Markov chains, $\mathbf{X}_t^m$ is the state of the mth modality at time frame t, $h_{x_t}^m$ are the variational parameters, and Z_Q is a normalization constant. Although general continuous analysis of such an approach is unavailable, the method in [103] employed a structured variational analysis for the case of discrete hidden state and observation. Under the assumption of linear observation, a set of fixed point equations for $h_{x_t}^m$ to minimize $KL(Q||P)$ was obtained [103]:

$$\widetilde{h_{x_t}^m} = g(\mathbf{Z}_t, \{E[\mathbf{X}_t^n|\underline{Z}_t^n, h^n] : \forall n \neq m\}) \tag{4.3}$$

where $g(\cdot,\cdot)$ is a function whose details can be found in the Appendix A, $E[\mathbf{X}_t^n|\underline{Z}_t^n, h^m] \equiv \langle \mathbf{X}_t^m \rangle$ is the estimation of the hidden state $\mathbf{X}_t^n$ at the nth uncoupled Markov chain, based on the variational parameters h^n. Using these variational parameters, a new set of expectations for the hidden states $\langle \mathbf{X}_t^m \rangle$ will be fed back into Equation (4.3), which can be solved iteratively. It is very similar to the EM algorithm [106]. To make it clear, we could explicitly write up in Equation (4.4) the fixed point equations in Equation 4.3 for the case of two modalities, for example, shape and color:

$$\begin{cases} \widetilde{h_{x_t}^s} &= g(\mathbf{Z}_t, E[\mathbf{X}_t^c|\underline{Z}_t^c, h^c]) \\ \widetilde{h_{x_t}^c} &= g(\mathbf{Z}_t, E[\mathbf{X}_t^s|\underline{Z}_t^s, h^s]) \end{cases} \tag{4.4}$$

where $\mathbf{X}_t^s$ is the shape state, $\mathbf{X}_t^c$ is the color state, and $h_{x_t}^s$ and $h_{x_t}^c$ represent the shape and color variational parameters, respectively.

It should be noted that the original densely connected graphical model is uncoupled. The hidden states of each uncoupled Markov chain could be estimated separately, given the set of variational parameters. The estimation of the variational parameters of one of the chains depends on the hidden states of the other chains. Such a phenomenon becomes quite clear in Equation (4.4), where in the iteration of the fixed point equations, the estimation of shape state $E[\mathbf{X}_t^s|\underline{Z}_t^s, h^s]$ is used to calculate the variational parameters of color modality $\widetilde{h_{x_t}^c}$, and the estimation of color state $E[\mathbf{X}_t^c|\underline{Z}_t^c, h^c]$ is used to calculate the variational parameters of shape modality $\widetilde{h_{x_t}^s}$. We call such interesting phenomenon *co-inference* [75], since one modality could be inferred iteratively by other modalities.

The variational analysis of the factorized model is meaningful for the problem of multiple cues integration, since it reveals the interactions among different modalities. It thus suggests an efficient approach to track multiple cues, which will be presented in Section 4.5.

4.4 Sequential Monte Carlo Techniques

As described in previous sections, the visual tracking problem could be formulated in a probabilistic framework by representing tracking as a process of conditional probability density propagation. Denote the target state and observation at time t as $\mathbf{X}_t$ and $\mathbf{Z}_t$, respectively, and $\underline{X}_t = \{\mathbf{X}_1, \ldots, \mathbf{X}_t\}, \underline{Z}_t = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_t\}$. The tracking problem is formulated as

$$p(\mathbf{X}_{t+1}|\underline{Z}_{t+1}) \propto p(\mathbf{Z}_{t+1}|\mathbf{X}_{t+1}) p(\mathbf{X}_{t+1}|\underline{Z}_t) \tag{4.5}$$

Although closed-form solutions of dynamic systems are generally intractable for many cases, Monte Carlo methods offer a way to approximate the inference and to characterize the evolution of the dynamic systems.

In statistics, sampling techniques are widely used to approximate a complex probability density. Sequential Monte Carlo methods for dynamic systems are also studied in the area of statistics [107, 108, 109]. A set of weighted random samples $\{(s^{(n)}, \pi^{(n)})\}, n = 1, \ldots, N$ is *properly weighted* with respect to the distribution $f(\mathbf{X})$ if for any integrable function $h(\cdot)$,

$$\lim_{N\to\infty} \frac{\sum_{n=1}^{N} h(s^{(n)})\pi^{(n)}}{\sum_{n=1}^{N} \pi^{(n)}} = E_f(h(\mathbf{X})) \tag{4.6}$$

In this sense, the distribution $f(\mathbf{X})$ is approximated by a set of discrete random samples $s^{(n)}$, each having a probability proportional to its weight $\pi^{(n)}$. Since the *a posteriori* density $p(\mathbf{X}_t|\underline{Z}_t)$ is represented by a set of weighted random samples $\{(s_t^{(n)}, \pi_t^{(n)})\}$, such a sample set will evolve into a new sample set $\{(s_{t+1}^{(n)}, \pi_{t+1}^{(n)})\}$ representing the posterior $p(\mathbf{X}_{t+1}|\underline{Z}_{t+1})$ at time $t+1$. In this sense, tracking could be characterized by the evolution of such a set of weighted samples in the state space.

4.4.1 Factored sampling

To represent the *a posteriori* density $p(\mathbf{X}_t|\underline{Z}_t)$, a set of random samples $\{\mathbf{X}_t^{(n)}, n = 1, \ldots, N\}$ could be drawn from a prior $p(\mathbf{X}_t|\underline{Z}_{t-1})$, and weighted by their measurements, i.e., $\pi_t^{(n)} = p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{X}_t^{(n)})$, such that the *a posteriori* density $p(\mathbf{X}_t|\underline{Z}_t)$ is represented by a set of weighted random samples $\{s_t^{(n)}, \pi_t^{(n)}\}$. This sampling scheme is called *factored sampling* in statistics. It could be shown that such a sample set is properly weighted. This sample set will evolve to a new sample set at time $t+1$ and the new sample set $\{s_{t+1}^{(n)}, \pi_{t+1}^{(n)}\}$ represents the *a posteriori* density $p(\mathbf{X}_{t+1}|\underline{Z}_{t+1})$ at time $t+1$. This is the sequential Monte Carlo method employed in the CONDENSATION algorithm [35, 91, 36].

CONDENSATION achieved quite robust tracking results. The robustness of Monte Carlo tracking is due to the maintaining of a pool of hypotheses. Since each hypothesis needs to be measured and associated with a likelihood value, the computational cost mainly comes from the image measurement processes. Generally, the more the samples, the better the chance to obtain accurate tracking results, but the slower the tracking speed. Consequently, the number of samples becomes an important factor in Monte Carlo based tracking, since it determines

the tracking accuracy and speed. Unfortunately, when the dimensionality of the state space increases, the number of samples increases exponentially.

This phenomenon has been observed, and different methods have been taken to reduce the number of samples. A semiparametric approach was taken in [110], which retained only the modes (or peaks) of the probability density and represented the local neighborhood surrounding each mode as a Gaussian distribution. This approach eliminated the need for a large number of samples for representing the distribution around each mode nonparametrically. Different sampling techniques were also investigated to reduce the number of samples. In [111], a partitioned sampling scheme was proposed to track articulated objects. It was basically a hierarchical method to generate the hypotheses. A similar approach was taken in [112] to track multiple objects. In [113], an annealed particle filtering scheme was taken to search the global maximum of the *a posteriori* probability density. In [114], an exclusion scheme was proposed to approach the occlusion problem in multiple target tracking.

4.4.2 Importance sampling

In practice, it might be difficult to draw random samples from the distribution $f(\mathbf{X})$. Samples could be drawn from another distribution $g(\mathbf{X})$, but their weights should be properly adjusted. This is the basic idea of the technique *importance sampling*. When samples $s^{(n)}$ are drawn from $g(\mathbf{X})$, but weights are compensated as

$$\pi^{(n)} = \frac{f(s^{(n)})}{g(s^{(n)})}\tilde{\pi}^{(n)}$$

then it can be proved that the sample set $\{s^{(n)}, \pi^{(n)}\}$ is still *properly weighted* with respect to $f(\mathbf{X})$. This is illustrated in Figure 4.3.

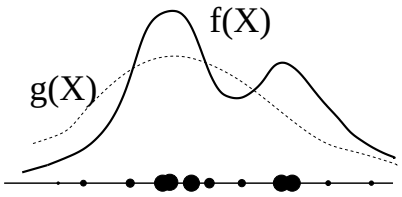


Figure 4.3 Importance sampling. Samples that are drawn from another distribution $g(\mathbf{X})$ but with adjusted weights could still be used to represent density $f(\mathbf{X})$.

To employ the importance sampling technique in dynamic systems, we need to let $f_t(\mathbf{X}_t^{(n)}) = p(\mathbf{X}_t = \mathbf{X}_t^{(n)}|\underline{Z}_{t-1})$, where $f_t(\cdot)$ is the tracking prior, i.e., a prediction density. So, when we want

to approximate the posterior $p(\mathbf{X}_t|\underline{Z})$, we could draw random samples from another distribution $g_t(\mathbf{X}_t)$, instead of the prior density $f_t(\mathbf{X}_t)$. But the sample weights should be compensated as

$$\pi_t^{(n)} = \frac{f(\mathbf{X}_t^{(n)})}{g(\mathbf{X}_t^{(n)})} p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{X}_t^{(n)}) \tag{4.7}$$

To evaluate $f_t(\mathbf{X}_t)$, we have

$$\begin{aligned} f_t(\mathbf{X}_t^{(n)}) &= p(\mathbf{X}_t = \mathbf{X}_t^{(n)}|\underline{Z}_{t-1}) \\ &= \sum_{k=1}^{N} \pi_{t-1}^{(k)} p(\mathbf{X}_t = \mathbf{X}_t^{(n)}|\mathbf{X}_{t-1} = \mathbf{X}_{t-1}^{(k)}) \end{aligned}$$

To approximate a posterior $p(\mathbf{X}_t|\underline{Z}_t)$, instead of sampling directly from the prior $p(\mathbf{X}_t|\underline{Z}_{t-1})$, samples $s^{(n)}$ could be drawn from another source $g_t(\mathbf{X}_t)$, and the weight of each sample is

$$\pi_t^{(n)} = \frac{f_t(s_t^{(n)})}{g_t(s_t^{(n)})} p(\mathbf{Z}_t|\mathbf{X}_t = s_t^{(n)}) \tag{4.8}$$

where $f_t(s_t^{(n)}) = p(\mathbf{X}_t = s_t^{(n)}|\underline{Z}_{t-1})$. We should notice here that in order to sample from $g_t(\mathbf{X}_t)$ instead of $f_t(\mathbf{X}_t)$, both $f_t(s_t^{(n)})$ and $g_t(s_t^{(n)})$ should be evaluatable. The *importance sampling* technique is an important part in the proposed *co-inference tracking* in Section 4.5. A dynamic system could be formulated in a probabilistic framework, and sampling techniques could be used to approximate probabilistic inferences.

4.5 The Co-inference Tracking Algorithm

The structured variational analysis of the factorized graphical model in Section 4.3 suggests a way to uncouple the dynamics of the states. In this section, we present an efficient algorithm to approximate the co-inference of the variational analysis based on statistical sampling and a sequential Monte Carlo technique.

Let $s_t^{(n)} = (s_t^{s,(n)}, s_t^{c,(n)})$ denote the nth sample of the target's state at time t, where $s_t^{s,(n)}$ and $s_t^{c,(n)}$ represent the shape and color states of a sample, respectively. Let $\pi_t^{s,(n)}$, $\pi_t^{c,(n)}$, and $\pi_t^{(n)}$ denote the sample weight based on shape observation, color observation, and a combination of shape and color observation, respectively. At time t, we have a set of samples associated with weights $\{(s_t^{s,(n)}, s_t^{c,(n)}, \pi_t^{s,(n)}, \pi_t^{c,(n)}, \pi_t^{(n)}), n = 1, \ldots, N\}$. To generate the samples for time $t+1$, i.e., $\{(s_{t+1}^{s,(n)}, s_{t+1}^{c,(n)}, \pi_{t+1}^{s,(n)}, \pi_{t+1}^{c,(n)}, \pi_{t+1}^{(n)}), n = 1, \ldots, N\}$, an iterative procedure is shown in Figure 4.4.

Generate $\{(s_{t+1}^{s,(n)}, s_{t+1}^{c,(n)}, \pi_{t+1}^{s,(n)}, \pi_{t+1}^{c,(n)}, \pi_{t+1}^{(n)})\}$ from $\{(s_t^{s,(n)}, s_t^{c,(n)}, \pi_t^{s,(n)}, \pi_t^{c,(n)}, \pi_t^{(n)})\}$,$n = 1, \ldots, N$:

//Step(0): Initialization
$s_{(0)}^{(\cdot)} = s_t^{(\cdot)}$; $\quad \pi_{(0)}^{*,(\cdot)} = \pi_t^{*,(\cdot)}$;

`for` $k = 0 : K-1$
  //Step(1): Shape samples generating
  $s_{(k+1)}^{s,(\cdot)} =$ `I_Samping(`$\{(s_{(k)}^{s,(\cdot)}, \pi_{(k)}^{c,(\cdot)})\}$`);`

  //Step(2): Shape observation
  $\pi_{(k+1)}^{s,(\cdot)} =$`Shape_Obsrv(`$s_{(k+1)}^{s,(\cdot)}$`);`

  //Step(3): Color samples generating
  $s_{(k+1)}^{c,(\cdot)} =$`I_Sampling(`$\{(s_{(k)}^{c,(\cdot)}, \pi_{(k+1)}^{s,(\cdot)})\}$`);`

  //Step(4): Color observation
  $\pi_{(k+1)}^{c,(\cdot)} =$`Color_Obsrv(`$s_{(k+1)}^{c,(\cdot)}$`);`
`end`

$s_{t+1}^{(\cdot)} = s_{(K)}^{(\cdot)}$; $\quad \pi_{t+1}^{*,(\cdot)} = \pi_{(K)}^{*,(\cdot)}$; $\quad \pi_{t+1}^{(\cdot)} = \pi_{t+1}^{s,(\cdot)} \pi_{t+1}^{c,(\cdot)}$;

Figure 4.4 Co-inference tracking algorithm I: *top-down*.

The basic idea behind the above iteration is that one modality receives priors from other modalities such that the *co-training* among all the modalities will tend to maximize the likelihood. Specifically, at first shape samples are drawn according to color measurements based on importance sampling, i.e., shape samples are drawn from $g_s \sim \{(s_t^{s,(n)}, \pi_t^{c,(n)})\}$ instead of $f_s \sim \{(s_t^{s,(n)}, \pi_t^{s,(n)})\}$. Since the clutter could also incur high shape measurements, sampling only from shape measurements make it difficult to handle cluttered backgrounds, especially when a generic shape representation is taken. However, sampling according to color measurements would largely ease this difficulty, since the samples with higher color measurements would have higher probability to propagate. Weight corrections are

$$\begin{aligned} \pi_t^{s,(n)} &= \frac{f_s(s_t^{s,(n)})}{g_s(s_t^{s,(n)})} p(\mathbf{Z}_t | \mathbf{X}_t = s_t^{(n)}) \\ f_s(s_t^{s,(n)}) &= \sum_{k=1}^{N} \pi_{t-1}^{s,(k)} p(\mathbf{X}_t^s = s_t^{s,(n)} | \mathbf{X}_{t-1}^s = s_{t-1}^{s,(k)}) \end{aligned}$$

Symmetrically, color samples are then drawn according to shape measurements based on importance sampling, i.e., color samples are drawn from $g_c \sim \{(s_t^{c,(n)}, \pi_t^{s,(n)})\}$ instead of $f_c \sim \{(s_t^{c,(n)}, \pi_t^{c,(n)})\}$. This step would let color samples with higher shape measurements have a better chance to propagate to the next time step.

$$\begin{aligned} \pi_t^{c,(n)} &= \frac{f_c(s_t^{c,(n)})}{g_c(s_t^{c,(n)})} p(\mathbf{Z}_t | \mathbf{X}_t = s_t^{(n)}) \\ f_c(s_t^{c,(n)}) &= \sum_{k=1}^{N} \pi_{t-1}^{c,(k)} p(\mathbf{X}_t^c = s_t^{c,(n)} | \mathbf{X}_{t-1}^c = s_{t-1}^{c,(k)}) \end{aligned}$$

The above two steps could approximate the *co-inference*. The iteration would increase the likelihood of observations. For simplicity, we let $\pi_t^{(n)} = \pi_t^{s,(n)} \pi_t^{c,(n)}$, and the estimates of the shape and color states are given by

$$\bar{\mathbf{X}}_t^s = \sum_{n=1}^{N} s_t^{s,(n)} \pi_t^{(n)}; \tag{4.9}$$

$$\bar{\mathbf{X}}_t^c = \sum_{n=1}^{N} s_t^{c,(n)} \pi_t^{(n)} \tag{4.10}$$

Our approach is different from the ICondensation algorithm in [74]. Their method assumes a fixed color distribution and color is used as an extra prior, while our approach could track both shape and color due to the *co-inference* between them. If color dynamics were fixed, our approach would be similar to their method.

The above algorithm takes the *top-down* approach for both shape and color by generating samples in the joint shape and color state space. However, we notice that it would be more efficient to combine the *top-down* and *bottom-up* approaches, since color state could be estimated by taking a bottom-up approach. The basic idea is that we generate shape samples but train a color model of the target based the color data collected according to shape samples in an EM framework. The EM iteration would end up with a color model that maximizes the likelihood of color observation.

At time t, we have $\{(s_t^{s,(n)}, \pi_t^{s,(n)}, \pi_t^{c,(n)}, \pi_t^{(n)})\}$ and a color model M_t. The procedure for generating the samples at time $t+1$ is shown in Figure 4.5.

The E-step calculates the observation probability for color model hypotheses with respect to the current color model $\tilde{M}_{(k)}$ at different positions and with different shapes. The M-step trains a new color model $\tilde{M}_{(k+1)}$ based on such observations. The EM iteration in the algorithm

Generate $\{(s_{t+1}^{s,(n)}, \pi_{t+1}^{s,(n)}, \pi_{t+1}^{c,(n)}, \pi_{t+1}^{(n)}), M_{t+1}\}$ from $\{(s_t^{s,(n)}, \pi_t^{s,(n)}, \pi_t^{c,(n)}, \pi_t^{(n)}), M_t\}$, $n = 1, \ldots, N$:

//Step(1): Shape samples generating
$s_{t+1}^{s,(\cdot)} =$ `I_Sampling(`$\{(s_t^{s,(\cdot)}, \pi_t^{c,(\cdot)})\}$`)`;

//Step(2): Shape observation
$\pi_{(t+1)}^{s,(\cdot)} =$`Shape_Obsrv(`$s_{t+1}^{s,(\cdot)}$`)`;

//Step(3): Collecting of initial color observations
$Z_{t+1}^{(\cdot)} =$`Color_Collect(`$s_{t+1}^{s,(\cdot)}$`)`;
$\tilde{M}_{(0)} = M_t$;

//Step(4): Re-training of color model
`for` $k = 0 : K-1$
    // E-step
    $\pi_{(k)}^{c,(\cdot)} =$`E(`$Z_{t+1}^{(\cdot)}, \tilde{M}_{(k)}$`)`;

    // M-step
    $\tilde{M}_{(k+1)} =$`M(`$Z_{t+1}^{(\cdot)}, \pi_{(k)}^{(\cdot)}$`)`;
`end`

$M_{t+1} = \tilde{M}_{(K)}$;

Figure 4.5 Co-inference tracking algorithm II: combining *top-down* and *bottom-up*.

basically is a bottom-up routine to learn a new color model based on the old one and a set of training data obtained from the shape model. It is similar to the transductive learning approach for color tracking in [72].

4.6 Implementation

Section 4.5 proposed a framework for tracking and integrating multiple cues based on the importance sampling technique. The remainder of this paper presents a specific implementation of a real-time tracker.

4.6.1 Shape representation

Instead of using a detailed shape model by B-splines, we employ the conics model for a general purpose, since it is more flexible. The conics model is suitable for certain specific

applications, such as tracking human heads or fingertips. We take a generic form of the conics, i.e.,

$$\mathbf{X}'A\mathbf{X}' + 2B\mathbf{X} + C = 0$$

A shape template is initialized by conics fitting. The deformation of the shape is governed by an affine transformation:

$$\mathbf{Y} = A\mathbf{X} + \mathbf{t} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

which characterizes the shape space $\mathcal{S}$. Thus, the dimensionality of the shape space $\mathcal{S}$ is 6. Taking the idea of shape space [35], we could determine a conic shape given the template and an affine transformation. The shape samples in our algorithms are drawn in the shape space, i.e., $\mathbf{X}^s = (A_{11}, A_{12}, A_{21}, A_{22}, t_1, t_2)^T$.

4.6.2 Shape observation

It is crucial to have an accurate shape observation in tracking. Our implementation takes a similar approach to that used in [35]. Edge detection is performed in 1-D along the normal lines of the hypothesized shapes, shown in Figure 4.6. Thus, observation reduces to a set of scalar positions $\mathbf{z} = (z_1, \ldots, z_M)$ due to the presence of clutter. The true observation $\tilde{z}$ could be any one of them. So,

$$p(\mathbf{z}|x) = qp(\mathbf{z}|\text{clutter}) + \sum_{m=1}^{M} p(\mathbf{z}|x, \tilde{z} = z_m)P(\tilde{z} = z_m)$$

where x is the point on the shape contour and q is the probability that none of such M positions could be viewed as an observation.

$$q = 1 - \sum_{m=1}^{M} P(\tilde{z} = z_m)$$

When we assume that any true observation is unbiased and normally distributed with standard deviation σ, $P(\tilde{z} = z_m) = p$ for all z_m, and the clutter is a Poisson process with density λ, then,

$$p(\mathbf{z}|x) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma q\lambda} \sum_{m} \exp -\frac{(z_m - x)^2}{2\sigma^2} \tag{4.11}$$

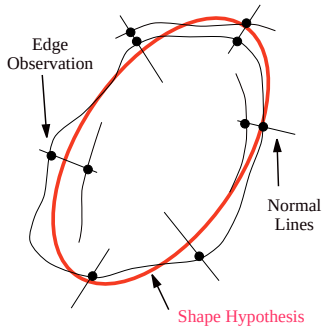


Figure 4.6 Shape observation and measurement. A set of 1-D observations are applied along the contour of a shape hypothesis. Each 1-D measurement calculates the likelihood of a segment of shape contour by observing some edges.

To measure a shape hypothesis, we can apply a set of N 1-D observations $p(\mathbf{z}_n^s|x_n^s)$, where $n \in \{1, \ldots, N\}$ along the shape contour. Such 1-D measurements are obtained on a set of normal lines of the contour. In our experiments, we use an ellipse as the target shape representation because it is a general shape for many convex objects such as the human head. An example is shown in Figure 4.6, in which the red line is a shape hypothesis, the curves are the edges detected, and the line segments are the normal lines of the ellipse. When assuming all individual 1-D observations are independent of each other, we obtain the measurement of a shape hypothesis:

$$p(\mathbf{Z}^s|\mathbf{X}^s) \propto \prod_{n=1}^{N} p(\mathbf{z}_n^s|x_n^s) \tag{4.12}$$

4.6.3 Color representation

We take a parametric color representation in normalized-RGB color space. If the object is uniform in color, a Gaussian distribution is taken to model the color distribution. For simplicity, we represent the color state by $\mathbf{X}^c = (\mu_{\tilde{r}}, \mu_{\tilde{g}}, \mu_{\tilde{b}}, \sigma_{\tilde{r}}, \sigma_{\tilde{g}}, \sigma_{\tilde{b}})$. If the target has two salient colors, a mixture of two Gaussians could model such a distribution. To keep the dimensionality small, we represent the color state by $\mathbf{X}^c = (\mu_{\tilde{r}}^1, \mu_{\tilde{g}}^1, \mu_{\tilde{b}}^1, \mu_{\tilde{r}}^2, \mu_{\tilde{g}}^2, \mu_{\tilde{b}}^2)$.

We also take a nonparametric representation by 2-D color histogram, which uses two normalized colors such as $\tilde{r}$ and $\tilde{g}$ with N bins. We set $N = 3$ for our approach I and $N = 8$ for approach II.

4.6.4 Color observation

A set of color pixels is collected inside the shape contour. If the parametric approach is taken, a parametric color model will be estimated based on these color pixels, and the Mahalanobis distance is used to measure the similarity of the two distributions.

If the nonparametric approach is taken, a color histogram will be built based on these color pixels, and the histogram intersection $\phi_c(s)$ [94, 115] is computed between the hypothesis color model $\mathbf{X}^c$ and the observed histogram $\mathbf{Z}^c = I_s$:

$$\phi_c(s) = \frac{\sum_{k=1}^{N} \min(I_s(k), \mathbf{X}^c(k))}{\sum_{k=1}^{N} I_s(k)} \tag{4.13}$$

We can use such a histogram intersection to approximate the color likelihood:

$$p(\mathbf{Z}^c|\mathbf{X}^c) \sim \phi_c(s)$$

4.7 Experiments

This section reports some experimental results of sequential Monte Carlo tracking techniques and our tracking algorithm based on co-inference learning. The tracking performances of both signal cue and multiple cues are examined in this section.

4.7.1 Single cue

We implement the CONDENSATION algorithm, and run it with two different methods of hypotheses measurements, i.e., shape and color, respectively. The CONDENSATION maintains a set of particles (or hypotheses) that represent the posterior probability of the target's state $p(\mathbf{X}_t|\underline{Z}_t)$. Here, we use shape representation. Generally, the expectation $E[\mathbf{X}_t|\underline{Z}_t]$ could be calculated based on factored Monte Carlo sampling to approximate the target estimation.

When the shape hypotheses are solely measured against edge observation, the tracking algorithm works well in simple backgrounds and where strong edges can be observed. However, when the background is cluttered, the tracking often fails because some hypotheses with high probability might be distractors in terms of shape. In Figure 4.7, we use red ellipses to represent shape hypotheses. The brighter the red ellipse, the higher the likelihood of such a hypothesis. The blue ellipse is the shape estimation of the target. In these examples, we observed that

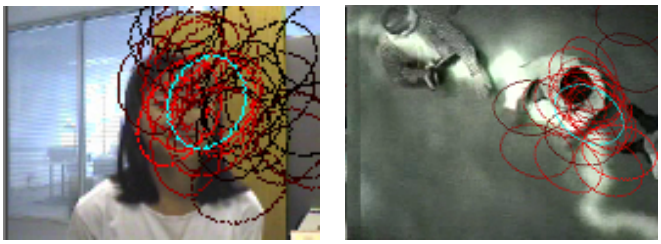
many hypotheses have high probability on the cluttered backgrounds, such as the keyboard area in Figure 4.7(a) and the bookshelf area in Figure 4.7(b).

(a) (b)

Figure 4.7 CONDENSATION using shape observation alone: Many hypotheses were generated on clutter. (a) Tracking hand. The keyboard area produces many false shape hypotheses. (b) Tracking head. The bookshelf area is highly cluttered so that many hypotheses have incorrect shape measurements.

