Learning Visual Behavior for Gesture Analysis

by
Andrew David Wilson

B.A., Computer Science
Cornell University, Ithaca, NY
May 1993

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE IN MEDIA ARTS AND SCIENCES
at the
Massachusetts Institute of Technology
June 1995

Signature of Author _____
Program in Media Arts and Sciences
May 26, 1995

Certified by _____
Aaron F. Bobick
Assistant Professor of Computational Vision
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by _____
Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Learning Visual Behavior for Gesture Analysis

by
Andrew David Wilson

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on May 26, 1995
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

Abstract

Techniques for computing a representation of human gesture from a number of example image sequences are presented. We define *gesture* to be the class of human motions that are intended to communicate, and *visual behavior* as the sequence of visual events that make a complete gesture. Two main techniques are discussed: the first computes a representation that summarizes configuration space trajectories for use in gesture recognition. A prototype is derived from a set of training gestures; the prototype is then used to define the gesture as a sequence of states. The states capture both the repeatability and variability evidenced in a training set of example trajectories. The technique is illustrated with a wide range of gesture-related sensory data. The second technique incorporates multiple models into the Hidden Markov Model framework, so that models representing instantaneous visual input are trained concurrently with the temporal model. We exploit two constraints allowing application of the technique to view-based gesture recognition: gestures are modal in the space of possible human motion, and gestures are viewpoint-dependent. The recovery of the visual behavior of a number of simple gestures with a small number of low resolution example image sequences is shown.

We consider a number of applications of the techniques and present work currently in progress to incorporate the training of multiple gestures concurrently for a higher level of gesture understanding. A number of directions of future work are presented, including more sophisticated methods of selecting and combining models appropriate for the gesture. Lastly, a comparison of the two techniques is presented.

Thesis Supervisor: Aaron F. Bobick
Title: Assistant Professor of Computational Vision

Learning Visual Behavior for Gesture Analysis

by
Andrew David Wilson

The following people served as readers for this thesis:

Reader: _____

Mubarak Shah
Associate Professor, Director Computer Vision Laboratory
Computer Science Department, University of Central Florida

Reader: _____

Whitman Richards
Professor of Cognitive Science
Head, Media Arts and Sciences Program

Acknowledgments

Thanks to everyone in the Vision and Modeling Group at the MIT Media Laboratory for introducing me to the weird, wonderful world of machine vision. Big thanks to Aaron for granting me the freedom to make my own sense of it. Special thanks to Tom Minka, Kris Popat and Baback Moghaddam for valuable discussions regarding some ideas presented in the thesis, including multiple models and eigenspace representations.

Contents

List of Figures

Chapter 1

Introduction

1.1 "Killer Demos"

Imagine the following applications:

3-D CAD system: An architect models a new structure by sitting in front of his workstation and manipulating shapes by moving his hands in space. The workstation is familiar with how the various pieces of the structure fit together and so takes care of the precisely placing the objects, while the designer concentrates on the overall design. Two cameras trained on the space in front of the space track the architect's hands; the spatial aspect of her hand movements is exploited in conveying the spatial relationship of the structure's components.

Virtual orchestra: The conductor of the future needs only a MIDI-equipped workstation and a camera to conduct a brilliant performance by his favorite orchestra. Trained to recognize a person's individual conducting style, the workstation decodes the conductor's movements and, equipped with the full score, changes the performance of the orchestra appropriately. The conductor looks to and gestures toward specific sections of the orchestra to craft a personal, tailored performance.

Athletic performance analysis: Practicing for the Summer Olympics, a gymnast uses a vision-based automated judge to score his newest pommel horse routine. The automated judge looks for the completion of certain moves within the routine, scores the precision of the moves, and finally scores his performance according to the exact

criteria used in real performance. After trying a number of routines, the gymnast decides to use the routine yielding the most consistent score from the automated judge.

Smart cameras on the Savannah: Rather than sit in a tree all day waiting for the cheetah to chase an antelope across the plain, the nature show videographer set up his smart video camera and leaves. While away, the video camera scans and films interesting activity, including "antelope-like" and "cheetah-like" movement. Other activities, such as the movement of the grass, are ignored. [1]

1.2 Understanding Human Motion from Video

A key component of turning these dream applications into a daily occurrence is the manner in which the computer learns to recognize human (or as in the last example, animal) movements from visual input. Clearly, we need the computer to become an expert in human motion, or perhaps an expert in just one person's movements. Implicit in the most of the examples is the observation that human movements are unique and can be identified in isolation, but that the particular movements seen are predictable, highly constrained and context-sensitive. If it is indeed true that human movements are predictable and highly constrained, then a computer may be able to infer what it needs to know to understand human motion, or at least some subclass of human motion for which those assumptions have some degree of merit. This thesis presents a number of techniques that may be useful in the implementation of computer systems that understand human motion from visual input.

Vision-based recognition of human body motion is difficult for a number of reasons. Though the human skeleton may be modeled by an articulated structure with many degrees of freedom, the remarkably flexible covering of tissue over the skeleton hides the underlying structure. The staggering variety of configurations and motions that our bodies permit overwhelms any attempt to enumerate all human motion. Furthermore, the degree of variation allowed for a class of movements often depends on context and is difficult to quantify. Except for a few notable exceptions on the face, there are few features clearly useful for tracking. Appearances can be dramatically affected by variations in size, propor-

[1] Originally suggested by Matthew Brand.

tions, clothing, and hair (to name a few dimensions). These factors combine with the other usual problems in computer vision such as the recovery of depth, invariance to changes in illumination and the use of noisy video images, to make the machine understanding of human body motion very difficult.

Consider the motion of a single human hand in a sequence of monocular views. The hand can appear in a practically limitless number of configurations, only a few of which do not exhibit some degree of self-occlusion (recall making shapes in front of the projector). Though a three-dimensional articulating model of the fingers might be useful, such a model would not be able to simply characterize