Part of the reason for the tracking failure is that we use a very generic shape model, which does not accurately align to the contours of the target. The advantage of using a generic model is that it will allow more shape deformation. On the other hand, if a specific shape model is employed such as the B-spline model in [35], it will largely alleviate the difficulty of clutter because of the uniqueness of the shape representation. This is the direct reason that a leaf could be tracked against a background full of leafs in [35]. However, it would be difficult to handle the large shape deformation caused by out-of-plane rotation when using affine shape space.

In another experiment we made color observations for the shape hypotheses. We constructed a color model of the target in the initialization step of tracking, for example, the histogram of the target. For each shape hypothesis, we measured the likelihood that such color model produces the color distribution inside the shape hypothesis. When the hypotheses are solely measured against their color distributions, the tracking algorithm succeeds when the background does not have a comparable area with similar color as the target. However, the tracking often fails when the background has similar colors to the targets. Figure 4.8(a) shows the case when the wooden color is similar to skin tone such that false hypotheses are generated. In Figure 4.8(b), the lighting conditions change dramatically, which makes it difficult to track the shoulder of the person.



(a) (b)

Figure 4.8 CONDENSATION using color observation alone: Color distracters and nonstationary environments make tracking difficult. (a) Tracking face. Many hypotheses are generated for the wooden door area because many unlikely hypotheses survive the color measurements. (b) Tracking shoulder in a dynamic environment. Due to the changing of the illumination conditions, the color measurements will not be accurate anymore.

4.7.2 Multiple cues

Our tracking algorithm has been applied to a variety of environments and tracking tasks. Our experiments show that the tracking algorithm with multiple cues performs very robustly. The tracking algorithm runs on a 1-processor PIII 850 MHz PC at around 30 Hz.[1]

Our co-inference tracking algorithm has been successfully applied to different tracking scenarios. Figure 4.9 shows the example of tracking a hand fist with large rotations against a cluttered background. Figure 4.10 shows the example of tracking a head with out-of-plane rotations in an office environment.[2] In Figure 4.11, the task is to track the speaker's head in a lecture room where the lighting changes dramatically. Figure 4.12 shows an example of tracking a person's shoulder from the top view in a virtual environment. We also experimented with some occlusions, which are shown in Figure 4.13.

In Figure 4.9, a hand is moving and rotating in a cluttered background. If tracking is solely based on shape and edge, it will be lost when the hand leaves the keyboard area, which has been shown in the previous section. However, our algorithm, which employs both shape and color representation, and make both shape and color observations, can overcome this difficulty. The reason behind the success is that different modalities provide reinforcement to each other in a co-inference fashion. Including both shape and color results in a richer target representation such that the tracking algorithm can work in a more complex environment. Although the

[1]Some of the demo sequences can be obtained from `http://www.ifp.uiuc.edu/~yingwu`.

[2]The testing sequences were obtained from `http://robotics.stanford.edu/~birch/headtracker`.

Figure 4.9 A hand in clutter.

dimensionality of its state space is higher than in the case of single modality, the co-inference among different modalities can explore the high dimensional state space efficiently.

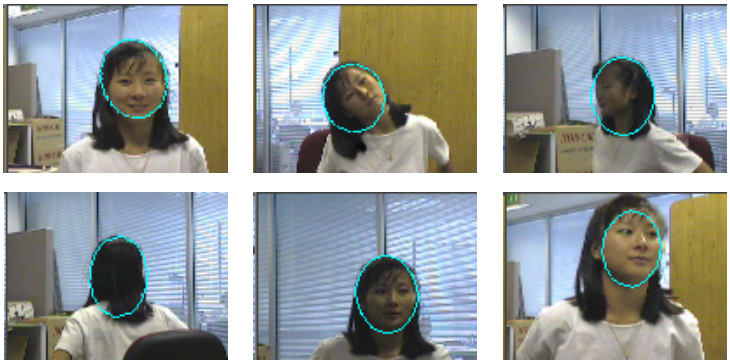


Figure 4.10 A moving face in an office environment. *Testing sequence courtesy of Dr. Stan Birchfield.*

Figure 4.10 shows the result of our algorithm to track a head in an office. Obviously, the color distribution of the girl's head has at least two components: skin color tone and black hair tone. So, when she turns her head around, it makes nonstationary color changes of the visible side of the head. In this scenario, it does not make sense to construct a fixed color model for her head, since the color distribution of the visible side of her head varies when she rotates her head. One of the solutions is to make a 3-D head model with color features [94]. A similar texture model was reported in [77]. Another approach is to adapt the color model to the lighting changes [32, 72]. Our co-inference tracking algorithm adapts the nonstationary color distribution, with the assumption that the changing of color distribution is slow. Generally, this assumption is valid for many scenarios. Our algorithm tracks the head very accurately, even

when she moves in front of the wooden door. The reason for this is that the shape modality provides an external constraint for color modality.

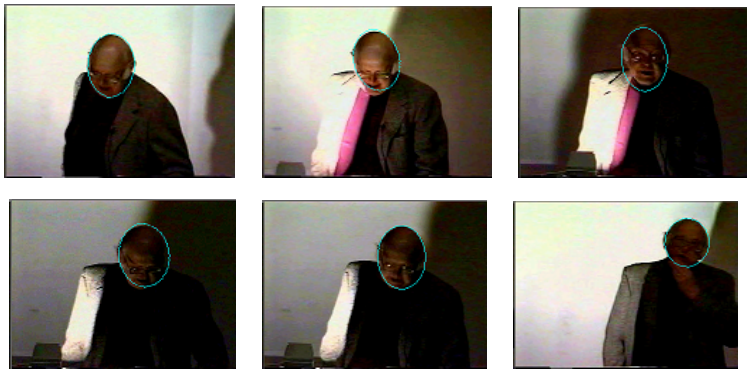


Figure 4.11 Head in a lecture room with dramatic lighting variations. *Testing sequence courtesy of Dr. Kentaro Toyama.*

Figure 4.11 shows the case of a lecture room where the lighting changes dramatically due to an overhead projector. The color of the speaker's head varies in a wide range of intensities. Our algorithm tracks the speaker's head pretty robustly, although it will fail reasonably due to large movements of the camera and speaker's uncertain movements in very dim light.

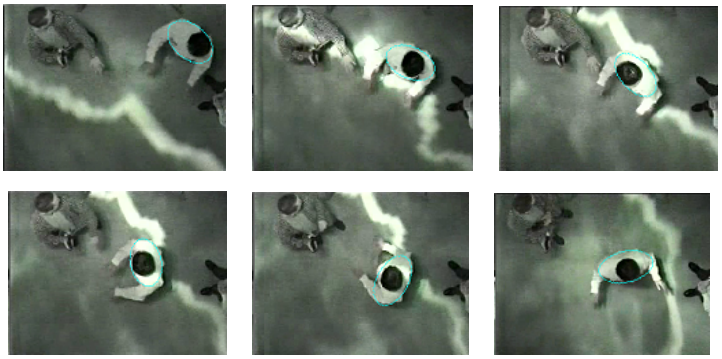


Figure 4.12 Shoulder in CAVE, a large virtual environment with lighting diffused from screen displays.

Figure 4.12 shows the tracking scenario in a large virtual environment, which has four displays on three sides and floor. The camera is mounted on the ceiling. It is of interest to estimate the user's position and orientation by tracking his head and shoulder. The difficulty is that the displays will diffuse a large amount of lighting in the environments. Tracking the shoulder is even harder than tracking head, since the shoulder deforms and rotates much more,

and in addition it does not produce strong edges as the head does. It is a very difficult scenario for the methods using single modality. However, employing multiple modalities in the target representation, our algorithm works robustly when parameters are properly set.

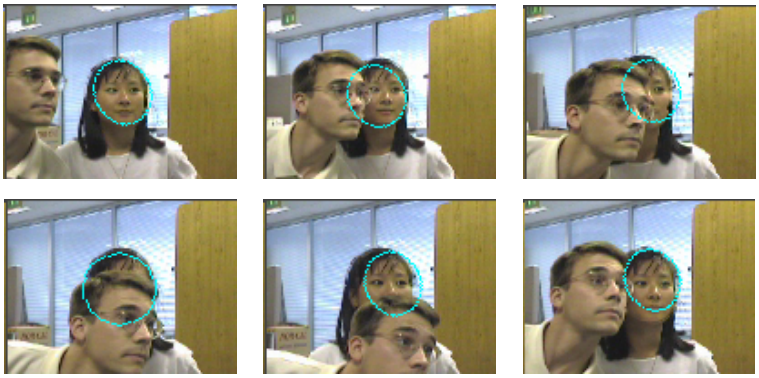


Figure 4.13 A face occluded by another moving face in an office environment. *Testing sequence courtesy of Dr. Stan Birchfield.*

Figure 4.13 shows a good example of our algorithm for handling occlusion. In this example, a boy is moving in front of the target, i.e., the girl, which causes the occlusion of the girl's face in a period of time. The co-inference tracking algorithm keeps tracking the girl when occlusion occurs. The reason for this example is that the occluding object (the boy) has a different size from the target (the girl), which avoids generating too many hypotheses on the occluding object.

From our extensive experiments on both live video and recorded sequences, we found that our co-inference tracking algorithm performed very robustly against clutter backgrounds, non-stationary color changes, slow dynamic illumination environments and some occlusion scenarios.

4.8 Discussion

Visual tracking is a fundamental problem in computer vision. It receives more and more attention due to the rapid development of visual surveillance and vision-based human-computer interaction, in which robust tracking methods are desirable. However, one of the difficulties confronting us is how to measure a tracking hypothesis from image observations. The richer the target's representation we use, the more accurate we can measure the likelihood of a hypothesis. Basically, the tracking process is to search for the best match in the target's state space.

Thus, when we use multiple modalities for target representation, many tracking methods, such as the CONDENSATION algorithm, will involve exponential increases in computation with the dimensionality of the state space.

In this paper we have presented a co-inference approach for integrating and tracking multiple cues. The tracking problem could be formulated as the inference problem of a graphical model. This approach is based on the structured variational analysis of a factorized graphical model, which suggests that the inference in a higher dimensional state space can be factorized by several lower dimensional state spaces in an iterative fashion. We call this co-inferencing. A sequential Monte Carlo tracking algorithm, based on the importance sampling technique, is proposed to approximate the co-inference process among different modalities. Our tracking algorithm is robust in dealing with target deformation and color variations, since a richer representation of the target is taken.

The co-inferencing phenomenon is very interesting since it involves the information fusion and exchanges between different modalities. We can see that it might be a general phenomenon for the learning in high dimensional space. In the area of text classification, although it does not involve dynamics, a similar approach, called *co-training* [116], has been taken to explore the correlation among different modalities. We will investigate the occlusion problem further and extend our work to the case of tracking multiple objects and articulated objects in the future.

CHAPTER 5

CAPTURING HUMAN HAND MOTION

5.1 Introduction

Hand gestures could be a more natural and articulate way for many human-computer interaction applications. For example, people could use their hands to manipulate virtual objects directly in virtual environments. But one of the difficulties confronting us is how to capture human hand motion. As an alternative to glove-based techniques that require users to wear a special glove, vision-based techniques are noninvasive and more affordable. However, capturing hand and finger motion in video streams is a quite challenging task. Typical hand motions include global translation, rotation, and natural finger movements. Hand rotation would incur self-occlusion in that some fingers may become invisible. Since the finger motion has high degrees of freedom, many techniques of estimating finger articulations often involve a formidable search problem in a high-dimensional space.

Different methods have been proposed to analyze human hand motion. One approach makes use of deformable hand shape models [19], in which the hand shape deformation could be governed by Newtonian dynamics or statistical training methods such as the principal component analysis (PCA) technique. However, accurate estimates of hand poses are hard to obtain by these methods.

Another approach is the appearance-based approach, which tries to establish the mapping between the image feature space and the hand motion space [42, 117]. However, the appearance-based approach generally involves a quite difficult learning problem, and it is not trivial to collect large sets of training data.

Many other methods could be categorized into the 3-D model-based approach. Hand motion could be recovered by matching the projected 3-D model and observed image features, so that the problem becomes a search problem in a high-dimensional space. To construct the correspondences between the model and the images, different image observations have been studied, such as fingertips [18, 23, 24], line features [22, 118], and silhouettes [17, 37, 113].

Many methods tackle the global hand motion and local finger motion simultaneously, such that the optimization would have a very high chance to get into local minima. On the other hand, a divide-and-conquer approach [23] could be taken to separate the hand pose determination and articulation estimation. It could be a general method for articulate objects.

This paper proposes an approach to capture both hand pose and finger articulation in the divide-and-conquer framework. We propose an algorithm combining the iterative closed points (ICP) algorithm and the factorization method to determine global hand pose. Section 5.3 will describe this approach. The fact that the natural finger motion is also highly constrained helps us reduce the dimensionality of the feasible hand configuration space. We propose to use a set of linear manifolds to characterize hand configuration. Details of the hand motion model can be found in Section 5.2. An efficient algorithm based on sequential Monte Carlo techniques for finger motion is also presented in this paper, which will be presented in Section 5.4. To enhance the accuracy, the iteration between the pose estimation and finger articulation tracking is performed in an EM fashion. Details will be given in Section 5.5. Our experiments including simulation and real sequences will be shown in Section 5.6.

5.2 Hand Motion and Hand Model

Hand motion consists of global hand pose M_G and local finger articulation M_L. Global pose M_G is the 3-D translation $\mathbf{t}$ and rotation $\mathbf{R}$ of the palm, and finger motion M_L could be represented by the set of joint angles Θ. We can see from our finger and hand model in Figure 5.1(a) that $\Theta \in \mathcal{R}^{20}$. The names of the joints are also shown in Figure 5.1(a). Thus the task of motion capturing is to estimate $\{\mathbf{R}, \mathbf{t}, \Theta\}$ from image sequences.

The hand could be modeled by a *cardboard* model, in which each finger could be represented by a set of three connected planar patches. The length and width of each patch should be

calibrated to individual people. The cardboard model is shown in Figure 5.1(b). Although it is a simplification of the real hand, it offers a good approximation for motion capturing.

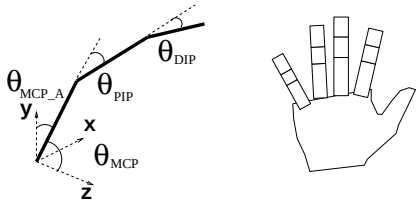
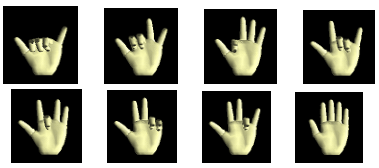


Figure 5.1 Hand Model: (a) kinematical chain of one finger, (b) cardboard hand model.

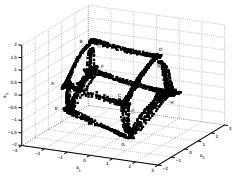
Integrating finger motion constraints and reducing the dimensionalities, we employ hand configuration space $\mathbf{X} \in \Xi$ to represent local finger motion, instead of using the joint angle $\Theta \in \mathcal{R}^{20}$. We are particularly interested in the dimensionality and topology of Ξ. Using CyberGlove, we have collected a set of more than 30,000 joint angle data, which could well represent the finger motion constraints. The PCA technique is employed to eliminate the redundancy. We can project $\Theta \in \mathcal{R}^{20}$ into a 7-dimensional subspace while maintain 95% of the information. Thus, $\mathbf{X} \in \Xi \subset \mathcal{R}^7$. Furthermore, we define 28 basis configurations $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_M : \forall \mathbf{b}_k \in \Xi, M = 28\}$ to characterize the configuration space Ξ. These basis configurations could be identified by clustering the data in Ξ or selecting intuitively. Some of them are shown in Figure 5.2(a). Surprisingly, after examining the data in Ξ, we found that natural hand articulation lies largely in the linear manifolds spanned by any two basis configurations. For example, if the hand moves from a basis configuration $\mathbf{b}_i$ to another basis $\mathbf{b}_j$, the intermediate hand configuration lies approximately on the linear manifold spanned by $\mathbf{b}_i$ and $\mathbf{b}_j$, i.e., $\mathbf{X} \in \mathcal{L}_{ij} = s\mathbf{b}_i + (1-s)\mathbf{b}_j$. Consequently, hand articulation could be characterized in the configuration space by

$$\Xi \approx \bigcup_{i,j} \mathcal{L}_{ij}, \text{ where } \mathcal{L}_{ij} = span(\mathbf{b}_i, \mathbf{b}_j) \tag{5.1}$$

A lower-dimensional illustration is shown in Figure 5.2(b), in which the black dots are real finger motion data points.



(a)



(b)

Figure 5.2 Hand articulation in the configuration space, which is characterized by a set of basis configurations and linear manifolds. (a) A subset of the basis configurations, (b)linear manifolds in the configuration space.

5.3 Capturing Global Motion

The global hand motion is defined by the pose of the palm. In this paper, we treat the palm as a rigid planar object. The pose determination is formulated under scaled orthographic projection in Section 5.3.1 and the global motion is computed via the iterative closed points approach in Section 5.3.2.

5.3.1 Hand pose determination

In this section, we assume the correspondences have been constructed for pose determination. The process of building the correspondences will be presented in Section 5.3.2. Let a point on the plane be $\mathbf{x}_i = [x_i, y_i]^T$, and its image point be $\mathbf{m}_i = [u_i, v_i]^T$. Under scaled orthographic projection, we have

$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ 0 & 0 & 0 & t_3 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 0 \\ 1 \end{bmatrix}$$

That is

$$t_3 \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Or $t_3\mathbf{m}_i = \mathbf{A}\mathbf{x}_i + \mathbf{t}$, where

$$\mathbf{A} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

We can subtract the centroids of the projection points and model points, i.e., $\hat{\mathbf{m}}_i = \mathbf{m}_i - \bar{\mathbf{m}}$ and $\hat{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$, which gives $t_3\hat{\mathbf{m}}_i = \mathbf{A}\hat{\mathbf{x}}_i$. Let $\mathbf{B} = \mathbf{A}/t_3 = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$; then we write

$$\hat{\mathbf{m}}_i = \mathbf{B}\hat{\mathbf{x}}_i$$

Denoting $[u_i^k, v_i^k]^T$ the ith image point at the kth frame, we can write

$$\mathbf{W} = \begin{bmatrix} u_1^1 & u_2^1 & \dots & u_N^1 \\ v_1^1 & v_2^1 & \dots & v_N^1 \\ u_1^2 & u_2^2 & \dots & u_N^2 \\ v_1^2 & v_2^2 & \dots & v_N^2 \end{bmatrix} = \mathbf{M}\mathbf{S} \tag{5.2}$$

where

$$\mathbf{M} = \begin{bmatrix} M^1 \\ M^2 \end{bmatrix} \quad \text{and} \quad \mathbf{S} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \\ y_1 & y_2 & \dots & y_N \end{bmatrix}$$

The factorization method [119] could be taken to solve $\mathbf{M}$. Based on SVD, we can write

$$\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = (\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}})(\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^T) = \hat{\mathbf{M}}\hat{\mathbf{S}}$$

It is well known that the factorization can recover the motion $\mathbf{M}$ and shape $\mathbf{S}$ up to a matrix $\mathbf{D}$, since $\mathbf{W} = \hat{\mathbf{M}}\hat{\mathbf{S}} = (\hat{\mathbf{M}}\mathbf{D})(\mathbf{D}^{-1}\hat{\mathbf{S}}) = \mathbf{M}\mathbf{S}$. But $\mathbf{D}$ could be found by the constraints of $\mathbf{M}$. When we construct the feature points correspondences among several image frames, such structure from motion technique could allow us to calibrate a 3-D hand model automatically, which is more desirable than manually calibrating the hand model. Once the 3-D model is calibrated, i.e., $\mathbf{S}$ is computed, calculation of the motion $\mathbf{M}$ is straightforward. We let $\mathbf{S} = \mathbf{D}^{-1}\hat{\mathbf{S}}$ and solve $\mathbf{D}$ by $\mathbf{D} = \hat{\mathbf{S}}\mathbf{S}^\dagger$, where $\mathbf{S}^\dagger = \mathbf{S}^T(\mathbf{S}\mathbf{S}^T)^{-1}$. Then the motion could be solved by

$$\mathbf{M} = \hat{\mathbf{M}}\mathbf{D} = \hat{\mathbf{M}}\hat{\mathbf{S}}\mathbf{S}^\dagger$$

Since $\mathbf{M}$ contains only $\mathbf{B}$ and t_3, in Appendix B, we show how to estimate $\mathbf{R}$ and t_3 from $\mathbf{B}$. Once the rotation matrix $\mathbf{R}$ and the depth translation t_3 are computed, we can easily compute

$$\begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix} = t_3\bar{\mathbf{m}} - \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \bar{\mathbf{x}}$$

For simplicity, we could use the first frame that shows the front palm for calibration, and take the image points along the palm contour as the model points.

5.3.2 The iterative closed points algorithm

The pose determination method presented in the previous section assumes point correspondences. In this section we describe a method for establishing point correspondences by adapting the idea of the iterative closed points (ICP) algorithm. A comprehensive description of ICP for free-form curve registration can be found in [120]. The basic idea is to refine the correspondences and the motion parameters iteratively.

Since we treat the palm as a rigid planar object, it can be represent by its contour curve. A curve can be represented by a set of points. Let $\mathbf{x}_j (1 \leq j \leq N)$ be the N chained points on the 3-D curve model $\mathcal{C}$. Let $\mathcal{C}'$ be the edges observed in the image. The objective is to construct the correspondences between the two curves, such that

$$\mathcal{F}(\mathbf{R}, \mathbf{t}) = \sum_{j=1}^{N} w_j D(\mathbf{P}(\mathbf{R}\mathbf{x}_j^t + \mathbf{t}), \mathcal{C}') \tag{5.3}$$

is minimized, where $D(\mathbf{x}, \mathcal{C}')$ denotes the distance of the point $\mathbf{x}$ and the curve $\mathcal{C}'$, and w_j takes value 1 if there is a match for $\mathbf{x}_j$ and takes 0 otherwise, and $\mathbf{P}$ represents the image formation.

The ICP algorithm takes the image edge point that is closest to the projected 3-D model point, i.e., $\mathbf{P}(\mathbf{R}\mathbf{x}_k^t + \mathbf{t})$, as its correspondence. When all image edge points are far enough from the projection, this model point $\mathbf{x}_k$ cannot be matched and we set $w_k = 0$. Motion $(\mathbf{R}, \mathbf{t})$ could be computed based on such temporary correspondences using the pose determination method present in Section 5.3.1. The computed motion would result in a new matching. Iteratively applying this procedure, ICP could yield a better and better pose estimation. It should be pointed out that the ICP procedure converges only to local minima, which means that we need a close initial start. Obviously, the ICP algorithm could be easily extended to two-frame registration.

5.4 Capturing Local Motion

This section presents our method for tracking local finger motion. Since the articulated finger motion is highly constrained, we propose a sequential Monte Carlo algorithm taking advantage of these constraints.

5.4.1 Sequential Monte Carlo techniques

The tracking problem could be formulated as a process of conditional probability density propagation. Denote the target state and image observation by $\mathbf{X}_t$ and $\mathbf{Z}_t$, respectively, and $\underline{X}_t = \{\mathbf{X}_1, \ldots, \mathbf{X}_t\}, \underline{Z}_t = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_t\}$. The tracking problem is formulated as

$$p(\mathbf{X}_{t+1}|\underline{Z}_{t+1}) \propto p(\mathbf{Z}_{t+1}|\mathbf{X}_{t+1})p(\mathbf{X}_{t+1}|\underline{Z}_t) \tag{5.4}$$

Monte Carlo methods offer a way to approximate the evolution of the densities.

The *a posteriori* density $p(\mathbf{X}_t|\underline{Z}_t)$ could be represented by a set of weighted random samples $\{s_t^{(n)}, \pi_t^{(n)}\}$. This sample set will evolve to a new sample set at time $t+1$ such that the new sample set $\{s_{t+1}^{(n)}, \pi_{t+1}^{(n)}\}$ represents the *a posteriori* density $p(\mathbf{X}_{t+1}|\underline{Z}_{t+1})$ at time $t+1$. According to the source of sampling priors, sequential Monte Carlo could have different sampling schemes, including factored sampling, i.e., the CONDENSATION algorithm [91], which samples directly from the prediction prior $P(\mathbf{X}_t|\underline{Z}_{t-1})$, and importance sampling [74] which samples from an outside density. Importance sampling provides a flexible way to choose tracking priors, and the extra computation is to compensate the weights of the samples.

Since finger articulation has very high degrees of freedom, the CONDENSATION still requires a huge amount of samples for density propagation; thus an intensive computation would be involved. Fortunately, we could take advantage of the constrained finger motion model described in Section 5.2, which represents the finger configuration space by a set of linear manifolds. Although such representation is not perfectly accurate, it could still serve as a good outside prior for the importance sampling technique to reduce the computational complexity.

Let $f_t(\mathbf{X}_t^{(n)}) = p(\mathbf{X}_t = \mathbf{X}_t^{(n)}|\underline{Z}_{t-1})$, where $f_t(\cdot)$ is the prediction tracking prior. When we want to approximate the posterior $p(\mathbf{X}_t|\underline{Z})$, we could draw random samples from another distribution $g_t(\mathbf{X}_t)$, instead of the prior density $f_t(\mathbf{X}_t)$. But the sample weights should be

compensated by

$$\pi_t^{(n)} = \frac{f(\mathbf{X}_t^{(n)})}{g(\mathbf{X}_t^{(n)})} p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{X}_t^{(n)}) \tag{5.5}$$

For natural finger motion, each hand configuration $\mathbf{X}$ should be either around a basis state $\mathbf{b}_k, 1 \leq k \leq M$, or on the manifold $\mathcal{L}_{ij}$, where $i \neq j, 1 \leq i, j \leq M$. Suppose at time frame t, the hand configuration is $\mathbf{X}_t$, which is not on any basis states. The process of generating new hypotheses is shown in Figure 5.3(a). First, we find the projection $\bar{\mathbf{X}}_t$ of $\mathbf{X}_t$ onto the nearest manifold $\mathcal{L}_{ij}^*$. Accordingly,

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

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

$$s_{t+1}^{(n)} \sim p_{ij} = N(s_t, \sigma) \tag{5.6}$$

$$\widetilde{\mathbf{X}}_{t+1}^{(n)} = s_{t+1}^{(n)}\mathbf{b}_i + (1 - s_{t+1}^{(n)})\mathbf{b}_j \tag{5.7}$$

Then, perform random walk on $\widetilde{\mathbf{X}}_{t+1}^{(n)}$ to obtain hypothesis $\mathbf{X}_{t+1}^{(n)}$. So, we could write the importance function as $g_{t+1}(\mathbf{X}_{t+1}^{(n)}) = p(s_{t+1}^{(n)}|s_t)p(\mathbf{X}_{t+1}^{(n)}|\widetilde{\mathbf{X}}_{t+1}^{(n)})$. So,

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

So, the weight of each sample is

$$\pi_{t+1}^{(n)} = \frac{f_{t+1}(\mathbf{X}_{t+1}^{(n)})}{g_{t+1}(\mathbf{X}_{t+1}^{(n)})} p(\mathbf{Z}_{t+1}|\mathbf{X}_{t+1} = \mathbf{X}_{t+1}^{(n)}) \tag{5.8}$$

If the previous hand configuration is one of the basis configurations, say $\mathbf{X}_t = \mathbf{b}_k$, the process of generating hypotheses is illustrated in Figure 5.3(b). It is reasonable to assume that it takes any one of the manifolds of $\{\mathcal{L}_{kj}, 1 \leq j \leq M\}$ with equal probability. Consequently, random samples are drawn from a mixture density p_k. Details of the algorithm can be found in [37].

5.4.2 Model matching

We employ both edge and silhouette observations to measure the likelihood of articulate motion hypotheses, i.e., $p(\mathbf{Z}_t|\mathbf{X}_t)$. Self-occlusion is handled by constructing an occlusion map for

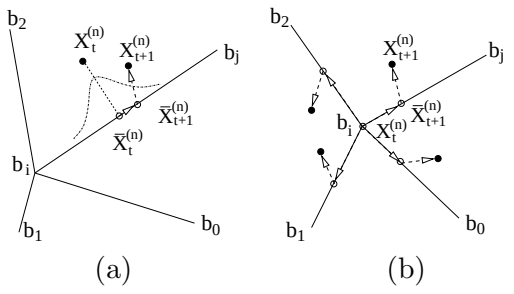


Figure 5.3 Generating hypotheses: (a) $\mathbf{X}_t^{(n)} \neq \mathbf{b}_i$, (b) $\mathbf{X}_t^{(n)} = \mathbf{b}_i$.

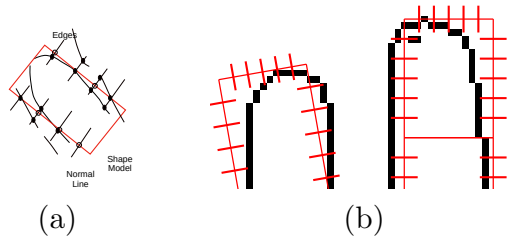


Figure 5.4 Shape measurements.

the hand model. Since the hand is represented by a cardboard model, it is expected to observe two edges for each planar patch. The observation process can be illustrated by Figure 5.4.

The cardboard model is sampled at a set of K points on the laterals of the patches. For each such sample, edge detection is performed on the points along the normal of this sample. When we assume that M edge points $\{z_m, 1 \leq m \leq M\}$ are observed, and the clutter is a Poisson process with density λ, then,

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

We also consider the silhouette measurements by calculating the difference between the areas of the image A_I and the projected cardboard model A_M, i.e., $p^a \propto \exp - \frac{(A_I - A_M)^2}{2\sigma_a^2}$. Thus, the likelihood can be written as

$$p(\mathbf{Z}|\mathbf{X}) \propto p^a \prod_{k=1}^{K} p_k^e \tag{5.9}$$

5.5 Divide and Conquer

Sections 5.3 and 5.4 treat global hand poses and local finger articulations independently. However, the method for local finger motion capturing is based on global hand pose, since the 3-D model projection depends on both hand pose and finger joint angles. Any inaccuracy in

global pose estimation will induce errors when estimating local articulation, because the method would mistakenly try to stretch and bend fingers to match the observation when there are some errors in global pose.

Unfortunately, the pose determination method in Section 5.3 would induce inaccuracies, since the method matches the palm against all the edges observed in the images. The inaccuracy occurs when the index or the pinky finger is straight, which would result in wrong scaling and rotation. We do observe such phenomena in our experiments.

Our basic idea to tackle this difficulty is to introduce more feature points for pose determination. These extra feature points could largely reduce the ambiguity. We could pick some of these points when the local finger motion is known. For example, if we know the MCP joint of the index or the pinky finger is nonzero, we could use the point at the MCP joint. If we know any of the fingers is straight, its fingertip could be used. The principle is that those points lie on the same plane as the palm (on or outside the palm certainly). Generally, these points provide bounds of the model for matching. Our extensive experiments have verified the usefulness of these extra points.

Obviously, we can only find such extra points when we know the local finger configurations. So, we iterate between two steps: (a) pose determination based on palm contour and some extra points (using the method in Section 5.3) and (b) tracking local finger configuration (using the method in Section 5.4) and finding some extra points. The iterations between global and local motion estimation would converge to a stationary point that locally minimizes the discrepancy between the observations and the model projections. Below we shall see the reason behind this.

The human hand is a special articulated object because the role of the fingers in motion is much different from that of the palm. Based on the observation that the global hand motion is represented by the position and orientation of the palm, we define the global hand motion as the exact pose of the palm. The local hand motion is largely determined by the motion of five fingers. Therefore, the problem of hand motion capturing can be specifically formulated as

$$(\mathbf{R}, \mathbf{T},\ \theta) = arg \min_{\{R,t,\ \theta\}} \mathcal{D}(\mathbf{X} - \mathbf{P}M_G(\mathbf{R}, \mathbf{T})M_L(\ \theta)\mathbf{x}^m) \tag{5.10}$$

where $M_G(\mathbf{R}, \mathbf{T})$ is the global motion, $M_L(\ \theta)$ is the local finger motion, and perspective projection is approximated by scaled orthographic projection. Then, it is possible to estimate M_G and M_L.

Given a specific local motion, we can define an operation $\mathcal{G}(\ \theta)$ to estimate the global motion such that

$$(\mathbf{R}, \mathbf{T}) = \mathcal{G}(\ \theta) = arg \ \min_{\{R,T\}} e(\mathbf{R}, \mathbf{T}, \ \theta) \tag{5.11}$$

Given a global motion, an operation $\mathcal{L}(\mathbf{R}, \mathbf{T})$ is defined to estimate the local motion such that

$$\theta = \mathcal{L}(\mathbf{R}, \mathbf{T}) = arg \ \min_{\theta} e(\mathbf{R}, \mathbf{T}, \ \theta) \tag{5.12}$$

where e is an error measurement defined as

$$e = e(\mathbf{R}, \mathbf{T}, \ \theta) = \mathcal{D}(\mathbf{X} - \mathbf{P}M_G(\mathbf{R}, \mathbf{T})M_L(\ \theta)\mathbf{x}) \tag{5.13}$$

We propose a two-step iterative algorithm to estimate the global hand motion M_G and the local motion M_L for one time frame:

- Track 2-D features $\mathbf{I}(t)$ at time t, using the previous hand model, $H(t) = H(t-1) = H(\ \theta(t-1))$. The features are detected and tracked by the feature tracking algorithm based on prediction from time $t-1$.

- The iterative algorithm is applied to estimate the global motion $\mathbf{R}(t), \mathbf{T}(t)$ and the hand state $\ \theta(t)$ at time t. Here, x^k is used to represent $x^k(t)$ for short.

  1. Take motion parameters of time $t-1$ as initial values, i.e., $\mathbf{R}^0 = \mathbf{R}(t-1)$, $\mathbf{T}^0 = \mathbf{T}(t-1)$, $\ \theta^0 = \ \theta(t-1)$.
  2. Estimate the global motion parameters at the $2k$th iteration. $(\mathbf{R}^{2k}, \mathbf{t}^{2k}) = \mathcal{G}(\ \theta^{2k-1})$, i.e., keep the local motion of previous iteration $\ \theta^{2k-1}$.
  3. Estimate the local motion parameters $\ \theta^{2k+1} = \mathcal{L}(\mathbf{R}^{2k}, \mathbf{T}^{2k})$, i.e., hold the global motion $\mathbf{R}^{2k}, \mathbf{T}^{2k}$ of the $2k$th iteration for the $(2k+1)$th iteration.
  4. Terminate the iteration when the change in error falls below a preset threshold. Then, the estimations of $\mathbf{R}(t)$, $\mathbf{T}(t)$, and $\ \theta(t)$ are obtained at time t.

- Update the 3-D model $H(t) = H(\theta(t))$, then process the next time frame.

A proof of convergence is given below. The ideas of the two-step iterative algorithm are that the operation $\mathcal{G}(\ \theta)$ finds the best global motion given known local motion, and the operation $\mathcal{L}(\mathbf{R}, \mathbf{T})$ also finds the best hand state given known global motion.

Convergence Theorem: The two-step iterative algorithm converges monotonically to a limit point with respect to certain nonnegative error measurements such as the mean square error.

Proof: We can suppose the error measurement to be the mean square error w.l.g.:

$$e^k = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{X}_i - \mathbf{P}M_G(\mathbf{R}^k, \mathbf{T}^k)M_L(\ \theta^k)\mathbf{x}_i^m||^2 \tag{5.14}$$

Since $\theta^{2k} = \theta^{2k-1}$, apply the operation $\mathcal{G}$ to estimate global motion at the $2k$th iteration.

$$(\mathbf{R}^{2k}, \mathbf{T}^{2k}) = \mathcal{G}(\ \theta^{2k-1}) = arg \min_{\{R,T\}} e(\mathbf{R}, \mathbf{T},\ \theta^{2k-1}) \tag{5.15}$$

So, the error of the $2k$th iteration is

$$e^{2k} = e(\mathbf{R}^{2k}, \mathbf{T}^{2k},\ \theta^{2k-1}) = \min_{\{R,T\}} e(\mathbf{R}, \mathbf{T},\ \theta^{2k-1}) \tag{5.16}$$

Obviously, $e^{2k} \leq e^{2k-1}$. Then, the operation $\mathcal{L}$ is applied to estimate local motion at the $(2k+1)$th iteration:

$$\theta^{2k+1} = \mathcal{L}(\mathbf{R}^{2k}, \mathbf{T}^{2k}) = arg \min_{\{\ \theta\}} e(\mathbf{R}^{2k}, \mathbf{T}^{2k},\ \theta) \tag{5.17}$$

Since we keep the global motion $(\mathbf{R}^{2k+1}, \mathbf{T}^{2k+1}) = (\mathbf{R}^{2k}, \mathbf{T}^{2k})$, the error of the $(2k+1)$th iteration is

$$e^{2k+1} = e(\mathbf{R}^{2k}, \mathbf{T}^{2k},\ \theta^{2k+1}) = \min_{\{\ \theta\}} e(\mathbf{R}^{2k}, \mathbf{T}^{2k},\ \theta) \tag{5.18}$$

Obviously, $e^{2k1} \leq e^{2k}$. So, we have

$$0 \leq e^{2k+1} \leq e^{2k} \leq e^{2k-1}\ \forall k \tag{5.19}$$

Since the error measurement cannot be negative, the lower bound occurs. Because the error sequence is nonincreasing and bounded below, this two-step iterative algorithm should converge to a limit point. Furthermore, it can be shown that the algorithm converges to a stationary point. Q.E.D.

Our divide-and-conquer approach has a lot of advantages over other methods treating hand motion as a whole in which the optimization problem is always hard to handle. In our decoupled problem, we have many alternatives to deal with each subproblem. For example, a simple but effective GA-based method was used to estimate local finger motion in [23], but here we use a sequential Monte Carlo tracking method.

5.6 Experiments

We have performed several simulation experiments to evaluate the proposed algorithm, and applied our algorithm to real image sequences.

5.6.1 Simulation

Since it is generally difficult to get the ground truth of hand motion of real video sequences, we have produced a synthetic sequence of 200 frames containing typical hand motions. This synthetic sequence will facilitate quantitative evaluations of our algorithm.

(a) (b)

Figure 5.5 Sample of our results on synthetic sequences. (a) A synthetic image, (b) the image with model aligned.

Some examples are shown in Figure 5.5. Figure 5.6 shows some of the motion parameters for comparison. The solid curves are our estimates and the dashed curves are the ground truth. The figure plots 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.

5.6.2 Real sequences

We have also performed our motion capturing algorithm with real video sequences. We have compared different schemes for local motion capturing. The first one is a random search scheme in the $\mathcal{R}^7$ space. Our experiment used 5,000 random samples. Since this scheme does not consider the finger motion constraints, it performed poorly for local motion estimation, and it even destroyed global pose determination. The second scheme is the CONDENSATION with 3,000 samples in $\mathcal{R}^7$. It performed better than the first method, but it was not robust. It seems that 3,000 samples is still not enough for this task. The third scheme is the proposed method.

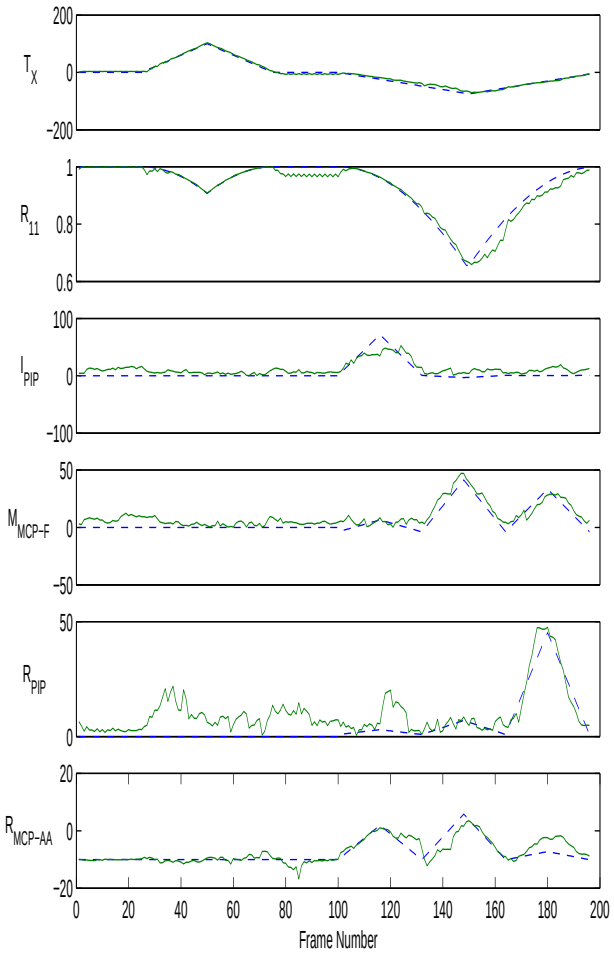


Figure 5.6 The comparison of our results and the ground truth on a synthetic sequence. The dashed curves are the ground truth, and the solid curves are our estimates.

We found that this method worked accurately and robustly. The articulation model makes the computation more efficient and the local motion estimation enhances the accuracy of hand pose determination. Some of our results are shown in Figure 5.6.2.

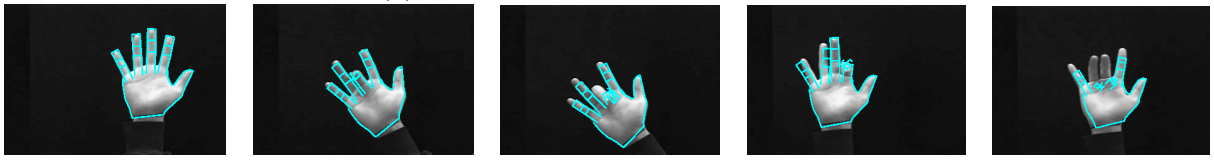
5.7 Discussion

It is a difficult problem to capture both global hand poses and local finger articulations in video sequences because of the high degrees of freedom of the articulated hand. This chapter presented a divide-and-conquer approach to this problem by separating hand poses and finger articulations. We treat the palm as a rigid planar object and use a 3-D cardboard hand model to determine hand pose based on an ICP algorithm. The local finger articulation is tracked

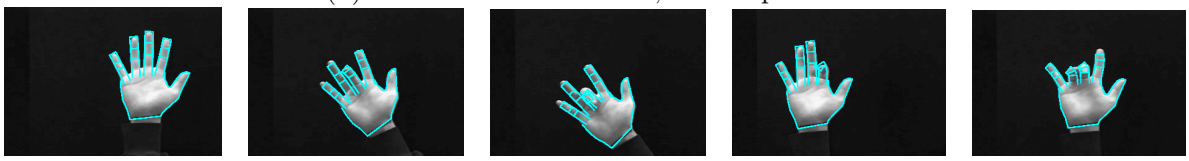
through a sequential Monte Carlo technique. The iteration between the estimation of global hand pose and that of local finger motion results in accurate motion capturing.

Our current technique assumes clean backgrounds, which largely simplify the image observation processes. We shall investigate the problem of clutter backgrounds. Since we are using a cardboard model, it would be difficult to tackle large out-of-plane rotations. Our future work includes a better hand model for out-of-plane rotations. Our current method needs manual initialization for tracking. It would be interesting if we could achieve automatic initialization through combining the 3-D model-based approach and the appearance-based approach.

(a) Random search 5,000 points in $\mathcal{R}^7$.



(b) CONDENSATION with 3,000 samples in $\mathcal{R}^7$.



(c) Our approach with 100 samples.

Figure 5.7 Comparison of different methods on real sequences. Our method is more accurate and robust than the other two methods in our experiments.

CHAPTER 6

THE DISCRIMINANT-EM ALGORITHM

6.1 Introduction

Invariant object recognition is a fundamental but challenging computer vision task, since finding effective object representations is generally a difficult problem. 3-D object reconstruction suggests a way to invariantly characterize objects. Alternatively, objects could also be represented by their visual appearance without explicit reconstruction. However, representing objects in the image space is formidable, since the dimensionality of the image space is intractable. Dimension reduction could be achieved by identifying invariant image features. In some cases, domain knowledge could be exploited to extract image features from visual inputs; however, many other cases need to *learn* such features from a set of examples when image features are difficult to define. Many successful examples of learning approaches in the area of face and gesture recognition can be found in the literature [46, 121].

Generally, characterizing objects from examples requires huge training data sets because input dimensionality is large and the variations that object classes undergo are significant. Labels or supervised information of training samples are needed for recognition tasks. The generalization abilities of many current methods largely depend on training data sets. In general, good generalization requires large and representative labeled training data sets. Unfortunately, collecting labeled data can be a tedious, if not altogether impossible, process. Although unsupervised or clustering schemes have been proposed [50, 122], it is difficult for pure unsupervised approaches to achieve accurate classification without supervision.

This problem can be alleviated by *semisupervised* or *self-supervised* learning techniques which take hybrid training data sets. This learning paradigm could be viewed as an integration

of pure supervised and unsupervised learning. These algorithms assume that only a fraction of the data is labeled with ground truth, but still take advantage of the entire data set to generate good classifiers; they make the assumption that nearby data are likely to be generated by the same class. Work in this area has been successfully applied to text classification [116, 123, 124, 125].

Expectation–Maximization (EM) is a powerful technique for self-supervised learning because it can handle missing values or hidden variables in probabilistic interference. Generally, parametric generative models could be as concise and efficient approximation of probability densities. Since such parametric generative models include strong assumptions of probabilistic structures and distribution models, however, a naïve combination of EM and generative model confronts some difficulties in practice.

Discriminant-EM (D-EM) presented in Section 6.6 is a self-supervised learning algorithm for such purposes by taking a small set of labeled data with a large set of unlabeled data. The basic idea of this algorithm is to learn discriminating features and the classifier simultaneously by inserting a multiclass linear discrminant step in the standard Expectation–Maximization iteration loop. D-EM makes the assumption that the probabilistic structure of data distribution in the lower-dimensional discrimination space is simplified and could be captured by lower-order Gaussian mixtures. Because the discrimination step in D-EM is linear, however, it has difficulty handling data sets that are not linearly separable, and input data is likely to be highly nonlinearly separable, regardless of the features used as input.

Based on nonlinear kernel discriminant analysis, Section 6.7 will present a *kernel D-EM* algorithm [126]. Kernel discriminant analysis transforms the original data space $\mathcal{X}$ to a higher-dimensional kernel feature space $\mathcal{F}$ and then projects to a lower-dimensional discrimination space Δ, such that nonlinear discriminating features could be identified, and training data could be better classified in a nonlinear feature space. Two novel algorithms are presented for sampling training data for efficient learning of nonlinear kernel discriminants. Our experiments include standard benchmark testing, view-independent hand posture recognition and invariant fingertip tracking.

In this chapter, based on the EM framework, we make use of a parametric generative model to investigate self-supervised learning. Section 6.3 gives a brief description of our generative model. Section 6.4 briefly reviews the EM algorithm. After pointing out some possible problems

with the naïve usage of EM and the generative model, we present the linear D-EM and kernel D-EM in Section 6.6 and 6.7, respectively. Experiments of invariant 3-D object recognition and content-based image retrieval will be described in Section 6.8. At the end, we will have some discussions of self-supervised learning in Section 6.9.

6.2 Learning from Hybrid Data Sets

6.2.1 Setting of learning

Different from pure supervised and unsupervised learning, the training data set is hybrid, i.e., both labeled and unlabeled data are used. For a learning task, a learner L is to learn a function f in a hypothesis space H of functions, given a set of labeled training data set $\mathcal{L}$

$$\mathcal{L} = (\mathcal{L}_x, \mathcal{L}_y) = \{(\mathbf{x}_1^l, y_1^l), (\mathbf{x}_2^l, y_2^l), \ldots, (\mathbf{x}_n^l, y_n^l)\}$$

where $\mathbf{x}^l \in \mathcal{X}^l$ is a data point, $y^l \in \mathcal{Y}^l$ is a label and n is the size of $\mathcal{L}$. For classification problems, $\mathcal{Y}$ is a discrete set. For regression problems, $\mathcal{Y}$ could be $\mathcal{R}^d$. At the same time, the learner L is also given an unlabeled data set $\mathcal{U}$:

$$\mathcal{U} = \{\mathbf{x}_1^u, \mathbf{x}_2^u, \ldots, \mathbf{x}_m^u\}$$

where m is its size. In most cases, we assume the data points in the labeled set $\mathcal{L}$ and are drawn from the same distribution as the unlabeled set $\mathcal{U}$. Sometimes, we do not make such an assumption.

In some cases, labeled data $\mathcal{L}$ are too few to perform pure supervised learning successfully, since the generalization would be very poor. However, if we have a large set of unlabeled data from the same distribution, the performance of the learner is expected to be boosted by such an unlabeled set. In these cases, if we do not have the distribution assumption, and it does not make sense to use unlabeled data, because the prior of the unlabeled data is zero.

In other cases, we cannot assume $\mathcal{L}_x$ and $\mathcal{U}$ from the same distribution. In order to take the advantage of unlabeled data $\mathcal{U}$, we should know the joint probability $p(\mathcal{L}_x, \mathcal{U})$. In this case, learner is "optimal" in $\mathcal{U}$.

The learner L aims to learn a function $y = f(\mathbf{x})$ from the hypothesis space H based on $\mathcal{L}$ and $\mathcal{U}$ such that the risk

$$R(L) = \int \frac{1}{m} \sum_{j=1}^{m} D(f(\mathbf{x}_j^u), y_j^*) dP(\mathbf{x}_1^l, y_1^l) \dots dP(\mathbf{x}_n^l, y_n^l) dP(\mathbf{x}_1^u, y_1^*) \dots dP(\mathbf{x}_m^u, y_m^*) \tag{6.1}$$

on the unlabeled data set is minimized.

Vapnik [127] gives bounds on the relative uniform deviation of training error

$$R_{train}(L) = \frac{1}{n} \sum_{j=1}^{n} D(f(\mathbf{x}_j^l), y_j^l)$$

and test error

$$R_{test}(L) = \frac{1}{m} \sum_{j=1}^{m} D(f(\mathbf{x}_j^u), y_j^*)$$

that satisfy

$$R_{test}(L) \leq R_{train}(L) + \Omega(n, m, d, \eta) \tag{6.2}$$

where the confidence interval $\Omega(n, m, d, \eta)$ depends on the number of training examples n, the number of working samples m, and the VC-dimension d of the hypothesis space H.

6.2.2 Some approaches

Some of the theoretical basis are constructed by Vapnik in his statistical learning theory of structural risk minimization (SRM) [127]. Some interesting algorithms have been proposed recently in learning from hybrid data. One approach is based on SVM to study transductive learning. Another approach is based on the EM principle for induction from hybrid data sets. Some methods are proposed to investigate the interaction between multiple learners.

6.2.2.1 SVM-based approaches

S^3VM [123, 128] are constructed using a mixture of labeled data (the training set) and unlabeled data (the working set). The working set is labeled by S^3VM. If the working set is empty, S^3VM becomes the standard SVM; if the training set is empty, S^3VM becomes a form of unsupervised learning. It is mentioned by the authors that S^3VM can be viewed as a form of *semisupervised clustering* if the working set is large but the training set is small, and a form of *transductive problem* on the other hand.

The learning of S^3VM is to find a classification function that minimizes the empirical misclassification rate and maximizes the capacity of the classification function. Two constraints are added for each sample in the working set. One constraint calculates the misclassification error as if the point were in class 1, and the other constraint calculates the misclassification error as if the point were in class -1. The objective function contains the minimum of the two possible misclassification errors. The new optimization problem can be formulated as an integer programming problem:

$$\min_{W,b,\eta,\varepsilon,z} C[\sum_{i=1}^{l} \eta_i + \sum_{j=l+1}^{l+k} min(\varepsilon_j, z_j)] + ||w|| \tag{6.3}$$

The problem that the authors look into is a transductive problem by including a small set of unlabeled date (50 unlabeled points). Their formulation only works for the linear case so far.

Tranductive SVM was proposed for text classification [124]. The problem for this type of problem is *learning from a small training set*. Another specification of this type of learning problem is that the learner can only be the best for a given working set or a given database. In this sense, it is *transductive learning*. *Inductive learning* is unnecessarily complex for this type of problem. In this type of transductive learning, the training set is very small. One could learn a model solely based on the training set by inductive learning, and then apply to the testing set. However, the generalization will not be good since we are not given enough training samples. A common problem of SVM-based methods is that they involve a formidable and expensive optimization problem.

6.2.2.2 EM-based approaches

An EM-based approach is taken to deal with an inductive learning problem in [125]. The application background is also text classification. Their situation is that the training set is small, but a large unlabeled data set is freely available. Their objective is to augment the accuracy of the classifier. By this means, the reduction in the number of labeled samples needed can be dramatic.

The basic EM scheme is easy to apply in this problem [129, 130], since the labels of unlabeled data can be treated as missing values, which can be estimated by the EM algorithm. Combing a set of unlabeled data in training, classifier accuracy can be improved by EM. They made several

assumptions. The first assumption is that data are generated by a mixture model. Another assumption is that there is a correspondence between mixture components and classes.

Since there may be a discrepancy between the generative model and the ground truth data distribution, the naïve EM would risk to fail. Still, based on EM, two extensions of EM are proposed in their research to alleviate the misfit between the modeling assumption and the unlabeled data.

Although the EM-based approach has a very sound basis and intuition, it poses some difficulties when we use parametric generative models. We should develop more robust methods when the true data distribution disagrees the generative model. And we should be able to handle the learning in a high-dimensional space.

6.2.2.3 Co-training approaches

One very interesting approach to learning from hybrid data is to exploit the cross-modality structure of training data [131, 132]. For some of the unlabeled samples, their labels can be obtained by making use of structure between the pattern distributions over different sensory modalities. The situation is that classification can be achieved by combining several different subclassifiers from different modalities. They showed that minimizing the disagreement between these subclassifiers is a good approximation to minimizing the number of misclassification of each subclassifier. Their approach was based on SOM and LVQ.

Recently, taking a similar idea as in [131], the method in [116, 133, 134] described the co-training approach to Web page classification. They gave a bipartite graph representation of this problem.

The basic idea is to train two independent classifiers rather than one. Initially, two classifiers are trained using whatever labeled training examples are available. This results in two imperfect classifiers. Each classifier is allowed to examine the unlabeled data and to pick its most confidently predicted positive and negative examples, and add them to the set of labeled examples. Then, both classifiers are now retrained on this augmented set of labeled data set, and the process is repeated until it converges. However, this algorithm is based on an assumption that the features to train two classifiers are *redundantly sufficient* to the classification problem.

There are some application examples of this approach. Blum and Mitchell [116, 133] apply it to Web page classification. A page-based classifier and a hyperlink-based classifier are trained.

Riloff and Jones [134] employ this approach to learn to classify noun phrase as positive and negative examples of locations. De Sa and Ballard [132] employ this approach to classify speech phonemes based on both the audio signal and the video signal watching the speaker's lips.

Learning from hybrid data by investigating the interaction among different subclassifiers is a very interesting and promising approach in integration of multiple cues and multimodalities.

6.3 Generative Model

We assume that the hybrid data set is drawn from a mixture density distribution of C components $\{c_j, j = 1, \ldots, C\}$, which are parameterized by $\Theta = \{\theta_j, j = 1, \ldots, C\}$. The mixture model can be represented as

$$p(\mathbf{x}|\Theta) = \sum_{j=1}^{C} p(\mathbf{x}|c_j; \theta_j) p(c_j|\theta_j) \tag{6.4}$$

where $\mathbf{x}$ is a sample drawn from the hybrid data set $\mathcal{D} = \mathcal{L} \bigcup \mathcal{U}$. We make another assumption that each component in the mixture density corresponds to one class, i.e., $\{y_j = c_j, j = 1, \ldots, C\}$. The generative model represents a strong assumption that some parts of the probabilistic structure of the data distribution are known.

6.4 Expectation–Maximization

In practice, many learning approaches face the imperfection of the training and testing data. These imperfections could be from missing sensor information, from the specific learning settings, in which some variables are unobservable or partially observable. These hidden variables should be recovered in learning. Expectation–maximization (EM) [106] is a powerful way to deal with these situations.

6.4.1 EM algorithm for density estimation

The EM algorithm is a widely used approach to learning in the presence of unobserved variables. It can be used to train Bayesian belief networks. It is also the basis for many unsupervised learning algorithms.

Essentially, many learning problems can be approached by density modeling and estimation. The density-based approach tries to estimate the joint density and allows for representation of

any related variables, so that the inference can be made possible. The density-based approach is applicable both to supervised and unsupervised learning in exactly the same way [129], and it is able to naturally handle incomplete data.

We assume that the data $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is generated independently from a mixture density as in Equation (6.4). The log likelihood of the parameters given the data set is

$$l(\Theta|\mathcal{D}) = \sum_{i=1}^{N} \log \sum_{j=1}^{C} P(\mathbf{x}_i|c_j;\theta_j)P(c_j) \tag{6.5}$$

The best density model Θ for the data set $\mathcal{D}$ maximizes the log likelihood $l(\Theta|\mathcal{D})$. However, this function is difficult to maximize numerically due to the log of a summation.

Considering the nature of the mixture density model, we can introduce a "hidden" indicator variable $\mathbf{z} = (z_1, \ldots, z_C)'$ that indicates which component generates the given data point, i.e., $z_k = 1$, if the data point is generated by the kth component, $z_k = 0$, otherwise. By this means, the maximization problem would decouple into a set of simple maximizations, given $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$. Using the indicator $\mathbf{z}$, we obtain the log likelihood of a "complete-data,"

$$l_c(\Theta|\mathcal{D}, \mathcal{Z}) = \sum_{i=1}^{N} \sum_{j=1}^{C} z_{ij} \log P(\mathbf{x}_i|\mathbf{z}_i;\Theta)P(\mathbf{z}_i;\Theta) \tag{6.6}$$

Since $\mathbf{z}$ is unknown, and l_c cannot be used directly. Instead, we work with the expectation of l_c, i.e.,

$$Q(\Theta|\Theta_k) = E[l_c(\Theta|\mathcal{X}, \mathcal{Z})|\mathcal{X}, \Theta_k]$$

As shown by Dempster [106], $l(\Theta|\mathcal{X})$ can be maximized by iterating the following two steps:

- E-step: $Q(\Theta|\Theta_k) = E[l_c(\Theta|\mathcal{X}, \mathcal{Z})|\mathcal{X}, \Theta_k]$
- M-step: $\Theta_{k+1} = arg\ max_{\Theta}\, Q(\Theta|\Theta_k)$

The E-step computes the expected complete data log likelihood, and the M-step finds the parameters that maximize this likelihood.

In this case, the E-step is just to compute the expectation of $\mathbf{z}_j$, given the data point $\mathbf{x}_j$ and the model Θ_k of the previous iteration, i.e.,

$$h_{ij} = E[z_{ij}|\mathbf{x}_i, \Theta_k] \tag{6.7}$$

since

$$E[l_c(\Theta|\mathcal{X},\mathcal{Z})|\mathcal{X},\Theta_k] = \sum_{i=1}^{N}\sum_{j=1}^{C} E[z_{ij}|\mathbf{x}_i,\Theta_k]\log P(\mathbf{x}_i|\mathbf{z}_i;\Theta)P(\mathbf{z}_i;\Theta)$$

When we assume the generative model is a mixture of Gaussians, a closed-form solution of density estimation can be obtained. At the kth iteration, the E-step computes as

$$h_{ij} = \frac{|\Sigma_j^k|^{-1/2}exp\{-\frac{1}{2}(\mathbf{x}_i-\mu_j^k)^T\Sigma_j^{-1,k}(\mathbf{x}_i-\mu_j^k)\}}{\sum_{l=1}^{N}|\Sigma_l^k|^{-1/2}exp\{-\frac{1}{2}(\mathbf{x}_i-\mu_l^k)^T\Sigma_j^{-1,k}(\mathbf{x}_i-\mu_l^k)\}} \tag{6.8}$$

The M-step re-estimates the means and covariances of the Gaussians using the data set weighted by h_{ij}:

$$\mu_j^{k+1} = \frac{\sum_{i=1}^{N} h_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} h_{ij}} \tag{6.9}$$

$$\Sigma_j^{k+1} = \frac{\sum_{i=1}^{N} h_{ij}(\mathbf{x}_i-\mu_j^{k+1})(\mathbf{x}_i-\mu_j^{k+1})^T}{\sum_{i=1}^{N} h_{ij}} \tag{6.10}$$

EM is a powerful tool to estimate the density if the probabilistic structure of data distribution is given. This approach is very similar to the soft K-means algorithm in unsupervised clustering.

6.4.2 Dealing with missing values

Since the labels of unlabeled data can be treated as missing values, the expectation–maximization (EM) approach can be applied to the self-supervised learning problem. In self-supervised learning, since the training data set $\mathcal{D}$ is a union of a set of labeled data $\mathcal{L}$ and a set of unlabeled data $\mathcal{U}$, the joint probability density of the hybrid data set can be written as

$$p(\mathcal{D}|\mathbf{\Theta}) = \prod_{\mathbf{x}_i\in\mathcal{U}}\sum_{j=1}^{C} p(c_j|\mathbf{\Theta})p(\mathbf{x}_i|c_j;\mathbf{\Theta}) \prod_{\mathbf{x}_i\in\mathcal{L}} p(y_i=c_i|\mathbf{\Theta})p(\mathbf{x}_i|y_i=c_i;\mathbf{\Theta}) \tag{6.11}$$

This equation holds when we assume that each sample is independent of others. The first part of Equation (6.11) is for the unlabeled data set, and the second part is for the labeled data.

The parameters $\mathbf{\Theta}$ can be estimated by maximizing *a posteriori* probability $p(\mathbf{\Theta}|\mathcal{D})$. Equivalently, this can be done by maximizing $\lg(p(\mathbf{\Theta}|\mathcal{D}))$. Let $l(\mathbf{\Theta}|\mathcal{D}) = \lg(p(\mathbf{\Theta})p(\mathcal{D}|\mathbf{\Theta}))$, and we have

$$\begin{aligned} l(\mathbf{\Theta}|\mathcal{D}) = \lg(p(\mathbf{\Theta})) &+ \sum_{\mathbf{x}_i\in\mathcal{U}} \lg(\sum_{j=1}^{C} p(c_j|\mathbf{\Theta})p(\mathbf{x}_i|c_j;\mathbf{\Theta})) \\ &+ \sum_{\mathbf{x}_i\in\mathcal{L}} \lg(p(y_i=c_i|\mathbf{\Theta})p(\mathbf{x}_i|y_i=c_i;\mathbf{\Theta})) \end{aligned} \tag{6.12}$$

Since the log of the sum is hard to deal with, a binary indicator $\mathbf{z}_i$ is introduced, $\mathbf{z}_i = (z_{i1}, \ldots, z_{iC})$. And $z_{ij} = 1$ iff $y_i = c_j$, and $z_{ij} = 0$ otherwise, so that

$$l(\mathbf{\Theta}|\mathcal{D}, \mathcal{Z}) = \lg(p(\mathbf{\Theta})) + \sum_{\mathbf{x}_i \in \mathcal{D}} \sum_{j=1}^{C} z_{ij} \lg(p(O_j|\mathbf{\Theta}) p(\mathbf{x}_i|O_j; \mathbf{\Theta})) \tag{6.13}$$

The EM algorithm can be used to estimate the probability parameters $\mathbf{\Theta}$ by an iterative hill climbing procedure, which alternatively calculates $E(\mathcal{Z})$, the expected values of all unlabeled data, and estimates the parameters $\mathbf{\Theta}$ given $E(\mathcal{Z})$. The EM algorithm generally reaches a local maximum of $l(\mathbf{\Theta}|\mathcal{D})$. It consists of two iterativE-steps:

- E-step: set $\hat{\mathcal{Z}}^{(k+1)} = E[\mathcal{Z}|\mathcal{D}; \hat{\Theta}^{(k)}]$
- M-step: set $\hat{\Theta}^{(k+1)} = arg\ max_\theta\, p(\Theta|\mathcal{D}; \hat{\mathcal{Z}}^{(k+1)})$

where $\hat{\mathcal{Z}}^{(k)}$ and $\hat{\Theta}^{(k)}$ denote the estimation for $\mathcal{Z}$ and Θ at the kth iteration, respectively.

6.5 Some Problems

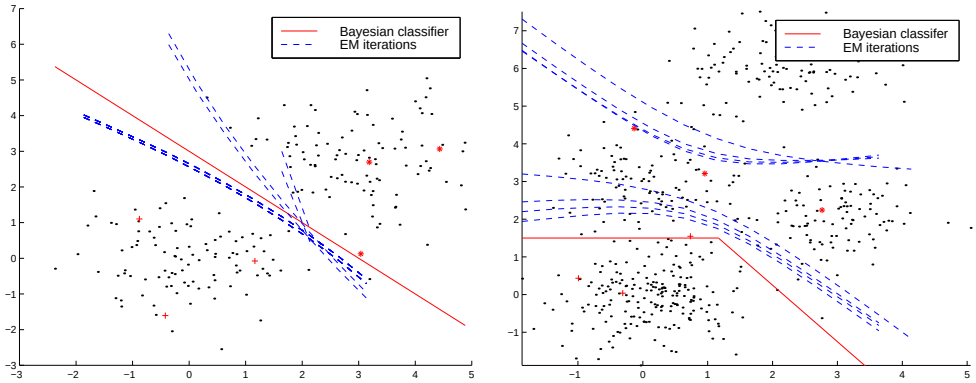
6.5.1 Model assumption

If the probabilistic structure, such as the number of components in mixture models, is known, EM could estimate true probabilistic model parameters. When the generative model does not capture the underlying data distribution, the performance of this approach could be very bad. We often assume the number of components is known and use a Gaussian or mixture of Gaussians distribution. Unfortunately, this assumption is often invalid in practice.

Generally, when we do not have such *a priori* knowledge about the data distribution, a Gaussian distribution is always assumed to represent a class. However, this assumption is often invalid in practice, which is partly the reason that unlabeled data hurt the classifier.

Figure 6.1 shows a simple example. In Figure 6.1.a, there are two classes of data drawn from two Gaussian distributions, and only six samples are labeled. EM assumes Gaussian for both classes. The iteration begins with a weak classifier learned from these labeled samples. This weak classifier is used to estimate the labels of all the other unlabeled samples. Then, all these data are employed to learn a new classifier, which labels the unlabeled samples again in the next iteration. In this special case, EM converges to the Bayesian classifier. On the other

hand, if the guess of probabilistic structure is not correct, EM may not give a good estimation. In Figure 6.1(b), one class of data is drawn from a three-component Gaussian mixture, but the model still assumes Gaussian distribution. EM fails to give a good classifier.



(a) (b)

Figure 6.1 "." represents unlabeled sample. "+" and "*" denote labeled sample. Six samples are labeled. Solid lines are Bayesian classifier, and dashed lines are the iteration results of EM. (a) Data are drawn from two Gaussian distributions. EM converges to the Bayesian classifier. (b) One class of data is drawn from a three-component Gaussian mixture, but EM still assumes Gaussian. One component is mislabeled. EM fails and unlabeled data do not help.

6.5.2 Learning in high dimensions

EM is a general and solid approach to deal with hidden variables. To model the distribution of data, parametric generative models are often employed, since they are analyzable as well as flexible. Mixture of Gaussians is a frequent choice. Although parametric generative models offer good analytical properties, they also bring some problems.

In Section 6.5.1, we explained that the structure of a generative model is a factor affecting the result of EM iteration. In practice, especially, in vision problems, learning techniques are performed in high-dimensional space. Consequently, the dimension of a generative model is also very high, such that the M-step has to estimate numerous parameters of the model. If the training data set is not large enough, the estimation could be highly biased and numerically unstable.

Although some regularization approaches have been proposed to handle such circumstances, we still ask whether it is necessary to perform learning in such a high-dimensional space? Is it possible to reduce the dimensions?

6.6 The Linear D-EM Algorithm

Since we generally do not know the probabilistic structure of a data distribution, EM often fails when structure assumption does not hold. One approach to this problem is to try every possible structure and select the best one. However, this requires more computational resources. An alternative is to find a mapping such that the data are clustered in the mapped data space, in which the probabilistic structure could be simplified and captured by simpler Gaussian mixtures. The multiple discriminant analysis (MDA) technique offers a way to relax the assumption of probabilistic structure, and EM supplies MDA a large labeled data set to select most discriminating features.

6.6.1 Linear multiple discriminant analysis

Multiple discriminant analysis (MDA) [78] is a natural generalization of Fisher's linear discrimination (LDA) in the case of multiple classes. MDA offers many advantages and has been successfully applied to many tasks such as face recognition. The basic idea behind MDA is to find a linear transformation $\mathbf{W}$ to map the original d_1-dimensional data space to a new d_2 space such that the ratio between the between-class scatter and within-class scatter is maximized in the new space.

$$\mathbf{W} = arg\,\max_{\mathbf{W}} \frac{|\mathbf{W}^T S_b \mathbf{W}|}{|\mathbf{W}^T S_w \mathbf{W}|} \tag{6.14}$$

Suppose $\mathbf{x}$ is an m-dimensional random vector drawn from C classes in the original data space. The ith class has a probability P_i, a mean vector $\mathbf{m}_i$. The within-class scatter matrix S_w is defined by

$$S_w = \sum_{i=1}^{C} P_i E[(\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T | c_i] \tag{6.15}$$

where c_i denotes the ith class. The between-class scatter matrix S_b defined by

$$S_b = \sum_{i=1}^{C} P_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T \tag{6.16}$$

where the grand mean $\mathbf{m}$ is defined as $\mathbf{m} = E[\mathbf{x}] = \sum_{i=1}^{C} P_i \mathbf{m}_i$. Details can be found in [78].

MDA offers a means to catch major differences between classes and discount factors that are not related to classification. Some features most relevant to classification are automatically selected or combined by the linear mapping $\mathbf{W}$ in MDA, although these features may not have

substantial physical meanings any more. Another advantage of MDA is that the data are clustered to some extent in the projected space, which makes it easier to select the structure of Gaussian mixture models.

It is apparent that MDA is a supervised statistical method, which requires enough labeled samples to estimate some statistics such as mean and covariance. By combining MDA with the EM framework, our proposed method, the Discriminant-EM algorithm (D-EM), is such a way to combine supervised and unsupervised paradigms. The basic idea of D-EM is to enlarge the labeled data set by identifying some "similar" samples in the unlabeled data set, so that supervised techniques are made possible with such an enlarged labeled set.

6.6.2 Expectation-Discrimination-Maximization

D-EM begins with a weak classifier learned from the labeled set. Certainly, we do not expect much from this weak classifier. However, for each unlabeled sample $\mathbf{x}_j$, the classification confidence $\mathbf{w}_j = \{w_{jk}, k = 1, \ldots, C\}$ can be given based on the probabilistic label $\mathbf{l}_j = \{l_{jk}, k = 1, \ldots, C\}$ assigned by this weak classifier.

$$l_{jk} = -\frac{p(\mathbf{x}_j|c_k)p(c_k)}{\sum_{k=1}^{C} p(\mathbf{x}_j|c_k)p(c_k)} \tag{6.17}$$

$$w_{jk} = \lg(p(\mathbf{x}_j|c_k)) \ k = 1, \ldots, C \tag{6.18}$$

By Equation (6.18), every unlabeled sample is weighted by its Mahalanobis distance to the class center. Euqation(6.18) is just a heuristic to weight unlabeled data $\mathbf{x}_j \in \mathcal{U}$, although there may be many other choices.

After that, MDA is performed on the new weighted data set $\mathcal{D}' = \mathcal{L} \bigcup \{\mathbf{x}_j, \mathbf{l}_j, \mathbf{w}_j : \forall \mathbf{x}_j \in \mathcal{U}\}$, by which the data set $\mathcal{D}'$ is linearly projected to a new space of dimension $C - 1$ but unchanging the labels and weights, $\hat{\mathcal{D}} = \{\mathbf{W}^T\mathbf{x}_j, y_j : \forall \mathbf{x}_j \in \mathcal{L}\} \bigcup \{\mathbf{W}^T\mathbf{x}_j, \mathbf{l}_j, \mathbf{w}_j : \forall \mathbf{x}_j \in \mathcal{U}\}$. Then parameters Θ of the probabilistic models are estimated on $\hat{\mathcal{D}}$, so that the probabilistic labels are given by the Bayesian classifier according to Equation (6.17). The algorithm iterates over these three steps: expectation, discrimination, and maximization. Figure 6.2 describes the D-EM algorithm.

It should be noted that the simplification of probabilistic structures is not guaranteed by linear MDA. If the components of data distribution are mixed up, it is very unlikely to find such a linear mapping. In this case, nonlinear mapping should be found so that a simple

Discriminant-EM algorithm (D-EM)
inputs: labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$
output: classifier with parameters Θ
begin Initialize: number of components C
$\mathbf{W} \leftarrow MDA(\mathcal{L})$
$lset \leftarrow Projection(\mathbf{W}, \mathcal{L})$
$uset \leftarrow Projection(\mathbf{W}, \mathcal{U})$
$\Theta \leftarrow MAP(lset)$
D-E-M iteration
E-step:
$plabel \leftarrow Labeling(\Theta, uset)$
$weight \leftarrow Weighting(plabel)$
$\mathcal{D}' \leftarrow \mathcal{L} \bigcup \{\mathcal{U}, plabel, weight\}$
D-step:
$\mathbf{W} \leftarrow MDA(\mathcal{D}')$
$lset \leftarrow Projection(\mathbf{W}, \mathcal{L})$
$uset \leftarrow Projection(\mathbf{W}, \mathcal{U})$
$\hat{\mathcal{D}} \leftarrow lset \bigcup \{uset, plabel, weight\}$
M-step:
$\Theta \leftarrow MAP(\hat{\mathcal{D}})$
return Θ
end

Figure 6.2 The D-EM algorithm.

probabilistic structure could be used to approximate the data distribution in the mapped data space. Generally, we use Gaussian or second-order Gaussian mixtures. Our experiments show that D-EM works better than pure EM.

6.7 The Kernel D-EM Algorithm

In this section, we try to extend the linear discriminant analysis to nonlinear analysis, in order to achieve better discrimination power. We take a kernel-based approach. The D-EM algorithm is generalized to the kernel D-EM algorithm in this section.

6.7.1 Nonlinear discriminant analysis

In linear discriminant analysis (LDA), a linear projection $\mathbf{W}$ is obtained, by which the original data space $\mathcal{X}$ is linearly transformed to a new space $\mathcal{Y}$. A classifier can be designed in the new space $\mathcal{Y}$. In many cases, we find that a linear projection is not enough to discriminate different patterns. It is natural to extend this linear approach to the nonlinear realm.

Nonlinear discriminant analysis finds a nonlinear mapping $\mathbf{y} = \phi(\mathbf{x})$ to nonlinearly transform the original data space $\mathcal{X}$ to a feature space (F-space) $\mathcal{F}$, in which linear discriminant analysis can be performed, i.e.,

$$J(\mathbf{V}) = \frac{|\mathbf{V}^T S_B^\phi \mathbf{V}|}{|\mathbf{V}^T S_W^\phi \mathbf{V}|} \tag{6.19}$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{c-1}]$, and $\mathbf{v}_k \in \mathcal{F}$. A linear subspace $\mathcal{V} = span\{\mathbf{v}_1, \ldots, \mathbf{v}_{c-1}\}$ of the feature space $\mathcal{F}$ is found. Patterns in the feature space $\mathbf{y} \in \mathcal{F}$ are linearly projected onto this subspace, while the corresponding patterns in the original data space $\mathbf{x} \in \mathcal{X}$ are projected nonlinearly.

Quadratic and *polynomial* mapping have been investigated [78]. For example, if $\mathcal{X}$ is a 1-D space, the quadratic mapping could be

$$\mathbf{y} = [1, x, x^2]^T \tag{6.20}$$

If $\mathcal{X}$ is a 2-D space, the complete quadratic mapping is

$$\mathbf{y} = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2]^T \tag{6.21}$$

Generally, even if the decision boundary in the feature space $\mathcal{F}$ is linear, the corresponding decision boundary in the original space $\mathcal{X}$ could be highly nonlinear and complex.

However, there are two problems related to this approach. First, the *curse of dimensionality* often makes this approach impractical. Quadratic mapping involves $O(d^2)$ terms, and n-order polynomial mapping involves $O(d^n)$ terms. The nonlinear mapping transforms the original space into a higher-dimensional feature space. Unfortunately, the dimensions of the feature space $\mathcal{F}$ will often be too high to handle. Second, parameter estimation in such a high-dimensional feature space will be impossible, since it will require unrealistic computation and data.

Sometimes, we may want to transform the original data space into an infinite-dimensional space. For example, we use an exponential mapping. It is nearly impossible to perform any analysis in such feature spaces, since it is impossible to explicitly represent a transformed pattern.

6.7.2 The kernel approach

The dot product of the form $(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$ can be represented as a kernel:

$$k(\mathbf{x}, \mathbf{y}) = (\phi(\mathbf{x}) \cdot \phi(\mathbf{y})) \tag{6.22}$$

By using the kernel presentation, we are able to calculate the value of the dot product in the feature space mapped by $\phi(\cdot)$, without having to find the mapped data $\phi(\mathbf{x})$ explicitly.

MERCER THEOREM: Mercer's theorem [135] gives conditions to construct the mapping $\phi(\cdot)$ from the eigenfunction decomposition of a kernel. If k is the continuous kernel of an integral operator $\mathcal{K} : L^2 \rightarrow L^2, (\mathcal{K})f(\mathbf{y}) = \int k(\mathbf{x}, \mathbf{y})f(\mathbf{x})d\mathbf{x}$, which is positive, i.e.,

$$\int k(\mathbf{x}, \mathbf{y})f(\mathbf{x})d\mathbf{x} \geq 0 \ \forall f \in L^2 \tag{6.23}$$

then, k can be expanded into a uniformly convergent series

$$k(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x})\phi_j(\mathbf{y}) \tag{6.24}$$

with $\lambda_j \geq 0$. In this case,

$$\Phi : \mathbf{x} \rightarrow (\sqrt{\lambda_1}\varphi_1(\mathbf{x}), \sqrt{\lambda_1}\varphi_1(\mathbf{x}), \ldots) \tag{6.25}$$

is a map into $\mathcal{F}$ such that k acts as the given dot product, i.e., $k(\mathbf{x}, \mathbf{y}) = (\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$.

For example, a simple second-order polynomial kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2$ can be represented as

$$(\mathbf{x} \cdot \mathbf{y})^2 = (x_1^2, x_2^2, \sqrt{2}x_1x_2)(x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$$

In general, the polynomial kernel

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})^d \tag{6.26}$$

corresponds to a dot product in the space of d-order monomials of the input coordinates. Many other forms of kernels are employed in the community. For example, radial basis functions

$$k(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}) \tag{6.27}$$

and sigmoid kernels

$$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \Theta) \tag{6.28}$$

have been employed in support vector machines [127], kernel PCA [136], and invariant feature extraction [137, 138, 139, 140]. The choice of the kernel k implicitly determines the mapping $\phi(\cdot)$ and the feature space $\mathcal{F}$.

6.7.3 Kernel MDA

Nonlinear discriminant analysis can be made possible by the kernel approach. Assume the d_1-dimensional original data space is $\mathcal{X}$. The number of classes is c, the size of the training data set is n, and n_j for class j, s.t., $\sum_{j=1}^{c} n_j = n$.

We try to find a nonlinear mapping $\phi(\cdot)$ to transform the original data space $\mathcal{X}$ into a higher-dimensional feature space $\mathcal{F}$, whose dimension is d_2 (d_2 can be ∞). Linear discriminant analysis is performed in such a feature space, i.e., to find a linear subspace $\mathcal{V}$ of the feature space $\mathcal{F}$ by a linear projection $\mathbf{V}$ to discriminate different patterns, i.e,

$$\Phi \quad : \quad \mathbf{x} \rightarrow \mathbf{y} \tag{6.29}$$

$$\mathbf{V} \quad : \quad \mathbf{y} \rightarrow \zeta \tag{6.30}$$

The pattern in the feature space $\mathbf{y} = \phi(\mathbf{x})$, and the pattern in the subspace $\mathcal{V}$ of feature space $\zeta = \mathbf{V}^T\mathbf{y} = \mathbf{V}^T\phi(\mathbf{x})$.

We reformulate the between-class scatter S_B and within-class scatter S_W in the feature space.

$$S_B = \sum_{j=1}^{C}(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T \tag{6.31}$$

$$S_W = \sum_{j=1}^{C}\sum_{k=1}^{n_j}(\phi(\mathbf{x}_k) - \mathbf{m}_j)(\phi(\mathbf{x}_k) - \mathbf{m}_j)^T \tag{6.32}$$

where

$$\mathbf{m} = \frac{1}{n}\sum_{k=1}^{n}\phi(\mathbf{x}_k) \tag{6.33}$$

$$\mathbf{m}_j = \frac{1}{n_j}\sum_{k=1}^{n_j}\phi(\mathbf{x}_k) \tag{6.34}$$

where $j = 1, \ldots, C$, and $\mathbf{m}$ and $\mathbf{m}_j$ are the total mean and class mean, respectively.

In feature space, the solution of linear discriminant analysis $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{c-1}]$ is given by

$$S_B\mathbf{v}_i = \lambda_i S_W\mathbf{v}_i \tag{6.35}$$

It is known that any solution $\mathbf{v}_i \in \mathcal{F}$ must lie in the span of all training samples in $\mathcal{F}$, i.e.,

$$\mathbf{v} = \sum_{k=1}^{n}\alpha_k\phi(\mathbf{x}_k) = \Phi\alpha \tag{6.36}$$

where $\alpha = [\alpha_1, \ldots, \alpha_n]^T$ and $\Phi = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]$.

We can find a projection of a data point $\mathbf{x}$ onto one coordinate of the linear subspace of $\mathcal{F}$ by

$$\begin{aligned}
\mathbf{v}^T\phi(\mathbf{x}_k) &= \alpha^T\Phi^T\phi(\mathbf{x}_k) && (6.37)\\
&= \alpha^T \begin{bmatrix} \phi^T(\mathbf{x}_1)\phi(\mathbf{x}_k) \\ \vdots \\ \phi^T(\mathbf{x}_n)\phi(\mathbf{x}_k) \end{bmatrix} && (6.38)\\
&= \alpha^T \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_k) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}_k) \end{bmatrix} && (6.39)\\
&= \alpha^T\xi_k && (6.40)
\end{aligned}$$

where

$$\xi_k = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_k) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}_k) \end{bmatrix} \quad (6.41)$$

Intuitively, ξ_k is sort of a representation of $\mathbf{x}_k$ in the feature space.

$$\begin{aligned}
\mathbf{v}^T\mathbf{m}_j &= \alpha^T\Phi^T\frac{1}{n_j}\sum_{k=1}^{n_j}\phi(\mathbf{x}_k) && (6.42)\\
&= \alpha^T\frac{1}{n_j}\sum_{k=1}^{n_j} \begin{bmatrix} \phi^T(\mathbf{x}_1)\phi(\mathbf{x}_k) \\ \vdots \\ \phi^T(\mathbf{x}_n)\phi(\mathbf{x}_k) \end{bmatrix} && (6.43)\\
&= \alpha^T\frac{1}{n_j}\sum_{k=1}^{n_j} \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_k) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}_k) \end{bmatrix} && (6.44)\\
&= \alpha^T \begin{bmatrix} \frac{1}{n_j}\sum_{k=1}^{n_j} k(\mathbf{x}_1, \mathbf{x}_k) \\ \vdots \\ \frac{1}{n_j}\sum_{k=1}^{n_j} k(\mathbf{x}_n, \mathbf{x}_k) \end{bmatrix} && (6.45)\\
&= \alpha^T\mu_j && (6.46)
\end{aligned}$$

where

$$\mu_j = \begin{bmatrix} \frac{1}{n_j}\sum_{k=1}^{n_j} k(\mathbf{x}_1, \mathbf{x}_k) \\ \vdots \\ \frac{1}{n_j}\sum_{k=1}^{n_j} k(\mathbf{x}_n, \mathbf{x}_k) \end{bmatrix} \tag{6.47}$$

We can view μ_j as the counterpart of the mean of the j class of the data space in the feature space. So,

$$\begin{aligned} \mathbf{v}^T S_B \mathbf{v} &= \sum_{j=1}^{C} \mathbf{v}^T(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T\mathbf{v} && (6.48)\\ &= \alpha^T[\sum_{j=1}^{C}(\mu_j - \mu)(\mu_j - \mu)^T]\alpha && (6.49)\\ &= \alpha^T K_B \alpha && (6.50) \end{aligned}$$

where

$$K_B = \sum_{j=1}^{C}(\mu_j - \mu)(\mu_j - \mu)^T \tag{6.51}$$

To find K_W, we have

$$\begin{aligned} \mathbf{v}^T S_W \mathbf{v} &= \sum_{j=1}^{C}\sum_{k=1}^{n_j} \mathbf{v}^T(\phi(\mathbf{x}_k) - \mathbf{m}_j)(\phi(\mathbf{x}_k - \mathbf{m}_j)^T\mathbf{v} && (6.52)\\ &= \sum_{j=1}^{C}\sum_{k=1}^{n_j} \alpha^T(\xi_k - \mu_j)(\xi_k - \mu_j)^T\alpha && (6.53)\\ &= \alpha^T K_W \alpha && (6.54) \end{aligned}$$

where

$$K_W = \sum_{j=1}^{C}\sum_{k=1}^{n_j}(\xi_k - \mu_j)(\xi_k - \mu_j)^T \tag{6.55}$$

We can write the kernel MDA as

$$J(\alpha) = \frac{|\alpha^T K_B \alpha|}{|\alpha^T K_W \alpha|} \tag{6.56}$$

The projection of a data point $\mathbf{x}$ onto the direction $\mathbf{v}$ in the feature space $\mathcal{F}$ can be obtained by

$$\mathbf{v}^T\phi(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k k(\mathbf{x}_k, \mathbf{x}) \tag{6.57}$$

or

$$\mathbf{v}^T\phi(\mathbf{x}) = \alpha^T\xi \tag{6.58}$$

which only involves a set of kernel computations and can be easily achieved.

6.7.4 Sampling data for efficiency

Because K_B and K_W are $n \times n$ matrices, where n is the size of the training set, the nonlinear mapping is dependent on the entire training samples. For large n, the solution to the generalized eigensystem is costly. Approximate solutions could be obtained by sampling representative subsets of the training data, $\{p_k | k = 1, \ldots, M, M < n\}$, and using $\tilde{\xi}_k = [k(\mathbf{x}_1, \mathbf{x}_k), \cdots, k(\mathbf{x}_M, \mathbf{x}_k)]^t$ to take the place of ξ_k.

6.7.4.1 PCA-based kernel vector selection

The first approach we propose is blind to the class labeling. We select representatives, or *kernel vectors*, by identifying those training samples which are likely to play a key role in $\Xi = [\xi_1, \ldots, \xi_n]$. Ξ is an $n \times n$ matrix, but $rank(\Xi) \ll n$ when the size of training data set is very large. This fact suggests that some training samples could be ignored in calculating kernel features ξ.

We first compute the principal components of Ξ. Denote the $n \times n$ matrix of concatenated eigenvectors with $\mathbf{P}$. Thresholding elements of $\texttt{abs}(\mathbf{P})$ by some fraction of the largest element of it allows us to identify salient PCA coefficients. For each column corresponding to a nonzero eigenvalue, choose the training samples which correspond to a salient PCA coefficient, i.e., choose the training samples corresponding to rows that survived the thresholding. Doing so for every nonzero eigenvalue, we arrive at a decimated training set, which represents data at the periphery of each data cluster. It is illustrated in Figure 6.3.

6.7.4.2 Evolutionary kernel vector selection

Another approach is to take advantage of class labels in the data. We maintain a set of kernel vectors at every iteration which are meant to be the key pieces of data for training. M initial kernel vectors, $KV^{(0)}$, are chosen at random. At iteration k, we have a set of kernel vectors $KV^{(k)}$ which are used to perform KMDA such that the nonlinear projection $\mathbf{y}_i^{(k)} = \mathbf{V}^{(k)T}\phi(\mathbf{x}_i) = \mathbf{A}_{opt}^{(k)T}\xi_I^{(k)} \in \Delta$ of the original data $\mathbf{x}_i$ can be obtained. We assume Gaussian distribution $\theta^{(k)}$ for each class in the nonlinear discrimination space Δ, and the parameters $\theta^{(k)}$ can be estimated by $\{\mathbf{y}^{(k)}\}$, such that the labeling and training error $e^{(k)}$ can be obtained by $\bar{l}_i^{(k)} = \arg\max_j p(l_j | \mathbf{y}_i, \theta^{(k)})$.

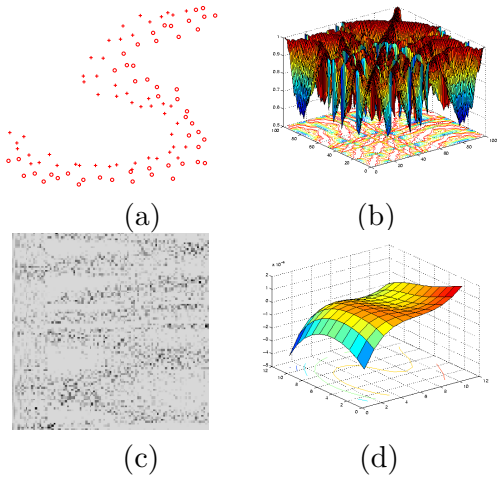


Figure 6.3 KMDA with a 2-D two-class nonlinearly separable example. (a) Original data, (b) the kernel features of the data, (c) the normalized coefficients of PCA on Ξ, in which only a small number of them are large (in black), and (d) the nonlinear mapping.

If $e^{(k)} < e^{(k-1)}$, we randomly select M training samples from the correctly classified training samples as kernel vector $KV^{(t+1)}$ at iteration $k+1$. Another possibility is that if any current kernel vector is correctly classified, we randomly select a sample in its topological neighborhood to replace this kernel vector in the next iteration. Otherwise, i.e., $e^{(k)} \geq e^{(k-1)}$, and we terminate.

The evolutionary kernel vector selection algorithm is summarized below in Figure 6.4.

6.7.5 Kernel D-EM algorithm

As an extension to expectation–maximization (EM), Wu and Huang [42] proposed a three-step algorithm, D-EM, which loops between an expectation step, a discrimination step (via MDA), and a maximization step. D-EM estimates the parameters of a generative model in a discrimination space.

We now apply KMDA to D-EM. *Kernel D-EM (KDEM)* is a generalization of D-EM, in which instead of a simple linear transformation of the data, KMDA is used to project the data nonlinearly into a feature space where the data is better separated linearly. The nonlinear mapping, $\phi(\cdot)$, is implicitly determined by the kernel function, which must be determined in advance. The transformation from the original data space $\mathcal{X}$ to the discrimination space Δ, which is a linear subspace of the feature space $\mathcal{F}$, is given by $\mathbf{V}^T\phi(\cdot)$ implicitly or $\mathbf{A}^T\xi$ explicitly.

Evolutionary Kernel Vector Selection: Given a set of training data $\mathcal{D} = (X, L) = \{(\mathbf{x}_i, l_i), i = 1, \ldots, N\}$, to identify a set of M kernel vectors $KV = \{\nu_i, i = 1, \ldots, M\}$.

$k = 0$; $e = \infty$; $KV^{(0)}$ =`random_pick(`X`)`; // Init
`do{`
  $\mathbf{A}_{opt}^{(k)}$ =`KMDA(`$\underline{X}, KV^{(k)}$`)`;// Perform KMDA
  $Y^{(k)}$ =`Proj(`$X, \mathbf{A}_{opt}^{(k)}$`)`; // Project $\mathcal{X}$ to Δ
  $\Theta^{(k)}$ =`Bayes(`$Y^{(k)}, L$`)`; //Bayesian classifier
  $\bar{L}^{(k)}$ =`Labeling(`$Y^{(k)}, \Theta^{(k)}$`)`; // Classification
  $e^{(k)}$ =`Error(`$\bar{L}^{(k)}, L$`)`; // Calculate error
  `if(`$e^{(k)} < e$`)`
    $e = e^{(k)}$; $KV = KV^{(k)}$; $k++$;
    $KV^{(k)}$ =`random_pick(`$\{\mathbf{x}_i : \bar{l}_i^{(k)} \neq l_i\}$`)`;
  `else`
    $KV = KV^{(k-1)}$; `break`;
  `end`
`}`
`return` KV;

Figure 6.4 Evolutionary kernel vector selection.

A low-dimensional generative model is used to capture the transformed data in Δ.

$$p(l|\Theta) = \sum_{j=1}^{c} p(\mathbf{V}^T \phi(\mathbf{x})|c_j; \theta_j) p(c_j|\theta_j) \tag{6.59}$$

Empirical observations suggest that the transformed data often approximates a Gaussian in Δ, and so in our current implementation we use low-order Gaussian mixtures to model the transformed data in Δ. Kernel D-EM can be initialized by selecting all labeled data as kernel vectors, and training a weak classifier based on only unlabeled samples. Then, the three steps of kernel D-EM are iterated until some appropriate convergence criterion:

- E-step: set $\hat{\mathcal{Z}}^{(k+1)} = E[\mathcal{Z}|\mathcal{D}; \hat{\Theta}^{(k)}]$
- D-step: set $\mathbf{A}_{opt}^{k+1} = \arg\max_A \frac{|\mathbf{A}^T K_B \mathbf{A}|}{|\mathbf{A}^T K_W \mathbf{A}|}$, and identify kernel vectors $KV^{(k+1)}$
- M-step: set $\hat{\Theta}^{(k+1)} = \arg\max_\theta p(\Theta|\mathcal{D}; \hat{\mathcal{Z}}^{(k+1)})$

The E-step gives unlabeled data probabilistic labels, which are then used by the D-step to separate the data. As mentioned before, this assumes that the class distributions are moderately smooth.

6.8 Experiments

In this section, KMDA is compared with other supervised learning techniques on some standard data sets in Section 6.8.1. Experimental results of D-EM and kernel D-EM algorithms on hand posture recognition and content-based image retrieval will be presented in Sections 6.8.2 and 6.8.3, respectively.

6.8.1 Benchmark tests for KMDA

We first verify the ability of KMDA with our data-sampling algorithms. Several benchmark data sets[1] are used in our experiments. The benchmark data has 100 different realizations. In [138], results of different approaches on these data sets have been reported. The proposed KMDA algorithms were compared to a single RBF classifier (RBF), a support vector machine (SVM), AdaBoost, and the kernel Fisher discriminant (KFD) [137]. RBF kernels were used in all kernel-based algorithms.

Table 6.1 Benchmark test of the kernel MDA algorithm: The average test error as well as standard deviation.

Benchmark	Banana	B-Cancer	Heart	Thyroid	F-Sonar
RBF	10.8±0.06	27.6±0.47	17.6±0.33	4.5±0.21	34.4±0.20
AdaBoost	12.3±0.07	30.4±0.47	20.3±0.34	4.4±0.22	35.7±0.18
SVM	11.5±0.07	26.0±0.47	16.0±0.33	4.8±0.22	32.4±0.18
KFD	10.8±0.05	25.8±0.46	16.1±0.34	4.2±0.21	33.2±0.17
KMDA-pca	10.7±0.25	27.5±0.47	16.5±0.32	4.2±0.21	33.5±0.17
KMDA-evol	10.8±0.56	26.3±0.48	16.1±0.33	4.3±0.25	33.3±0.17
#-KVs	120	40	20	20	40

In Table 6.1, KMDA-pca is KMDA with PCA selection, and KMDA-evol is KMDA with evolutionary selection, where #-KVs is the number of kernel vectors. The benchmark tests show that the proposed approaches achieve results comparable to those of other state-of-the-art techniques, in spite of the use of a decimated training set.

[1]The standard benchmark data sets in our experiments are obtained from `http://www.first.gmd.de/~raetsch`.

6.8.2 View-independent hand posture recognition

View-independent hand posture recognition identifies a posture in any view direction [42, 43, 141]. Some hand posture images are shown in Figure 6.5. Each row should be classified into the same posture class.

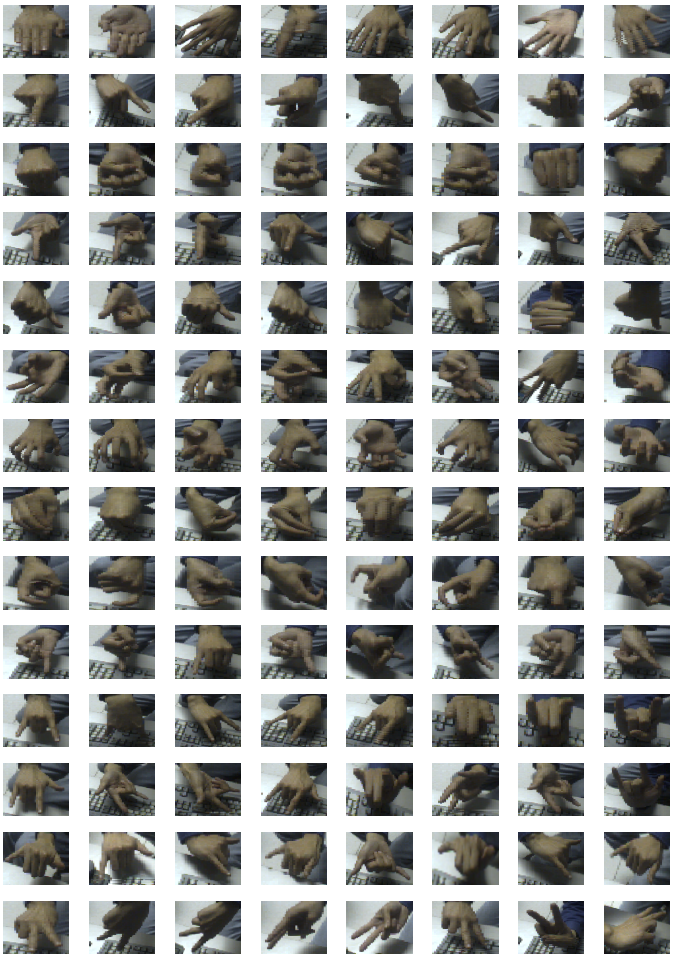


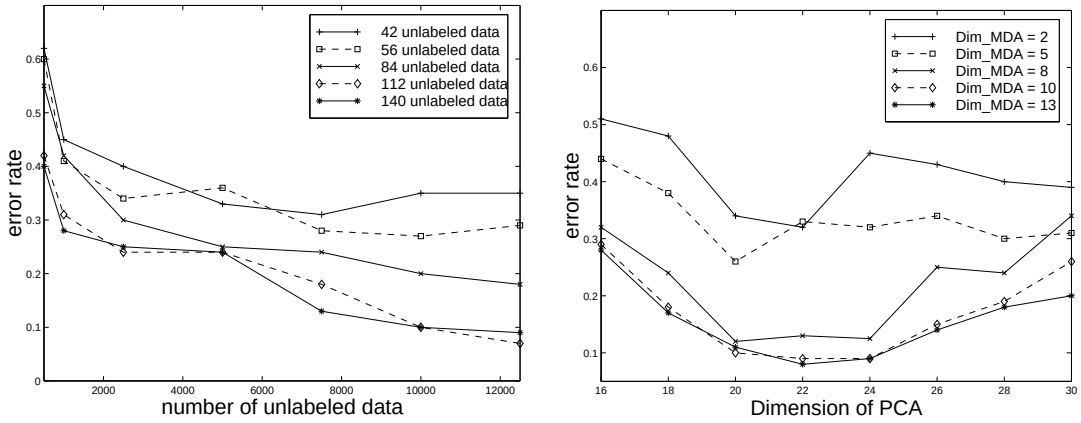
Figure 6.5 Fourteen different postures. Each row is one posture from eight different views.

The gesture vocabulary in our gesture interface is 14. The hand localization system is employed to automatically collect hand images which serve as the unlabeled data, since the localization system [90] only outputs bounding boxes of hand regions, regardless of hand postures. A large unlabeled database can be easily constructed. Currently, there are 14,000 unlabeled hand images in our database. It should be noted that the bounding boxes of some images are not tight, which introduces noise to the training data set. For each posture class, some samples

are manually labeled. To investigate the effect of using unlabeled data and to compare different classification algorithms, we construct a testing data set, which consists of 560 labeled images.

Image (I-) and eigen (E-) features are both used as hand representation in our experiments. Gabor wavelet filters with 3 levels and 4 orientations are used to extract 12 texture features, each of which is the standard deviation of the wavelet coefficients from one filter. Ten coefficients from the Fourier descriptor are used to represent hand shapes. We also use some statistics such as the hand area, contour length, total edge length, density, and second-order moments of edge distribution. Therefore, we have 28 low-level image features in total. After resizing the images to 20×20, some eigen features are extracted by PCA.

We feed the algorithm a different number of labeled and unlabeled samples. In this experiment, we use 500, 1000, 2500, 5000, 7500, 10000, 12500 unlabeled samples and 42, 56, 84, 112, 140 labeled data, respectively. In this experiment, we use the eigen features extracted by PCA with 22 principle components, and the dimension for MDA is set to 10. As shown in Figure 6.6(a), in general, combining some unlabeled data reduce the classification error by 20% to 30%.



(a) (b)

Figure 6.6 (a) The effect of labeled and unlabeled data on D-EM. (b) The effect of the dimension of PCA and MDA on D-EM.

In Figure 6.6(b), we study the effect of the dimension parameters in PCA and MDA. If fewer principle components of PCA are used, some minor but important discriminating features may be neglected so that those principle components may be insufficient to discriminate different classes. On the other hand, if more principle components of PCA are used, it would include more noise. Therefore, the number of principle components of PCA is an important parameter

for PCA. The dimension of MDA ranges between 1 and $C-1$, where C is the number of classes. We are interested in a lower-dimensional space in which different classes can be classified. In this experiment, we use 112 labeled data and 10000 unlabeled data, and we find that a good dimension parameter of PCA is around 20 to 24, and 8 to 13 for MDA.

Four classification algorithms are compared in this experiment. For E-features, the number of principle components of PCA is set to 22, and a set of 560 labeled data is used to perform MDA with dimension of 10. Using 1000 labeled training data, the multilayer perceptron used in this experiment has one hidden layer of 25 nodes. We experiment with two schemes of the nearest neighbor classifier. One is just of 140 labeled samples, and the other uses 140 labeled samples to bootstrap the classifier by a growing scheme, in which newly labeled samples will be added to the classifier according to their labels. The labeled and unlabeled data for both EM and D-EM are 140 and 10000, respectively. Table 6.2 shows the comparison.

Table 6.2 View-independent hand posture recognition: comparison among multiplayer perceptron (MLP), Nearest Neighbor with growing templates (NN-G), EM, linear D-EM (LDEM) and KDEM.

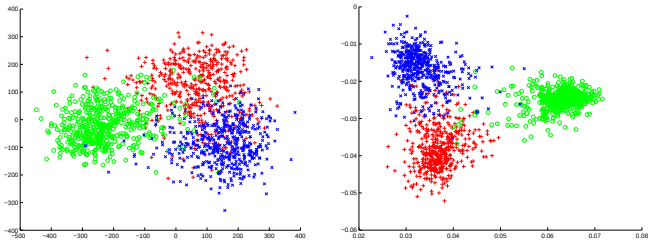
Algorithm	MLP	NN-G	EM	LDEM	KDEM
I-Feature	33.3%	15.8%	21.4%	9.2%	5.3%
E-Feature	39.6%	20.3%	20.8%	7.6%	4.9%

We observed that multilayer perceptrons are often trapped in local minima and nearest neighbor suffers from the sparsity of the labeled templates. The poor performance of pure EM is due to the fact that the generative model does not capture the ground-truth distribution well, since the underlying data distribution is highly complex. It is not surprising that LDEM and KDEM outperform other methods, since the D-step optimizes the separability of the classes. Some typical images classified and misclassified by LDEM and KDEM are shown in Figure 6.7.

Finally, note the effectiveness of KDEM. We find that KDEM often appears to project classes to approximately Gaussian clusters in the transformed space, which facilitates their modeling with Gaussians. Figure 6.8 shows typical transformed data sets for linear and nonlinear discriminant analysis, in a projected 2-D subspace of three different hand postures.

(a) (b) (c)

Figure 6.7 A comparison of linear D-EM and kernel D-EM. (a) Some correctly classified images by both LDEM and KDEM. (b) Images that are mislabeled by LDEM, but correctly labeled by KDEM. (c) Images that neither LDEM nor KDEM can correctly classify.



(a) (b)

Figure 6.8 Data distribution in the projected subspace. (a) Linear KMDA. (b) Kernel KMDA. Different postures are more separated and clustered in the nonlinear subspace by KMDA.

6.8.3 Transductive content-based image retrieval

In order to give some analysis and compare several different methods, we manually label an image database of 134 images, which is a subset of the COREL database. Our dataset has seven classes: airplane, bird, car, church painting, flower, mountain view, and tiger. All images in the database have been labeled as one of these classes. In all the experiments, these labels for unlabeled data are only used to calculate classification error.

To investigate the effect of the unlabeled data used in D-EM, we feed the algorithm a different number of labeled and unlabeled samples [142, 143, 144, 145]. The labeled images are obtained by relevance feedback. When using more than 100 unlabeled samples, the error rates drop to less than 10%. From Figure 6.9, we find that D-EM brings about 20% to 30% more accuracy. In general, combining some unlabeled data can largely reduce the classification error when labeled data are very few.

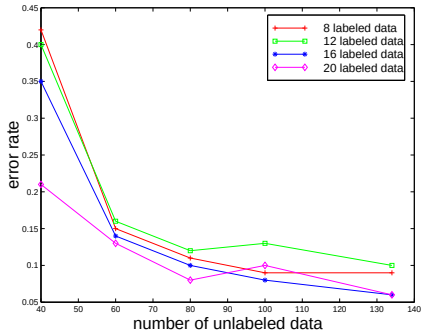


Figure 6.9 The effect of labeled and unlabeled data in D-EM. Error rate decreases when adding more unlabeled data. Combining some unlabeled data can largely reduce the classification error.

We test and compare four methods. The first one is to weight each feature by relevance feedback (WRF) [9], in which 37 image features are precalculated and prestored. The top 20 most similar images are obtained through ranking each image by comparing the Mahalanobis distances to the means of query images. The second method is a simple probabilistic method (SP), in which both classes (relevant and irrelevant) are assumed Gaussian distributions, and the model parameters are estimated by feedback images. The third method is the basic EM (EM) algorithm, which assumes Gaussian distributions for both classes. The fourth is the D-EM algorithm. In the last three probabilistic methods, the label of each image is given by maximizing *a posteriori* probability:

$$l_j = arg\max_k p(c_k|\mathbf{x}_j)$$

We also compare a set of image features (I-Features) and eigen features (E-Features). We use the same image features as in WRF [9]. The eigen features are extracted by PCA, in which the number of principle components is 30, and the image resolution is reduced to 20×20. Except for WRF, both I-features and E-features are tested.

These four methods are compared on this fully labeled database. The results are shown in Table 6.3. Classification error for each method is calculated for evaluation, although these errors are not available for the training. Suppose the database has N samples, C classes, and the kth class has N_k samples, and $N = \sum_{k=1}^{C} N_k$. The method to calculate error in WRF is different from the other three methods. In WRF, if the query images belong to the jth class, and m_j samples in the top N_j belong to the jth class, the error for this query is defined as

$$e_j = \frac{2(N_j - m_j)}{N} \tag{6.60}$$

In the other three methods, if there are m samples in total that are not correctly labeled, the error is defined as

$$e_j = \frac{m}{N} \tag{6.61}$$

The average error is obtained by averaging over M experiments, i.e.,

$$e = \frac{\sum_{j=1}^{M} e_j}{M} \tag{6.62}$$

Table 6.3 Error rate comparison among different algorithms. All comparisons are based on the first time relevance feedback with six relevant and six irrelevant images. D-EM outperforms the other three methods.

Algorithm	I-features	E-features
WRF	6.3%	N/A
SP	21.2%	15.7%
EM	23.4%	25.8%
D-EM	3.9%	5.3%

Our algorithm is also tested by several large databases. The COREL database contains more than 70,000 images over a wide range of more than 500 categories with 120×80 resolution. The VISTEX database is a collection of 832 texture images.

6.9 A Discussion on Self-Supervised Learning

Previous sections presented linear D-EM and kernel D-EM for self-supervised learning techniques. Extensive experiments show the effectiveness of the D-EM approach in many learning tasks. This section will give a further discussion on self-supervised learning.

6.9.1 A new learning paradigm

The cognitive processes of human beings are highly complicated. It is very difficult to represent such processes explicitly. However, there are some general measurements of cognitive ability. One of them is *induction*, which is to learn a model from a set of given examples. Obviously, the result of inductive learning largely depends on the training examples. *Deduction* is to learn a better model for a specific domain given a generic knowledge. More interestingly,

human beings are also able to conduct *transduction*, which is to transduce a specific domain model to another unknown domain.

Supervised learning and *unsupervised learning* are the two main learning paradigms investigated in current machine learning research. We can look the supervised learning and unsupervised learning as two extremes, since supervised learning tries to "teach" the learner all the examples, while unsupervised learning leaves all the studying to the learner without teaching anything. Pure supervised learning is probably not able to perform transduction because it is incapable in an unknown world. On the other hand, pure unsupervised learning is not able to perform effective induction because of lack of supervision. There are also some paradigms in the middle such as *reinforcement learning* and *supervised clustering*. Reinforcement learning tries to tell the learner "correct" and "incorrect" instead of "how." Supervised clustering essentially is an unsupervised learning technique. These two techniques start to find an answer to the question: How much should we teach a learner?

One answer is given by Vapnik in his *support vector machine* theory [127] that the classification boundary depends only on the *support vectors* instead of the whole training data set, which means the *support vectors* are the minimum training samples needed. This fact suggests that it might not be necessary to have every sample labeled in supervised learning. Although the identification of these support vectors is not trivial, it motivates us to think about the roles of nonsupport vectors.

If the probabilistic structure of data distribution is known, parameters of probabilistic models can be estimated by unsupervised learning alone, but it is still impossible to assign class labels without labeled data [78]. This fact suggests that labeled and unlabeled training data are both need in learning, in which labeled data (if enough) can be used to label the class and unlabeled data can be used to estimate the parameters of generative models.

We introduce a new learning paradigm called *self-supervised learning* (SSL) to unify induction, deduction, and transduction, by using both labeled and unlabeled training data. Labeled training data sets represent the knowledge we should teach the learner, and the unlabeled set is the inputs from an unknown world, which should be explored by the learner itself.

Some fundamental questions are raised in Section 6.9.2 and the new self-supervised learning paradigm is given in Section 6.9.3.

6.9.2 Some fundamental questions

Unlabeled data contain information about the joint distribution over features. If the parametric forms of the probability densities are known, the parameters can be estimated by unsupervised learning alone, but it is impossible to assign class label without labeled data. The assumption of the generative model must be held to reach this strong conclusion. This conclusion suggests that labeled data (if enough) can be used to label the class of each component and unlabeled data can be used to estimate the parameters of the model.

There should be an assumption about the distribution of the unlabeled data set. There are several possibilities. One is that the unlabeled set is from the same distribution as the labeled set, and we may know exactly the form of the distribution density. Second is that the distribution of the unlabeled set has some deviation from the labeled set, but we know the relationship between these two distributions, such as joint density or conditional density.

We should answer some important questions:

- Why do we use unlabeled data sets in supervised learning?
- What are the objectives and advantages?
- What are the assumptions about the unlabeled data, if any?
- When do unlabeled data help, and when do they hurt?
- What are the roles of labeled and unlabeled data?
- What are the necessary conditions for successful model transduction?
- How is the unlabeled data set to be used?

6.9.3 Self-supervised learning

6.9.3.1 Problem formulation

Self-supervised learning (SSL) employs a hybrid training data set $\mathcal{D}$ which consists of a labeled data set $\mathcal{L} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, where $\mathbf{x}_i$ is feature vector, y_i is label and n is the size of the set, and an unlabeled data set $\mathcal{U} = \{\mathbf{x}_i, i = 1, \ldots, m\}$, where m is the size of the set.

Generally, we make an assumption here that $\mathcal{L}$ and $\mathcal{U}$ are from the same distribution. We have a risk function

$$R(\Theta) = \int L(y, f(x, \Theta(\mathcal{L}, \mathcal{U})))dF(x, y) \tag{6.63}$$

where $L(y, f(x, \Theta(\mathcal{L}, \mathcal{U})))$ is the loss function. The learning is to find the function $f(x, \Theta^*)$ minimizing the risk without knowing the joint p.d.f. $F(x, y)$. The function $f(x, \Theta^*)$ depends on both labeled and unlabeled training data sets. Essentially, the classification can be represented as

$$y_i = arg \max_{j=1,...,C} p(y_j|\mathbf{x}_i, \mathcal{L}, \mathcal{U} : \forall \mathbf{x}_i \in \Psi) \tag{6.64}$$

where Ψ is a subset of the whole data space Ω and C is the number of classes. Consequently, the decision is made based on both $\mathcal{L}$ and $\mathcal{U}$. Inputs of the classifier are drawn from Ψ. According to different Ψ, self-supervised learning has different special cases.

6.9.3.2 Induction

When $\Psi = \Omega$, self-supervised learning becomes inductive learning.

$$y_i = arg \max_{j=1,...,C} p(y_j|\mathbf{x}_i, \mathcal{L}, \mathcal{U} : \forall \mathbf{x}_i \in \Omega) \tag{6.65}$$

Different from conventional learning paradigms, inductive learning depends both on supervised data set $\mathcal{L}$ and unsupervised data set $\mathcal{U}$. If $\mathcal{L} = \phi$, it degenerates to pure unsupervised learning. If $\mathcal{U} = \phi$, it degenerates to pure supervised learning.

Since it is not necessary to have every training sample labeled, the most interesting question in inductive learning is how much supervised information is needed. Generally, we use a large unlabeled training set while employing a relatively small labeled set.

6.9.3.3 Deduction

When Ψ is a subset of U, i.e., $\Psi \subset \mathcal{U}$, the problem degenerates to SSL deduction.

$$y_i = arg \max_{j=1,...,C} p(y_j|\mathbf{x}_i, \mathcal{L}, \mathcal{U} : \forall \mathbf{x}_i \in (\Psi \subset \mathcal{U})) \tag{6.66}$$

When we have a rough generic model of a domain, deduction is to learn a specific model which works in a specific subdomain based on this rough model. Deduction can be treated as a special case of transduction. For example, a generic model classifies men and women; deduction learns a specific model to differentiate Asian men and Asian women.

6.9.3.4 Transduction

When $\Psi = \mathcal{U}$, self-supervised learning becomes transductive learning.

$$y_i = arg \max_{j=1,\ldots,C} p(y_j|\mathbf{x}_i, \mathcal{L}, \mathcal{U} : \forall \mathbf{x}_i \in \mathcal{U}) \tag{6.67}$$

Generally, the classifier obtained from inductive learning could be highly nonlinear, and a huge labeled training set is required to achieve good generalization. However, the requirement of generalization could be relaxed to a subset of the whole data space. The generalization of transductive learning is only defined on the unlabeled training set $\mathcal{U}$, instead of the whole data space Ω.

In human cognition, we usually can learn a good model in a specific domain (i.e., a subset). This model may not be good for other domains. But the model can be adapted to solve problems in similar domains. This cognition process can be captured by SSL transduction, which can be used to transduce a specific model to another specific model. For example, based on a specific model which is to classify Asian men and Asian women, transduction learns another specific model to recognize European men and European women. It can be illustrated by an example of the nonstationary color model adaptation in Chapter 3.

CHAPTER 7

VISION-BASED GESTURE INTERFACE SYSTEMS

This chapter will present two interesting prototype vision-based gesture interface systems. One of them is called "Paper-Rock-Scissors," in which people could play a simple interactive video game against computers. The other is called "Visual Panel," through which people could control a remote display only using their fingers and an arbitrary paper.

7.1 "Paper-Rock-Scissors": An Interactive Video Game

The design of an interesting interactive video game, paper-rock-scissors, will be presented in this section. Using live video inputs, the system could localize the user's hand and recognize the hand postures. The framework of the design will be described in Section 7.1.1. The four subsystems of hand localization, motion capturing, posture recognition and hand animation will be given in Section 7.1.2, 7.1.3, 7.1.4 and 7.1.5, respectively.

7.1.1 System framework

The whole gesture interface consists of four subsystems: the hand localization system, articulate motion capturing system, gesture recognition system, and animation system. The framework is shown in Figure 7.1.

7.1.2 Localization system

Our localization system is based on color segmentation. Motion segmentation and region-growing method are also employed to make the system more robust and accurate without

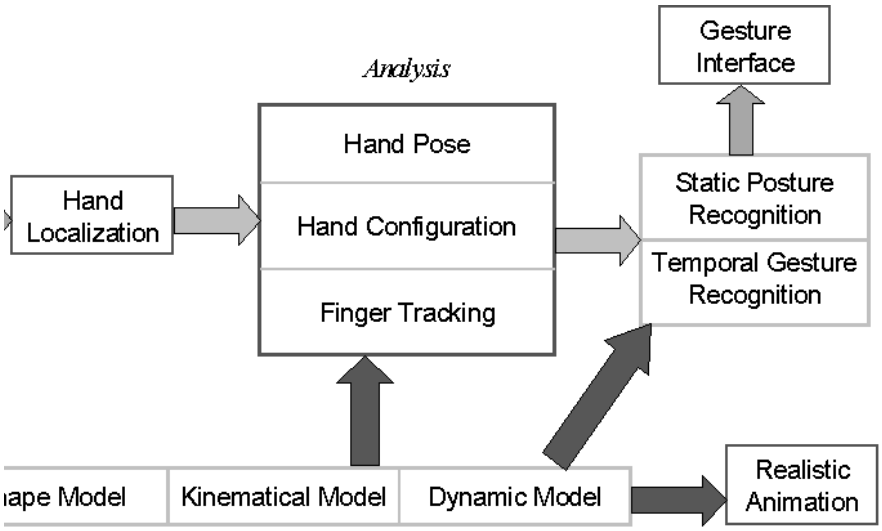


Figure 7.1 The framework of our vision-based gesture interface.

introducing too much computation cost. Figure 7.2 shows the overview of the localization system.

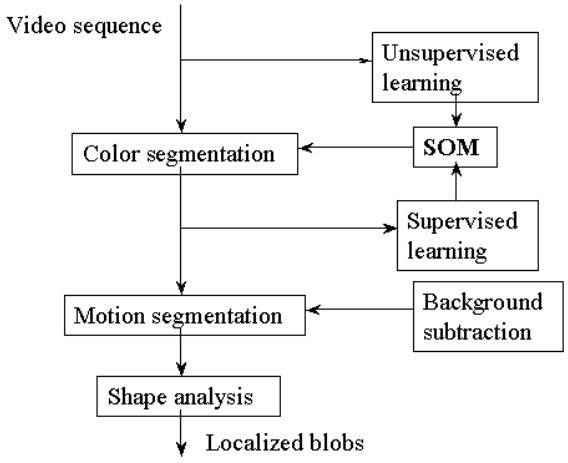


Figure 7.2 Hand localization system framework.

The first frame taken by a camera is used to initially train the SOM by the proposed self-organizing clustering algorithm that has been described in Section 3.2.3. In this initialization stage, the color distribution in the scene is initially mapped. In our experiments, the training is fast (less than 1 second) with a 640×480 color image. The inputs of the SOM are the HSI value of pixels, and the outputs are the indexes of winning nodes of SOM through competition. Typically, it takes fewer than six nodes to segment indoor working environments.

For each newly captured color image at time frame t, the SOM is transduced by the algorithm described in Section 3.4. Such SOM is used to segment the input image to find different color regions. This stage can be done on a lower resolution image to make the segmentation faster. Morphology operators are used to get rid of noise. After each pixel has been labeled, the SOM should be updated again by the supervised updating scheme described in the Section 3.4. The labeled training data set is randomly selected from the segmented image, ignoring those that are too bright or too dark.

Since there may be many different colors in the working space, and if the system does not specify which color to track, how to determine what to track is a problem. One possible solution is to specify a color region such as a human hand or face. Another solution is to use some rules to automatically find an interested color from motion intention. If we detect a motion region by examining the frame difference or optic field, the color of that region is taken to be the interested color.

There are some cases in which several objects have nearly the same color. For instance, tracking two faces or two hands is needed in recognizing sign languages. When the color segmentation algorithm separates them from the background, there are some ways to locate each region. One method is to use the same scheme of our self-organizing clustering to find the centroid of each isolated blob. Another way is to use a region-growing technique to label each blob or use some heuristics to find bounding boxes.

A typical hand-tracking scenario is controlling the display or simulating a 3-D mouse in desktop environments. A camera mounted at the top of the desktop computer looks below at the keyboard area to give an image sequence of moving a hand. Another typical application is to track a human face. Our localization system is able to simultaneously localize multiple objects, which is useful in tracking a moving human.

Since our localization system is essentially based on a global segmentation algorithm, it does not largely rely on the tracking results of previous frames. Even if the tracker may for some reason gets lost in some frames, it can recover by itself without interfering the subjects. In this sense, the tracking algorithm is very robust.

Our proposed system can handle changing lighting conditions to some extent because of the transduction of the SOM color classifier. At the same time, since hue and saturation are given more weight than intensity, our system is insensitive to changes in lighting intensity

such as when objects are shadowed or the intensity of the light source changes. However, there are still some problems. Insufficient lighting, too strong lighting, or very dark or bright backgrounds may pose problems for the color segmentation algorithm, since hue and saturation become unstable and the system does not give more weight to intensity. If lighting conditions change dramatically, the color segmentation algorithm may fail since the transduction cannot be guaranteed.

7.1.3 Motion capturing system

A model-based approach is taken to capture the articulated hand motion by employing a 3-D hand model. Here, a kinematical hand model is employed. Articulated hand motion is decoupled from its global hand motion and local finger motion, in which global motion is parameterized as the rotation and translation of the palm, and local motion is parameterized as the state of the hand. Global hand poses are captured by a robust pose determination algorithm based on ICP, and local finger articulations are estimated by a sequential Monte Carlo tracking algorithm, respectively, in an iterative fashion.

7.1.4 Posture recognition system

The hand posture recognition system recognizes static hand postures from the hand images located by the hand localization system. For each frame, the localization system gives a bounding box of the hand, and hand posture recognition is based on the cropped hand image.

We look into fourteen different hand postures, which are shown in Section 6.8.2. Eigen features are used as hand representation, due to the simplicity of feature extraction. The feature extractors are PCA and MDA. The training algorithms can take both linear and kernel D-EM algorithms. The generative model is a Gaussian mixture model. The training is off-line, in which the parameters of the generative model and an MDA projection matrix are estimated. Recognition is on-line based on Bayesian decisions. Some of the experiments are given in Section 6.8.2. Figure 7.3 illustrates the hand posture recognition subsystem.

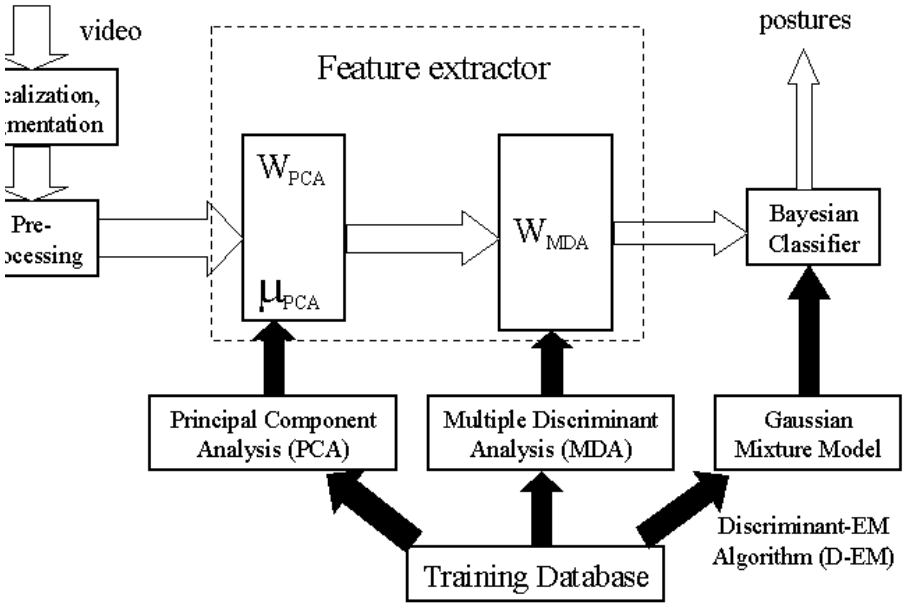


Figure 7.3 The hand posture recognition subsystem.

7.1.5 Animation system

In our hand animation system, hand model is animated by keyframe-based methods. The starting, ending, and several transitions hand states are given as the keyframes. The animation sequences are obtained by a cubic spline interpolation scheme.

If the model is driven by the set of joint angles which represents the state of hand, hand states can be interpolated by prespecified key-frame states. If the model is driven by the position of fingertips, an inverse kinematics problem must be solved. Although it is simple to implement, the drawback of this approach is apparent. In order to obtain a realistic effect, a large number of control points must be specified along the fitting curves. To reduce the amount of motion specification, some knowledge about hand motion should be built in the animation system to execute certain aspect of movement autonomously. Some high-level control schemes and physical rules can be used to achieve this goal: however, the disadvantage is lack of interactivity.

7.1.6 System performance and demos

The system is implemented in C++ on SGI O2 R10000 machines. The SOM-based hand localization system runs at 20-25 Hz, and the hand posture recognition subsystem runs at

around 15 Hz. Since the articulated motion capturing is very expensive, it is far from real-time. The animation looks very natural and realistic.

A simple demo is made to show some of the capacity of our gesture interface. It is a virtual game of *paper-rock-scissors*, which uses three hand signs: "rock," "scissors," and "paper," shown in Figure 7.4.

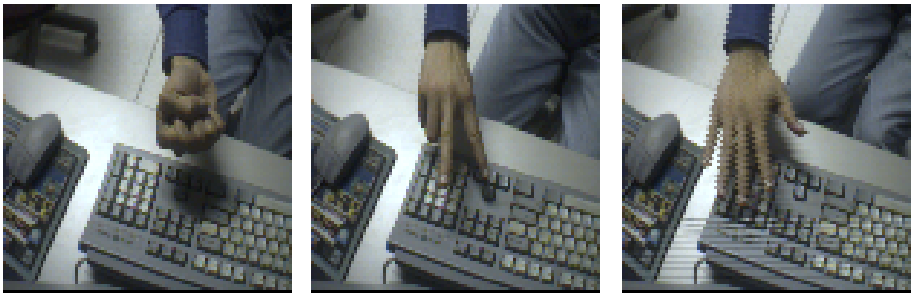


Figure 7.4 The children's game *paper-rock-scissors*.

There is a traditional children's game with these hand signs, in which rock beats scissors, scissors beats paper, and paper beats rock. In this demo, a person will play against a computer, and a second computer tracks and recognizes the sign made by the human and keeps the score. The computer player randomly generates one of these three hand signs, and the transition from one to another is shown in another window by the animation system.

7.2 "Visual Panel": A Vision-based Mobile Input System

The goal of the virtual panel system is to use an arbitrary quadrangle-shaped plane object, such as a paper, and a tip pointer, such as a fingertip and a pen, to serve as a natural and convenient input device for accurately controlling remote displays, based on computer vision techniques. Using fingertips, a natural body part, we developed an intuitive and immersive way for accurate interaction with remote and large displays.

The system consists of virtual panel tracking, tip pointer tracking, homography calculation/updating, and action detection/recognition. The whole system is shown in Figure 7.5.

Part of the user input is analyzed from video sequences by a panel tracker and a tip pointer tracker. The panel tracker can accurately track an arbitrary quadrangle-shaped plane object by outputting the positions of the four corners. Since an edge-based dynamical programming technique is employed in the panel tracker, it is quite robust and reliable, even some of the

corners of the panel are occluded. At the same time, since the positions of the corners are calculated by intersecting four lines of the quadrangle, such that the positions are calculated in subpixels, calculation of homography is more accurate.

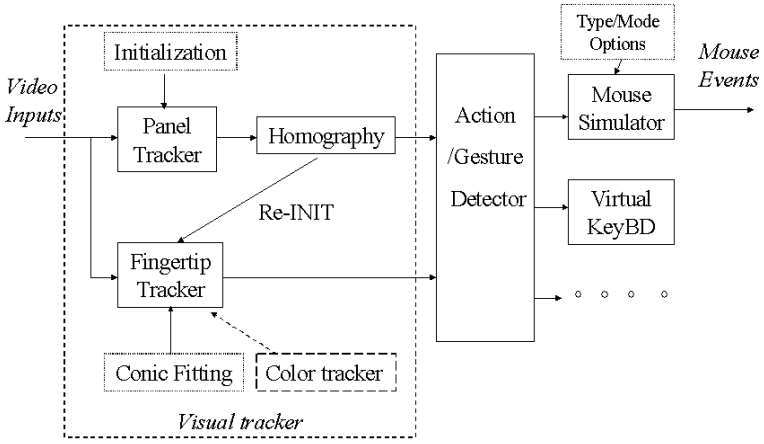


Figure 7.5 The system of "Virtual Panel", which consists of panel tracker, pointer tracker, action detector, and message generator.

The mapping will be constructed between this panel and a remote display by calculating a homography transformation, through which any point on the panel will be mapped to the corresponding position on the remote display. Obviously, remote mouse and remote keyboard can be used to control a remote display. However, such devices are expensive and emit electromagnetic radiation.

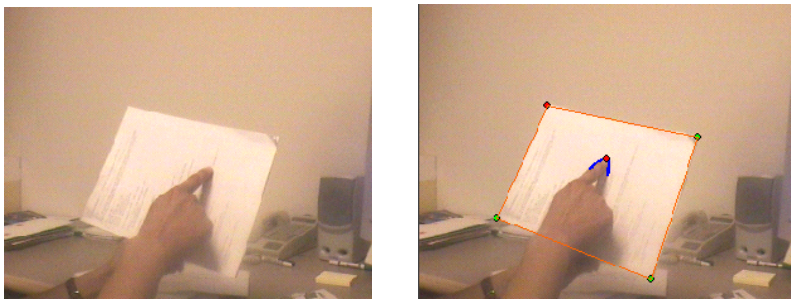
In the virtual panel system, users can use their fingertip as a mouse to simulate a cursor for the remote display. Consequently, the tracking of the tip pointer should be quite accurate and stable, since a small error of tip position will be magnified in the remote large screen. For instance, we assume the resolution of input video is 320×240 and the remote display is 1024×768. Since generally the panel in the image is roughly half the size of the image, it is obvious that the tracking error of 1 pixel will incur around 6 pixels error in the large display, which will make the mapped cursor position very shaky. This problem is solved in our system by representing any tip pointer as a conic and fitting a parametric conic to image observations. Therefore, the tip position can also be calculated in subpixels such that the error can be reduced.

The current system simulates the clicking/pressing gestures by holding the tip pointer on the position for a while. By this means, the system is capable of having two inputting methods: virtual mouse and virtual keyboard. Obviously, the position of the tip pointer can be mapped

to the remote display such that a cursor can be simulated. We also use a paper with a keyboard pattern printed on it as a virtual keyboard, by which users can point the keys on the paper to input texts.

The message generator in the system gets inputs from the action detector, and issues various mouse and keyboard events, according to the different user input methods.

Figure 7.6 shows the basic idea of visual tracking of the virtual panel system. Figure 7.6(a) is one frame of the video input, and Figure 7.6(b) shows the tracking results of the panel and the fingertip.



(a) (b)

Figure 7.6 The tracking in the visual panel system. (a) Input image. (b) Tracking outputs: A tracked panel and a tracked fingertip.

The virtual panel system is also scaleable and extendable, which makes the system portable, easy-to-use and cost-efficient.

- **Camera setting**: The setting of the camera can be quite flexible. It can be anywhere as long as the panel is not totally occluded. If the camera is fixed, the panel cannot move too far, and it should be in the field of view of the camera. We can mount the camera on the ceiling. The user can rotate, translate, and tilt the panel to reach a comfortable pose for use. Obviously, users should not let the normal of the panel be vertical to the optical axis of the camera. Under some circumstances when the user wants to walk around, we can mount a camera on top of his head by wearing a hat, or on his shoulders, such that the user can be anywhere to interact with the computer. It would be quite useful for a speaker who wishes to walk around when giving his PowerPoint presentation.

- **Panel design**: At the same time, the panel can be anything as long as it is quadrangle-shaped. For example, we can use white paper with some printing on it, which is widely available in offices and homes.

- **Tip pointers**: The system allows arbitrary tip pointers, such as fingertips and pens. In our usability studies, many users prefer using pens to fingertips in some applications such as finger painting, since pens are more intuitive for them, although using fingertips is of more fun.

- **Clicking**: The current virtual panel system simulates clicking and pressing by holding the tip pointer for a while, since this means is easy to use and more reliable for vision techniques. Alternatively, the system can use some natural gestures to act as clicking.

7.2.1 Homography

Since we use an arbitrarily rectangle-shaped panel to control the cursor position on the remote display, we have to know the mapping between a point on the panel and a point on the display. Furthermore, what is available is an image sequence of the panel which may undergo arbitrary motion (as long as the image of the panel does not degenerate into a line or a point), so we also need to know the mapping between a point in the image plane and a point on the panel. We assume the camera performs a perspective projection. As the display, the panel, and the image plane are all planes, both above relationships can be described by a *plane perspectivity*, to be explained below.

Given a point $\mathbf{p} = [x, y]^T$ on a plane Π, we use $\tilde{\mathbf{p}} = [x, y, 1]^T$ to denote its homogeneous coordinates. Then, the plane perspectivity between planes Π and Π' is described by a 3×3 matrix $\mathbf{H}$ such that

$$\lambda \tilde{\mathbf{p}}' = \mathbf{H}\tilde{\mathbf{p}}$$

where λ is an arbitrary nonzero scalar. This implies that the homography matrix is only defined up to a scale factor, and therefore has 8 degrees of freedom. If four couples of corresponding points (no three of them are collinear) are given, the homography matrix can be determined.

It is not difficult to see that the composition of two plane perspectivities is still a plane perspectivity. Thus, the mapping between the images of the panle and the remote display can be described by a homography matrix. This is very important because what we really need is to

use the detected tip position in the image to control the cursor position on the remote display. The composed homography can be easily determined once the four corners of the panel are located in the image. As we know the dimension of the display, we compute the homography by mapping each corner of the panel to a corner of the display.

7.2.2 Tracking a quadrangle

The panel can be represented by a quadrangle:

$$\mathcal{Q} = \{l_1, l_2, l_3, l_4\} \tag{7.1}$$

where l_i is a margin line. It can also be represented by four corners $\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$. There is a set of image edge features associated with each line.

$$\mathcal{E} = \{e_1, e_2, e_3, e_4\} \tag{7.2}$$

The gradient of each pixel on any edge is represented by $e_i^k = (e_i^k(x), e_i^k(y))$. In spite of its location, the appearance of an edge can be represented by a set of statistics, such as edge length and average gradient. Here, we represent the appearance of an edge as a random vector $\mathbf{X} = \{G, I\}$, where G is the gradient and I is the intensity. We assume the distribution of $\mathcal{X}$ a Gaussian, i.e.,

$$\mathbf{X} \sim N(\mu_x, \Sigma_x) \tag{7.3}$$

At time frame t, the location of the quadrangle is at $\mathcal{Q}(t) = \{p_1(t), p_2(t), p_3(t), p_4(t)\}$, and the appearance of the quadrangle is $\mathbf{X}(t)$. We assume at time $t+1$, these four corner points will be in a range $\mathcal{D}_i$ around $p_i(t)$, respectively. The tracking can be formulated as a MAP problem:

$$\mathcal{Q}^*(t+1) = \arg\max_{Q} p(\mathcal{Q}(t+1)|\mathcal{Q}(t), \mathbf{X}(t), \mathbf{X}(t+1)) \tag{7.4}$$

This problem can be approximated by

$$\mathcal{Q}^*(t+1) = \arg\max_{Q} p(\mathbf{X}(t+1)|\mathcal{Q}(t), \mathbf{X}(t) : \{D1, D2, D3, D4\}) \tag{7.5}$$

Obviously, this is a formidable searching problem. To illustrate this, we assume the size of each search area of D_i is N. The complexity of the exhausted search for this problem is $O(4^N)$.

However, since the four margins of the quadrangle are sequentially connected, this problem can be solved by the *dynamic programming* technique.

$$\mathcal{Q}^*(t+1) = \arg\max_{Q} \sum_{i=1}^{4} p(X_i(t+1)|X_i(t), Q_i(t) : D_i(q_i(t), q^*_{i-1}(t))) \tag{7.6}$$

$$= \arg \max_{\{q_1,q_2,q_3,q_4\}} \sum_{i=1}^{4} p(X_i(t+1)|X_i(t), q_i(t), q^*_{i-1}(t)) \tag{7.7}$$

It is illustrated in Figure 7.7:

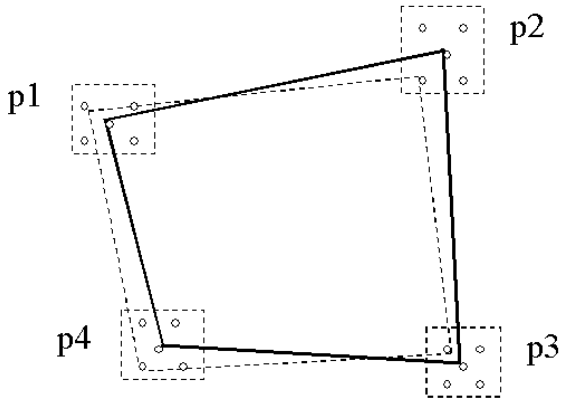


Figure 7.7 Tracking a quadrangle by dynamic programming technique.

In our implementation, the search region for each corner point is approximated by a line segment, instead of a region. This is equivalent to searching for side lines. Corner points are then computed from the intersections of these lines.

Criterion. As mentioned earlier, the appearance of each side line of the quadrangle is modeled by $\mathbf{x}$ that contains both the gradient information and the color information. Maximizing the probability in Equation (7.7) implies finding a pair of line segments between t and $t+1$ such that their appearances are closest. This can be done by minimizing the relative entropy between their distributions. Assume Gaussian distribution of X and Y, then the relative entropy:

$$D(X||Y) = \int p_(u) \lg \frac{p_x(u)}{p_y(u)} du = E[\lg \frac{p_x(u)}{p_y(u)}] \tag{7.8}$$

$$= \frac{d}{2} \lg \frac{|\Sigma_y|}{|\Sigma_x|} - \frac{1}{2} + \frac{1}{2} E[(x-\mu_y)' \Sigma_y^{-1} (x-\mu_y)] \tag{7.9}$$

$$= \frac{d}{2} \lg \frac{|\Sigma_y|}{|\Sigma_x|} - \frac{1}{2} + \frac{|\Sigma_y|}{2|\Sigma_x|} + \frac{1}{2} (\mu_x-\mu_y)' \Sigma_y^{-1} (\mu_x-\mu_y) \tag{7.10}$$

Thus, we have a symmetric distance metric:

$$\begin{aligned} D(X,Y) &= 2(D(X||Y+) + D(Y||X)) && (7.11) \\ &= \frac{|\Sigma_y|}{|\Sigma_x|} + \frac{|\Sigma_x|}{|\Sigma_y|} + (\mu_x - \mu_y)'(\Sigma_x^{-1} + \Sigma_y^{-1})(\mu_x - \mu_y) - 2 && (7.12) \end{aligned}$$

By this means, we can find the best-matched edge at time $t+1$ by

$$e_i^*(t+1) = \arg \min_{\{q_i, q_{i-1}\}} D(X(t), X(t+1) : \{q_i, q_{i-1}\}) \tag{7.13}$$

7.2.3 Tracking a fingertip

The section presents the method of determining the exact tip of a tip pointer by conic fitting for tip pointer tracking, and the method of reinitialization the tracking by a background subtraction technique.

7.2.3.1 Fingertip representation

A tip pointer, such as a fingertip, can be represented by a conic, say,

$$a_1 x^2 + a_2 y^2 + a_3 xy + a_4 x + a_5 y + 1 = 0 \tag{7.14}$$

If we know a set of positions $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ of edge pixels, we can fit a conic on such data by LSE. Explicitly, we have

$$\begin{bmatrix} x_1^2 \; y_1^2 \; x_1 y_1 \; x_1 \; y_1 \\ \vdots \\ x_n^2 \; y_n^2 \; x_n y_n \; x_n \; y_n \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_5 \end{bmatrix} = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} \tag{7.15}$$

Concisely, this equation can be written as $\mathbf{Ma} = \mathbf{b}$. So, the LSE solution of $\mathbf{x}$ is given by

$$\mathbf{a}^* = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{b} \tag{7.16}$$

The conic fitting is be shown in Figure 7.8.

7.2.3.2 Tracking the tip pointer

The tracking of a tip pointer is quite intuitive and cost-efficient. Assume the position of the tip at time t is $p(t)$. The Kalman filtering technique can be employed to predict the tip

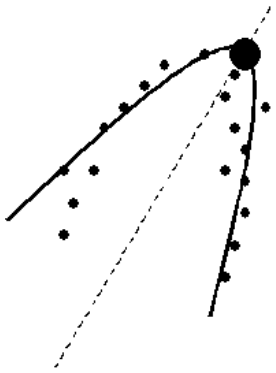


Figure 7.8 Fingertip detection by conic fitting.

position $\bar{p}(t+1)$ at time $t = 1$. In a small window, say 30×30, we identify as many edge pixels as possible that probably belong to the edge of the tip by thresholding the gradient and taking advantage of the color of the previous tracked tip edge. After that, we can fit a conic to these edge pixels and solve the exact tip $p(t+1)$ for time $t+1$.

7.2.3.3 Maintaining background and re-initialization

To make the system more robust and easy-to-use, the scheme of automatic tracking initialization and tracking recovery should be embedded in the system. Here, we developed a means for such a reinitialization task by a dynamic background subtraction technique.

Assume we have already registered the panel at the beginning of the application, i.e., the position of the panel $\mathcal{Q}(0)$ at time 0 is known. Since at time t, the system can track the panel position $\mathcal{Q}(t)$, the homography $H(t)$ between $\mathcal{Q}(0)$ and $\mathcal{Q}(t)$ can be easily calculated. Through the homography $H(t)$, the pixels $P_t(0)$ in the panel at time 0 are mapped to the panel at time t as $P_b(t)$ by

$$P_b(t) = H(t)P_t(0)$$

Here, $P_b(t)$ is the background of time t. Obviously, the foreground $P_f(t)$ at time t can be calculated by subtracting the bacground $P_b(t)$ from current image $P(t)$, i.e.,

$$P_f(t) = P(t) - P_b(t) = P(t) - H(t)P_t(0) \tag{7.17}$$

Figure 7.9 shows the basic idea of our approach.

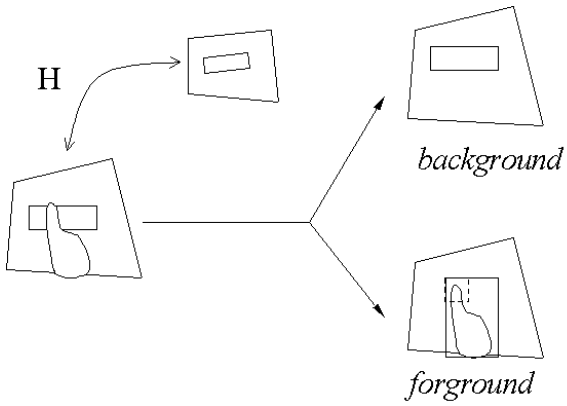


Figure 7.9 The foreground, i.e., the hand, can be segmented out from the background, since the current position of the panel is tracked and a background template is maintained.

7.2.4 Action detection and recognition

This section presents the techniques of action detection and recognition. The virtual panel system has two clicking modes (clicking mode and dragging mode), and two mouse types (absolute type and relative type).

7.2.4.1 Two mouse clicking modes

To enable our system clicking and dragging, we have two modes in virtual mouse: mode I (clicking mode) and mode II (dragging mode), as shown in Figure 7.10. In our current implementation, clicking/pressing is simulated by holding the tip pointer for a while, say 1 second.

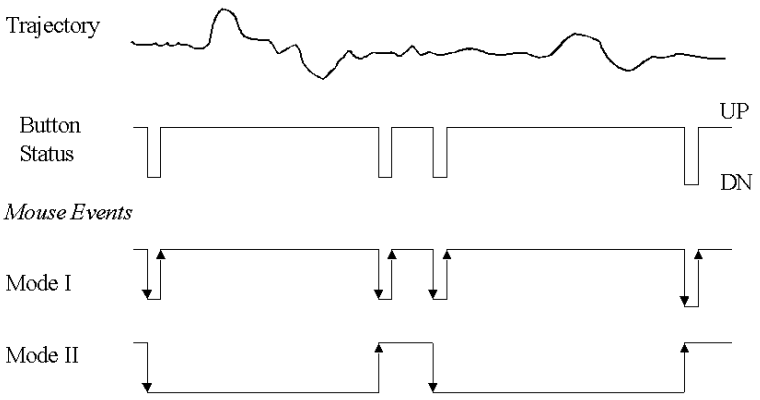


Figure 7.10 Simulating clicking (mode I) and dragging (mode II).

A state variable S maintains two states, UP and DN, to simulate the two natural state of a button. At the beginning, the variable S is UP. For the clicking mode (mode I), when the system detects that the tip pointer has been in a fixed place for 1 second (or other amount of time prespecified), the state variable S is set to UP; otherwise, S keeps UP. After 0.1 second, the state variable S will be automatically set to UP to simulate button release. When the state change from UP to DN is detected, a clicking action is detected.

Obviously, the clicking mode (mode I) has very limited ability to drag, since the release is automatic. To simulate dragging, the dragging mode (mode II) uses another state variable D to memorize the flip of clicking. If the previous D-state is D_UP, then the current click is D_DN; otherwise, the current click is D_UP. When the D-state change from D_UP to D_DN is detected, a pressing action is detected; when the D-state change from D_DN to D_UP is detected, a releasing action is detected. By this means, we facilitate the system the ability of dragging.

7.2.4.2 Two mouse motion types

The virtual panel system can simulate two mouse types: absolute type and relative type. In the absolute type, the panel will be mapped to the whole remote display, such that each point in the panel will be mapped to the corresponding point in the display. As we discussed before, this type needs very accurate tracking, since a small tracking error of the panel and tip pointer will be magnified. However, the absolute type is more intuitive.

The other type is the relative type, which will be much less sensitive to the tracking results, since the cursor is controlled by the relative motion of the tip pointer. Assume the motion direction of tip pointer is $\mathbf{d}_p(t)$ at time t. The moving direction of the cursor will be $\mathbf{d}_d(t) = H(t)\mathbf{d}_p(t)$. The amount movement of the cursor will be determined by the velocity of the tip pointer, i.e, $\Delta_d = \alpha||\mathbf{v}_p||$.

There could be many other alternatives of relative mouse movements. For an instance, the panel can be simply mapped to a window area centered at the previous cursor position on the remote display. In this method, the center of the panel corresponds to the previous cursor position. When the tip pointer moves from center to left, the cursor will move left. Obviously, the window area could be smaller than the panel in the image, such that the tracking error can even be minimized.

The relative type brings much smooth movement of the cursor due to the nonmagnification of the tracking error. However, compared to the absolute type, the relative type is less intuitive.

7.2.5 Demos

Based on the virtual panel system, several applications are made to demonstrate the capacity of the system. One of the applications shows that a remote display can be freely and accurately controlled. One can use his finger to draw a picture in another demo. Even more, one can input text without using any keyboards in the third demo.

7.2.5.1 Calculator controlling

This application demonstrates the accuracy and stability of the virtual panel system. The calculator, with around 30 buttons, takes a very small part area of the display. In this demo, the user can freely use his fingertip to click any buttons or menus of the calculator, as shown in Figure 7.11. The tracking error is less than 1 pixel, and the motion of the cursor is very smooth.

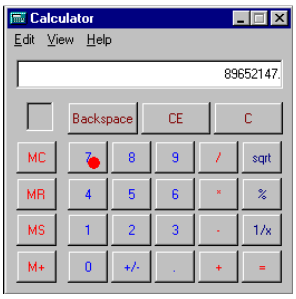


Figure 7.11 Controlling a calculator.

7.2.5.2 Finger painting

This application demonstrates different clicking modes. In Paint, a Windows application, user can use his finger to select any tools and draw anything (see Figure 7.12). Our usability study shows that users are quite adaptive to this system. After several minutes of training, all users can freely use their fingers to draw a picture and control the remote display.

Figure 7.12 Finger painting.

7.2.5.3 Virtual keyboard

This application demonstrates that the physical keyboard can be replaced by a printed virtual keyboard in the virtual panel system. We print a keyboard pattern on the panel, which is shown in Figure 7.13. When the user points to any of the keys on the panel, a key-down message will be sent to the operating system, such that the current active application will receive that key. For example, we can use Notepad to receive text inputted by the user.

Figure 7.13 Virtual keyboard.

In our current implementation, users only use one of their fingers. The typing speed is very slow. We are not claiming that we can get rid of the physical keyboard. However, our system could be a very promising alternative when the physical keyboard is not available under some circumstances.

7.2.6 Extensions and future work

We developed a prototype vision-based gesture interface system, virtual panel, which is capable of performing accurate control of remote display and simulating mouse and keyboard input. The virtual panel system employs an arbitrary quadrangle-shaped planar object as a panel, which can be viewed as a display or a keyboard. Users can use their fingers or other

tip pointers to simulate a cursor pointer and issue clicking/pressing instructions. The system can robustly and accurately track the panel and the tip pointer. After the action is detected, various events can be generated and sent to the operating system. Such vision-based gesture interface can achieve more natural and intuitive interaction between humans and computers. Three applications have been described: controlling a calculator, painting with fingers, and inputting text with a virtual keyboard.

The virtual panel system leaves a lot of room for extensions and improvements in various aspects, especially in action recognition. In our current implementation, action is triggered when the tip point stays immobile for a short duration. We are investigating more natural ways to do that, for example, by combining hand gesture recognition.

7.3 More Potential Applications

Many potential application systems can be developed given the capacity of our gesture interface.

- *Navigating 3-D VEs*: Since the hand can be robustly and efficiently tracked, the cursor pointer can be replaced by the natural hand. Some applications, such as map navigation and BattleField navigation, can be easily developed. Since our color-based hand localization algorithm can be easily extended to locate two hands simultaneously, and to 3-D localization by two cameras, a full 6 DOF pose parameters can be obtained by two hands because the 3-D position of two hands determines a 3-D line, which in turn determines a plane. Such technique is being developed in the EVL lab of UI-Chicago.

- *Switching commanding modes*: In many applications, different commanding modes are needed. For example, the two modes of navigating and selecting should be differentiated. Previously, such switching is achieved by speech recognition. Our gesture interface offers an alternative way of switching by recognizing different hand postures. Combining this visual recognition and speech recognition would largely enhance the robustness of command switching.

- *Controlling displays*: When a speaker makes a PowerPoint presentation in a large conference room, it would be inconvenient to control the slides by clicking a mouse. It would

be better if users could use their hands to do this. In fact, our gesture interface can be integrated into PowerPoint presentations.

- *Manipulating virtual objects*: If the articulated motion algorithm can be efficiently implemented and run on nearly real-time, we can use the hand as a high DOF input tool. Our research supplies an initial study in this direction.

CHAPTER 8

CONCLUSION AND FUTURE RESEARCH

The role and the functionality of computers have been changing since the boom in computer hardware technology. The computer is no longer just a machine for scientific computation, no longer just a machine for saving labor for trivial daily routines. It is an indispensable part of the information age for information acquisition, storage, analysis, organization, and distribution. It requires different levels of intelligence. Cutting-edge speech technology has achieved a big step toward making computers less dumb and deaf. To some extent, computers are now able to "listen" and "speak." Meanwhile, we also need to make computers "see" and "think."

Specifically, to have a more natural and more immersive interaction with human beings, computers need to be able to visually perceive and understand human activities. For example, instead of relying on devices such as a mouse, computers could understand the meanings conveyed through human body movements or gestures. However, we are still far from this goal. The main challenges lie in the richness of visual inputs and the large variations of human movements. To achieve immersive and intelligent human-computer interaction, we need to investigate two problems: visual motion capturing and visual learning.

8.1 Summary

This dissertation addressed several aspects of these two difficult problems by taking the human hand and hand gestures as a case study, because the hand is an important part of human communication. Three important problems regarding visual motion capturing were presented in this dissertation: nonstationary color tracking in Chapter 3, integration of multiple cues for robust tracking in Chapter 4, and capturing hand articulation in Chapter 5. The techniques

developed in those chapters could be easily extended to many other visual motion capturing tasks.

It is noticeable that many learning problems in vision-based interaction share a common difficulty, i.e., the lack of supervised information. These learning problems range from invariant object recognition to color model adaptation, from feature selection to sensor fusion. Chapter 6 proposed a study of a new learning paradigm, *self-supervised learning*, which employs both supervised and unsupervised training data sets. Many visual learning problems could be unified into this learning paradigm.

According to the source of labeled data set L and unlabeled data set U, i.e., L and U from the same or different pdf, and feature modalities, i.e., unimodal or multimodal, the self-supervised learning has four typical learning problems: *transduction*, *co-transduction*, *model transduction* and *co-inferencing*, as shown in Table 8.1.

Table 8.1 Four typical problems in self-supervised learning.

L/U source	From the same pdf	From different pdf
Unimodal	Transduction	Model transduction
Multimodal	Co-transduction	Co-inferencing

These four typical learning problems could be illustrated in Figure 8.1(a)-(d), respectively. M_t^j means the learning model of the jth modality at time t. L_t and U_t represent labeled and unlabeled data at time t.

The transduction problem is the fundamental learning task in self-supervised learning. It assumes that both labeled and unlabeled training data L and U are drawn from the same distribution. The task is to learn a classifier based on both L and U. The significance of investigating transduction is that it could save the tedious work of labeling a huge training data set. When given a partially labeled training set, we expect the transduction algorithm to propagate the labeling to those unlabeled data. Integrating discriminant analysis and the EM algorithm, the D-EM algorithm in Chapter 6 combines supervised and unsupervised learning. An example is view-independent hand posture recognition that requires the recognition of specific hand postures from different view directions. Instead of collecting the images covering nearly all possible viewpoints, we may only need to label a fraction of images under several

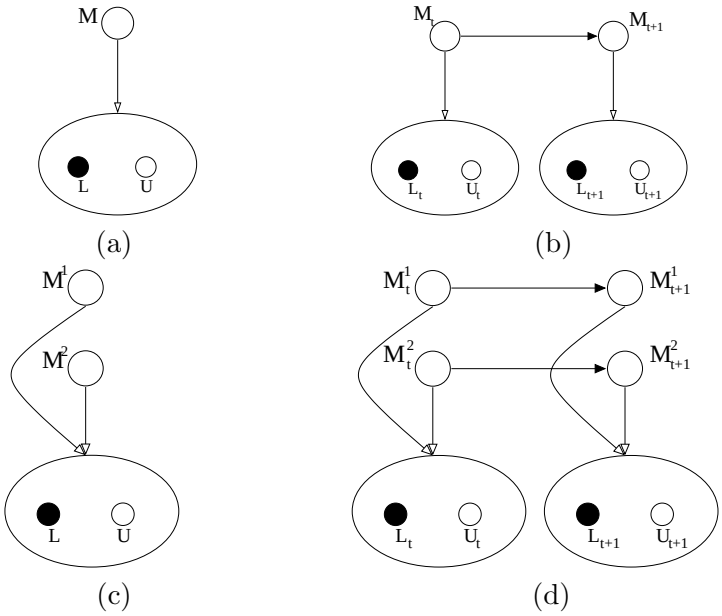


Figure 8.1 Illustration of the four typical problems of self-supervised learning: (a) transduction, (b) model transduction, (c) co-transduction, and (d) co-inferencing.

typical view directions. Another good example is content-based image retrieval (CBIR). These two problems have been studied in Chapter 6.

The model transduction problem does not assume L and U come from the same distribution, but assume a known relationship between these two distributions, which introduces "dynamics" into self-supervised learning. The learning task is to transduce an old model to a new one. Many tasks in vision-based interaction could be represented by model transduction. It is easy to see that an automatic adaptation of face or head models is a good example. An even more interesting problem is the color model adaptation in nonstationary environments. Chapter 3 investigated color model transduction for non-stationary color tracking.

The *co-transduction* problem takes multimodal training data and assumes L and U from the same distribution. The significance of this problem is that it could fuse multiple modalities and reduce the computational complexity. A simple example comes from the CBIR problem. Since it is difficult to represent a concept, a large number of features should be fed to the learner. When considering different modalities such as color, texture, and structure layout of images, the retrieval problem could become easier. Although this dissertation did not study *co-transduction*, many ideas in the other three could be easily extended to this learning problem.

The *co-inferencing* problem is the most complicated of these four problems. It takes multimodal training data and assumes L and U from different distributions. It is a combination of model transduction and co-transduction. The learning task is to transduce an old model to a new model based on partially labeled multimodal training data. Chapter 4 investigated this problem by taking the example of integrating multiple cues for robust visual tracking. The most important conclusion is that different modalities present a co-inferencing phenomenon, i.e., the model of anyone of the modalities is learned from all the other modalities iteratively. Mathematically, co-inferencing could be described by a set of fixed point equations. Chapter 4 also gives an implementation of co-inferencing for robust tracking based on sequential Monte Carlo techniques.

This dissertation also presented two interesting prototypes of vision-based gesture interface. One of them allows users to play a simple interactive video game against the computer, and the other allows users to control a remote display by just using their fingers and a piece of paper.

8.2 Future Research Directions

The research of combining labeled and unlabeled training data is still in its infancy. More research effort should be made in this area, not only for the theoretical foundation, but also for more realistic applications. Many interesting and important issues in self-supervised learning should be investigated further.

In Chapter 6, we proposed a novel learning algorithm named the D-EM algorithm. We notice that there is lack of analysis of this algorithm. We should study its convergence property further. In our preliminary experiments, we found that D-EM may not converge, and one of the classes may dominate the classification after several E-D-M iterations. It seems that the labeled data affects the convergence. The necessary condition of convergence should be obtained in our future study. It is also good to study the convergence rate. We should also study the stability of D-EM in the future.

The role of unlabeled data in self-supervised learning is still unclear. We only give a very intuitive explanation. When the unlabeled set helps an hurts is still a mystery. Although self-supervised learning captures some human cognition processes to some extent, the human

cognition model is still unknown, not only for psychology, but also for AI. We expect the model of self-supervised transduction to be a strong tool for active learning and incremental learning.

The investigation of self-supervised learning in this dissertation mainly concentrates on classification problems. It would be very useful to extend our approach to regression problems. In fact, we have many such regression problems in vision applications. Head pose estimation is a very good example, in which we need to regress 3-D head orientations given head observation and a 3-D head model. It is very expensive and time-consuming to collect a large annotated training data set for this problem. We should consider using fewer annotated samples combined with a large set of un-annotated samples, to make a rough 3-D head model [77, 146].

In the proposed vision-based gesture interfaces, we do not focus on temporal gesture modeling and recognition. One of the reasons for this is that we concentrate on hand and fingers, rather than the hand global motion. Generally, temporal gestures do not make much sense in hand and finger motion, since the most meaningful gestures can be represented by a large number of hand postures. However, temporal gestures are useful in many other interesting applications.

Finally, an important issue in recognition of temporal gestures is motion modeling. The motion models can be learned from training samples. Different modeling approaches, such as hidden Markov model (HMM), dynamic time warping (DTW), finite state machine (FSM), extended Kalman filtering (EKF), dynamic Bayesian networks (DBN) and dynamic self-organizing map (DSOM), should be investigated in the future. Basically, self-supervised learning techniques can also be applied to this problem, since the motion modeling problem is essentially a regression problem.

The techniques developed in this dissertation are general enough to be extended to many other research topics. The author hopes that this study could motive more and more investigation of self-supervised learning and serve as an important step toward realizing the dream of making computers "see" and "think."

APPENDIX A

DERIVATION OF FIXED POINT EQUATIONS

This appendix presents some details for the variational analysis of the factorized graphical model and the derivation of the fixed point equation described in Chapter 4. Our purpose is to illustrate the so-called *co-inference* phenomenon that presents in the interaction of different modalities. In the visual tracking scenario, the state variables are continuous, which makes it very difficult to analyze the graphical model. Here, the investigation of the case that takes discrete states is described.

To simplify the analysis, we assume the state $\mathbf{X}_t^m$ a multinomial random variable which takes one of K discrete states, i.e., $\mathbf{X}_t^m \in \{1, \ldots, K\}$. Here, we represent this random variable by a $K \times 1$ vector. Only one element of the vector will be 1 while others are 0, which means that the state variable is in one of these discrete states.

$$\mathbf{X}_t^m = \begin{pmatrix} x_{t,1}^m \\ \vdots \\ x_{t,K}^m \end{pmatrix}$$

Such a setting could be used to approximate a continuous case in the tracking scenario. The system dynamics will be approximated by a $K \times K$ transition matrix.

$$P(\underline{X}_t | \underline{Z}_t, \phi) = \frac{1}{Z} exp\{-H(\underline{X}_t, \underline{Z}_t)\} \tag{A.1}$$

where Z is a constant to normalize the probabilities, and where

$$\begin{aligned} H(\underline{X}_t, \underline{Z}_t) &= \frac{1}{2} \sum_{t=1}^{T} (\mathbf{Z}_t - \sum_{m=1}^{M} W^m \mathbf{X}_t^m)' C^{-1} (\mathbf{Z}_t - \sum_{m=1}^{M} W^m \mathbf{X}_t^m) \\ &\quad - \sum_{m=1}^{M} \mathbf{X}_1^{m\prime} \log \pi^m - \sum_{t=2}^{T} \sum_{m=1}^{M} \mathbf{X}_t^{m\prime} (\log P^m) \mathbf{X}_{t-1}^m \end{aligned}$$

where W^m is an observation matrix and C is the covariance matrix, since here we also assume a linear observation model to ease the derivation. Such a linear observation assumption may not be true for visual tracking, but we take it as an approximation to ease the analysis.

On the other hand, we would write the inference probability of the graphical model under structured variational approximation:

$$Q(\mathbf{X}_1^m|\theta) = \prod_{k=1}^{K}(h_{1,k}^m \pi_k^m)^{x_{1,k}^m}$$

$$\begin{aligned} Q(\mathbf{X}_t^m|\mathbf{X}_{t-1}^m,\theta) &= \prod_{k=1}^{K}\left(h_{t,k}^m \sum_{j=1}^{K} P_{kj}^m \mathbf{X}_{t-1,j}^m\right)^{x_{t,k}^m} && \text{(A.2)} \\ &= \prod_{k=1}^{K}\left(h_{t,k}^m \sum_{j=1}^{K} (P_{kj}^m)^{x_{t-1,j}^m}\right)^{x_{t,k}^m} && \text{(A.3)} \end{aligned}$$

And similarly, we have

$$Q(\underline{X}_t|\theta) = \frac{1}{Z_Q} exp\{-H_Q(\underline{X}_t)\}$$

where

$$H_Q(\underline{X}_t) = -\sum_{m=1}^{M} \mathbf{X}_1^{m\prime} \log \pi^m - \sum_{t=2}^{T}\sum_{m=1}^{M} \mathbf{X}_t^{m\prime} (\log P^m) \mathbf{X}_{t-1}^m - \sum_{t=1}^{T}\sum_{m=1}^{M} \mathbf{X}_t^{m\prime} \log h_t^m$$

Thus, the KL divergence can be written as

$$\begin{aligned} KL(Q||P) &= \langle H \rangle - \langle H_Q \rangle - \log Z_Q + \log Z \\ &= \sum_{t=1}^{T}\sum_{m=1}^{M} \langle \mathbf{X}_t^m \rangle \log h_t^m + \frac{1}{2}\sum_{t=1}^{T}\Bigg[Z_t' C^{-1} Z_t - 2\sum_{m=1}^{M} \mathbf{Z}_t' C^{-1} W^m \langle \mathbf{X}_t^m \rangle \\ &\quad + \sum_{m=1}^{M}\sum_{n\neq m}^{M} tr\left\{ W^{m\prime} C^{-1} W^n \langle \mathbf{X}_t^n \rangle \langle \mathbf{X}_t^{m\prime} \rangle \right\} \\ &\quad + \sum_{m=1}^{M} tr\left\{ W^{m\prime} C^{-1} W^m diag\langle \mathbf{X}_t^m \rangle \right\} \Bigg] - \log Z_Q + \log Z \end{aligned}$$

Then, to minimize the KL divergence of such two distributions, we can take the derivative with respect to $\log(h_t^n)$:

$$\begin{aligned} \frac{\partial KL(Q||P)}{\partial \log h_t^n} &= \langle \mathbf{X}_t^n \rangle + \sum_{t=1}^{T}\sum_{m=1}^{M} \Bigg\{ \log h_t^m - W^{m\prime} C^{-1} \mathbf{Z}_t + \sum_{k\neq m}^{M} W^{m\prime} C^{-1} W^k \langle \mathbf{X}_t^k \rangle \\ &\quad + \frac{1}{2}\Delta^m \Bigg\} \frac{\partial \langle \mathbf{X}_t^m \rangle}{\partial \log h_t^n} - \langle \mathbf{X}_t^n \rangle \end{aligned}$$

Setting the derivatives to zeros, we have

$$\log h_t^m - W^{m\prime} C^{-1} \mathbf{Z}_t + \sum_{k \neq m}^{M} W^{m\prime} C^{-1} W^k \langle \mathbf{X}_t^k \rangle + \frac{1}{2} \Delta^m = 0 \tag{A.4}$$

Therefore, we end up with a set of fixed point equations:

$$\widetilde{h_t^m} = exp \left\{ W^{m\prime} C^{-1} \left[\mathbf{Z}_t - \sum_{k \neq m}^{M} W^k \langle \mathbf{X}_t^k \rangle \right] - \frac{1}{2} \Delta^m \right\}$$

Since $\langle \mathbf{X}_t^m \rangle = E\left[\mathbf{X}_t^m | \theta, \underline{Z}_t\right]$, so, the set of fixed point equations can be written as:

$$\widetilde{h_t^m} = exp \left\{ W^{m\prime} C^{-1} \left[\mathbf{Z}_t - \sum_{k \neq m}^{M} W^k E\left[\mathbf{X}_t^m | \theta, \underline{Z}_t\right] \right] - \frac{1}{2} \Delta^m \right\} \tag{A.5}$$

To see them clearly, we could represent this set of fixed point equation by

$$\widetilde{h_t^m} = g(\mathbf{Z}_t, \{E[\mathbf{X}_t^n | \underline{Z}_t^n, h^n] : \forall n \neq m\}) \tag{A.6}$$

where $g(\cdot, \cdot)$ is a function. Obviously, this equation is the same as Equation (4.3) in Section 4.3.

APPENDIX B

SOLVING ROTATION MATRIX AND DEPTH

This appendix describes the details of solving the rotation matrix $\mathbf{R}$ and the depth $\mathbf{t}_3$ from the matrix $\mathbf{B}$. After we solve the motion based on the factorization method, $\mathbf{M}$ contains only $\mathbf{B}$ if we assume orthographic projection. In the following, we show how to estimate the rotation $\mathbf{R}$ and the depth translation t_3 from $\mathbf{B}$ under scaled orthographic projection. It is clear that

$$\mathbf{R} = \begin{bmatrix} t_3 B_{11} & t_3 B_{12} & R_{13} \\ t_3 B_{21} & t_3 B_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

From the property $\mathbf{R}^T\mathbf{R} = \mathbf{I}$, we obtain

$$t_3^2 b_1^2 + R_{31}^2 = 1 \quad \text{with } b_1^2 = B_{11}^2 + B_{21}^2 \tag{B.1}$$

$$t_3^2 b_2^2 + R_{32}^2 = 1 \quad \text{with } b_2^2 = B_{12}^2 + B_{22}^2 \tag{B.2}$$

$$t_3^2 d + R_{31} R_{32} = 0 \quad \text{with } d = B_{11}B_{12} + B_{21}B_{22} \tag{B.3}$$

From Equation B.1, we have

$$t_3^2 = \frac{1}{b_1^2}(1 - R_{31}^2) \tag{B.4}$$

Then, from Equation B.3, we obtain

$$R_{32} = \frac{d}{b_1^2} R_{31} - \frac{d}{b_1^2} \frac{1}{R_{31}} \tag{B.5}$$

Substitute them into Equation B.2, and we get

$$(b_1^2 b_2^2 - d^2) R_{31}^4 + (2d^2 - b_1^2 b_2^2 + b_1^4) R_{31}^2 - d^2 = 0 \tag{B.6}$$

We may have at most four solutions, but usually only two because R_{31}^2 should be always positive. Even the two solutions can be possibly resolved by checking whether the resulting matrix is a rotation matrix or a reflection (the determinant is equal to -1).

REFERENCES

[1] J. Aggarwal and Q. Cai, "Human motion analysis: A review," in *Proc. of IEEE Nonrigid and Articulated Motion Workshop*, 1997, pp. 90–102.

[2] R. Kjeldsen and J. Kender, "Finding skin in color images," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 312–317.

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

[4] Y. Wu and T. S. Huang, "Human hand modeling, analysis and animation in the context of HCI," in *Proc. IEEE Int'l Conf. on Image Processing*, vol. 3, Oct. 1999, pp. 6–10.

[5] 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.

[6] 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.

[7] D. M. Gavrila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding*, vol. 73, pp. 82–98, Jan. 1999.

[8] Q. Liu, S. Levinson, Y. Wu, and T. S. Huang, "Interactive and incremental learning via a mixture of supervised and unsupervised learning strategies," in *Proc. Joint Conf. on Information Systems*, Feb. 2000, pp. 555–558.

[9] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: A power tool for interactive content-based image retrieval," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8, pp. 644–655, 1998.

[10] S. Santini and R. Jain, "Similarity measures," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 871–883, 1999.

[11] D. Swets and J. Weng, "Hierarchical discriminant analysis for image retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 386–400, 1999.

[12] W. Stokoe, *Sign Language Structure*. Buffalo, NY: University of Buffalo Press, 1960.

[13] A. Kendon, "Current issues in the study of gesture," in *The Biological Foundation of Gestures: Motor and Semiotic Aspects*, J. Nespoulous, P. Perron, and A. Lecours, Eds., Hillsdale, NJ: Lawrence Erlbaum Associate, 1986, pp. 23–47.

[14] F. Quek, "Unencumbered gesture interaction," *IEEE Multimedia*, vol. 3, pp. 36–47, 1997.

[15] A. Bobick and Y. Ivanov, "Action recognition using probabilistic parsing," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 196–202.

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

[17] J. J. Kuch and T. S. Huang, "Vision-based hand modeling and tracking for virtual teleconferencing and telecollaboration," in *Proc. of IEEE Int'l Conf. on Computer Vision*, June 1995, pp. 666–671.

[18] 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 3rd Conf. on Face and Gesture Recognition*, 1998, pp. 268–273.

[19] T. Heap and D. Hogg, "Towards 3D hand tracking using a deformable model," in *Proc. of IEEE Int'l Conf. Automatic Face and Gesture Recognition*, 1996, pp. 140–145.

[20] C. Vogler and D. Metaxas, "ASL recognition based on a coupling between HMMs and 3D motion analysis," in *Proc. of IEEE Int'l Conf. on Computer Vision*, Jan. 1998, pp. 363–369.

[21] L. Tsap, D. Goldgof, and S. Sarkar, "Human skin and hand motion analysis from range image sequences using nonlinear FEM," in *Proc. IEEE Nonrigid and Articulated Motion Workshop*, 1997, pp. 80–88.

[22] J. Rehg and T. Kanade, "Model-based tracking of self-occluding articulated objects," in *Proc. of IEEE Int'l Conf. Computer Vision*, 1995, pp. 612–617.

[23] Y. Wu and T. S. Huang, "Capturing articulated human hand motion: A divide-and-conquer approach," in *Proc. IEEE Int'l Conf. on Computer Vision*, Sept. 1999, pp. 606–611.

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

[25] J. Lin, Y. Wu, and T. S. Huang, "Modeling human hand constraints," in *Proc. of Workshop on Human Motion*, Dec. 2000, pp. 121–126.

[26] J. Davis and A. Bobick, "The representation and recognition of action using temporal templates," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1997, pp. 928–934.

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

[28] A. Wilson and A. Bobick, "Recognition and interpretation of parametric gesture," in *Proc. of IEEE Int'l Conf. on Computer Vision*, 1998, pp. 329–336.

[29] V. Pavlović, "Dynamic bayesian networks for information fusion with applications to human–computer interfaces," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.

[30] M. Jones and J. Rehg, "Statistical color models with application to skin detection," Compaq Cambridge Research Lab., Cambridge, MA, Tech. Rep. CRL-98-11, 1998.

[31] 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. of Asian Conference on Computer Vision*, Jan. 2000, pp. 1106–1111.

[32] Y. Raja, S. McKenna, and S. Gong, "Colour model selection and adaptation in dynamic scenes," in *Proc. of European Conf. on Computer Vision*, 1998, pp. 460–475.

[33] C. Wren, A. Azarbayejani, T. Darrel, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 780–785, July 1997.

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

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

[36] M. Isard and A. Blake, "CONDENSATION — conditional density propagation for visual tracking," *Int'l Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[37] Y. Wu, J. Lin, and T. S. Huang, "Capturing natural hand articulation," in *Proc. IEEE Int'l Conference on Computer Vision*, vol. II, July 2001, pp. 426–432.

[38] Y. Cui, D. Swets, and J. Weng, "Learning-based hand sign recognition using SHOSLF-M," in *Proc. of Int'l Workshop on Automatic Face and Gesture Recognition*, 1995, pp. 201–206.

[39] C. R. Wren and A. Pentland, "Dynamic modeling of human motion," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1995, pp. 22–27.

[40] L. Campbell, D. Becker, A. Azarbayejani, A. Bobick, and A. Pentland, "Invariant features for 3-D gesture recognition," in *Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 157–162.

[41] D. Becker, "Sensei: A real-time recognition, feedback and training system for T'ai Chi gestures," M.S. thesis, Media Lab, MIT, Cambridge, MA, 1997.

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

[43] Y. Wu and T. S. Huang, "Self-Supervised learning for visual tracking and recognition of human hand," in *Proc. AAAI National Conf. on Artificial Intelligence*, July 2000, pp. 243–248.

[44] J. J. Kuch, "Vision-based hand modeling and gesture recognition for human computer interaction," M.S. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1994.

[45] C. Nolker and H. Ritter, "Illumination independent recognition of deictic arm posture," in *Proc. 24th Annual Conf. of the IEEE Industrial Electronics Society*, 1998, pp. 2006–2011.

[46] Y. Cui and J. Weng, "Hand segmentation using learning-based prediction and verification for hand sign recognition," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1996, pp. 88–93.

[47] F. Quek and M. Zhao, "Inductive learning in hand pose recognition," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 78–83.

[48] R. Rosales and S. Sclaroff, "Inferring body pose without tracking body parts," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 721–727.

[49] J. Triesch and C. von de Malsburg, "Robust classification of hand postures against complex background," in *Proc. Int'l Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 170–175.

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

[51] M. Black and A. Jepson, "Recognition temporal trajectories using the Condensation algorithm," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1998, pp. 16–21.

[52] A. Pentland and A. Liu, "Modeling and prediction of human behavior," in *Proc. of IEEE Intelligent Vehicles*, Sept. 1995, pp. 350–355.

[53] J. Rittscher and A. Blake, "Classification of human body motion," in *Proc. of IEEE Int'l Conf. on Computer Vision*, 1999, pp. 643–639.

[54] C. Cohen, L. Conway, and D. Koditschek, "Dynamical system representation, generation, and recognition of basic oscillation motion gestures," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 60–65.

[55] K. Jo, Y. Kuno, and Y. Shirai, "Manipulative hand gestures recognition using task knowledge for human computer interaction," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1998, pp. 468–473.

[56] G. Berry, "Small-wall: A multimodal human computer intelligent interaction test bed with applications," M.S. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1998.

[57] R. Cutler and M. Turk, "View-based interpretation of real-time optical flow for gesture recognition," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1998, pp. 416–421.

[58] C. Pinhanez and A. Bobick, "Human action detection using PNF propagation of temporal constraints," in *Proc. of IEEE Int'l Conf. on Computer Vision*, 1998, pp. 898–904.

[59] C. Bregler, "Learning and recognition human dynamics in video sequences," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1997, pp. 568–574.

[60] P. Stoll and J. Ohya, "Application of HMM modeling to recognizing human gestures in image sequences for a man-machine interface," in *Proc. of IEEE Int'l Workshop on Robot and Human Communication*, 1995, pp. 129–134.

[61] Y. Nam and K. Wohn, "Recognition of space-time hand-gestures using hidden Markov model," in *Proc. of ACM Symposium on Virtual Reality Software and Technology*, 1996, pp. 51–58.

[62] J. Yang, Y. Xu, and C. Chen, "Gesture interface: Modeling and learning," in *Proc. of IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 1994, pp. 1747–1752.

[63] T. Kobayashi and S. Hayamizu, "Partly-hidden Markov model and its application to gesture recognition," in *Proc. of IEEE ICASSP*, vol. VI, 1997, pp. 3081–3084.

[64] T. Darrell and A. Pentland, "Active gesture recognition using partially observable Markov decision processes," in *Proc. of IEEE Int'l Conf. on Pattern Recognition*, vol. 3, 1996, pp. 984–988.

[65] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1997, pp. 994–999.

[66] Y. Cui and J. Weng, "Hand sign recognition from intensity image sequences with complex background," in *Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 259–264.

[67] R. Polana and R. Nelson, "Low level recognition of human motion," in *Proc. of IEEE Workshop on Motion of Non-rigid and Articulated Object*, 1994, pp. 77–82.

[68] T. Watanabe and M. Yachida, "Real time gesture recognition using eigenspace from multi input image sequences," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1998, pp. 428–433.

[69] M. Yang and N. Ahuja, "Extraction and classification of visual motion patterns for hand gesture recognition," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 892–897.

[70] C. Vogler and D. Metaxas, "Toward scalability in ASL recognition: Breaking down signs into phonemes," 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.

[71] R. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1998, pp. 558–567.

[72] Y. Wu and T. S. Huang, "Color tracking by transductive learning," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, June 2000, pp. 133–138.

[73] F. Quek, D. McNeill, R. Bryll, C. Kirbas, H. Arslan, K. McCullough, N. Furuyama, and R. Ansari, "Gesture, speech, and gaze cues for discourse segmentation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. II, 2000, pp. 247–254.

[74] M. Isard and A. Blake, "ICondensation: Unifying low-level and high-level tracking in a stochastic framework," in *Proc. of European Conf. on Computer Vision*, vol. 1, 1998, pp. 767–781.

[75] Y. Wu and T. S. Huang, "Robust visual tracking by co-inference learning," in *Proc. IEEE Int'l Conference on Computer Vision*, vol. II, July 2001, pp. 26–33.

[76] K. Imagawa, S. Lu, and S. Igi, "Color-Based hands tracking system for sign language recognition," in *Proc. of Int'l Conf. on Face and Gesture Recognition*, 1998, pp. 462–467.

[77] K. Toyama and Y. Wu, "Bootstrap initialization of nonparametric texture models for tracking," in *Proc. of European Conf. on Computer Vision*, 2000.

[78] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[79] Y. Wu, "Image compression based on structural adaptation self-organizing VQ algorithm," M.S. thesis, Tsinghua University, Beijing, China, 1997.

[80] Y. Wu, B. Li, and P. Yan, "Nonparametric density estimation using wavelet transformation and scale-space zero-crossing reconstruction," in *Proc.IEEE Int'l Conf. on Signal Proceesing*, vol. I, October 1996, pp. 319–322.

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

[82] M. Cottrell and J. C. Fort, "A stochastic model of retinotopy: A self-organizing process," *Biological Cybernetics*, vol. 53, pp. 405–411, 1986.

[83] F. M. Muiler, "Statistical analysis of self-organization," *Neural Networks*, vol. 8, pp. 717–727, 1995.

[84] H. Ritter and K. Schulten, "On the stationary state of Kohonen's self-organizing sensory mapping," *Biological Cybernetics*, vol. 54, pp. 99–106, 1986.

[85] Y. Zheng and J. Greenleaf, "The effect of concave and convex weight adjustment in self-organizing maps," *IEEE Trans. on Neural Networks*, vol. 7, pp. 1458–171, 1996.

[86] D. Choi and S. Park, "Self-creating and organizing neural networks," *IEEE Trans. on Neural Networks*, vol. 5, pp. 1072–1085, 1994.

[87] B. Fritzke, "Growing cell structures – a self-organzing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, pp. 1441–1460, 1994.

[88] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds., 1995, pp. 625–632.

[89] T.-C. Lee, *Structure Level Adaptation for Artificial Neural Networks*. Boston, MA: Kluwer Academic Publishers, 1991.

[90] Y. Wu, Q. Liu, and T. S. Huang, "Robust real-time human hand localization by self-organizing color segmentation," in *Proc. ICCV'99 Workshop on Recognition, Analysis and Tracking of Face and Gestures in Real-Time Systems*, Sept. 1999, pp. 161–166.

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

[92] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. II, 2000, pp. 142–149.

[93] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. IEEE Int'l Conf. on Computer Vision*, 1999, pp. 255–261.

[94] S. Birchfield, "Ellitical head tracking using intensity gradient and color histograms," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 232–237.

[95] C. Rasmussen and G. Hager, "Joint probabilistic techniques for tracking multi-part objects," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 16–21.

[96] G. Hager and P. Belhumeur, "Real-time tracking of image regions with changes in geoemtry and illumination," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1996, pp. 403–410.

[97] B. Li and R. Chellapa, "Simultaneous tracking and verification via sequential posterior estimation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. II, 2000, pp. 110–117.

[98] H. Tao, H. Sawhney, and R. Kumar, "Dynamic layer representation with applications to tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 134–141.

[99] M. Black and A. Jepson, "Eigentracking: Robust matching and tracking of articulated object using a view-based representation," in *Proc. European Conf. Computer Vision*, vol. 1, 1996, pp. 343–356.

[100] K. Toyama and G. Hager, "Incremental focus of attention for robust visual tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1996, pp. 189–195.

[101] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[102] Z. Ghahramani, "Factorial learning and the EM algorithm," in *Advanced in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds., 1995, pp. 617–624.

[103] Z. Ghahramani and M. Jordan, "Factorial hidden Markov models," *Machine Learning*, vol. 29, pp. 245–275, 1997.

[104] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, pp. 183–233, 2000.

[105] L. Saul and M. Jordan, "Exploiting tractable substructures in intractable networks," in *Advances in Neural Information Processing Systems 8*, D. Touretzky, M. Mozer, and M. Hasselmo, Eds., 1996, pp. 486–492.

[106] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statistical Society Series B*, vol. 39, pp. 1–38, 1977.

[107] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

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

[109] J. Liu, R. Chen, and T. Logvinenko, "A theoretical framework for sequential importance sampling and resampling," in *Sequential Monte Carlo in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds., New York: Springer-Verlag, 2000.

[110] T.-J. Cham and J. Rehg, "A multiple hypothesis approach to figure tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 1999, pp. 239–244.

[111] J. MacCormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking," in *Proc. of European Conf. on Computer Vision*, vol. 2, 2000, pp. 3–19.

[112] H. Tao, H. Sawhney, and R. Kumar, "A sampling algorithm for detecting and tracking multiple objects," in *Proc. ICCV'99 Workshop on Vision Algorithm*, 1999.

[113] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. II, 2000, pp. 126–133.

[114] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," in *Proc. IEEE Int'l Conf. on Computer Vision*, 1999, pp. 572–578.

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

[116] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. Conf. Computational Learning Theory*, 1998, pp. 92–100.

[117] R. Rosales, S. Sclaroff, and V. Athitsos, "3D hand pose reconstruction using specialized mappings," in *Proc. IEEE Int'l Conf. on Computer Vision*, July 2001.

[118] 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.

[119] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography – a factorized method," *Int'l Journal of Computer Vision*, vol. 9, pp. 137–154, 1992.

[120] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int'l Journal of Computer Vision*, vol. 13, pp. 119–152, 1994.

[121] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," in *Proc. of European Conference on Computer Vision*, April 1996.

[122] M. Weber, M. Welling, and P. Perona, "Towards automatic discovery of object categories," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 101–108.

[123] K. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Advances in Neural Information Processing Systems 11*, M. Kearns, S. Solla, and D. Cohn, Eds., 1998, pp. 368–374.

[124] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. of Int'l Conf. on Machine Learning*, 1999, pp. 200–209.

[125] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, no. 2/3, pp. 103–134, 1999.

[126] Y. Wu, K. Toyama, and T. S. Huang, "Self-supervised learning for object recognition based on kernel Discriminant-EM algorithm," in *Proc. IEEE Int'l Conference on Computer Vision*, vol. I, July 2001, pp. 275–280.

[127] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[128] K. Bennett, "Combining support vector and mathematical programming methods for classification," in *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA: MIT Press, 1999.

[129] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via án EM approach," in *Advances in Neural Information Processing Systems 6*, J. Cowan, G. Tesauro, and J. Alspector, Eds., 1994, pp. 120–127.

[130] V. Tresp, R. Neuneier, and S. Ahmad, "Efficient methods for dealing with missing data in supervised learning," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds., 1995, pp. 689–695.

[131] V. R. de Sa, "Learning classification with unlabeled data," in *Advances in Neural Infomation Processing Systems 6*, J. Cowan, G. Tesauro, and J. Alspector, Eds., 1994, pp. 112–119.

[132] V. R. de Sa and Ballard, "Category learning through multi-modality sensing," *Neural Computation*, vol. 10, no. 5, 1998.

[133] T. Mitchell, "The role of unlabeled data in supervised learning," in *Proc. Sixth Int'l Colloquium on Cognitive Science*, 1999.

[134] E. Riloff and R. Jones, "Learning dictionaries for information extraction by multi-level bootstrapping," in *Proc. AAAI National Conf. on Artificial Intelligence*, 1999, pp. 1044–1049.

[135] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, vol. 1. New York: Interscience Publishers, 1953.

[136] B. Schölkopf, A. Smola, and K. Robert Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[137] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *IEEE workshop on Neural Networks for Signal Proceesing*, 1999.

[138] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K.-R. Müller, "Invariant feature extraction and classification in kernel spaces," in *Advances in Neural Information Processing Systems 12*, S. Solla, T. Leen, and K. Muller, Eds., 2000, pp. 526–532.

[139] V. Roth and V. Steinhage, "Nonlinear discriminant analysis using kernel functions," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Muller, Eds., 2000, pp. 568–574.

[140] V. Roth, V. Steinhage, S. Schröder, A. B. Cremers, and D. Wittmann, "Pattern recognition combing de-noising and linear discriminant analysis within a real world application," in *Computer Analysis of Image and Patterns*, Lecture Notes in Computer Science 1689, Springer-Verlag, 1999.

[141] Y. Wu, Q. Liu, and T. S. Huang, "Tracking, analyzing and recognizing gesture commands," in *Proc. 4th Annual Federated Laboratory Symposium*, March 2000.

[142] Y. Wu and T. S. Huang, "Using unlabeled data in supervised learning by discriminant-EM algorithm," in *Neural Information Processing Systems Workshop on Using Unlabeled Data for Supervised Learning*, Nov. 1999.

[143] Y. Wu, Q. Tian, and T. S. Huang, "Integrating unlabeled images for image retrieval based on relevance feedback," in *Proc. of IEEE Int'l Conf. on Pattern Recognition*, vol. 1, Sept. 2000, pp. 21–24.

[144] Q. Tian, Y. Wu, and T. S. Huang, "Incorporate discriminant analysis with EM algorithm in image retrieval," in *Proc. of IEEE Int'l Conf. on Multimedia and Expo*, vol. 1, 2000, pp. 299–302.

[145] Y. Wu, Q. Tian, and T. S. Huang, "Discriminant-EM algorithm with application to image retrieval," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, June 2000, pp. 222–227.

[146] Y. Wu, K. Toyama, and T. S. Huang, "Wide-range, person- and illumination-insensitive head orientation estimation," in *Proc.IEEE Int'l Conf. on Face and Gesture Recognition*, March 2000, pp. 183–188.

VITA

Ying Wu received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 1994 and the M.S. degree from the Tsinghua University, Beijing, China, in 1997, both in electrical engineering. Since 1997, he has been a Graduate Research Assistant at the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, pursuing his Ph.D. degree. During summer 1999 and 2000, he was a Research Intern with the Vision Technology Group, Microsoft Research, Redmond, WA. Since August 2001, he has been on the faculty as an assistant professor of the Department of Electrical and Computer Engineering at the Northwestern University, Evanston, IL.

His current research interests include computer vision, computer graphics, machine learning, human-computer intelligent interaction, image/video processing, multimedia, and virtual environments. He has published over 30 technical papers in these areas.

He received a Guanghuang Fellowship at the Tsinghua University in 1995-1996, a Schneider Fellowship at the Tsinghua University in 1996-1997, and the Robert T. Chien Award at the University of Illinois at Urbana-Champaign in 2001.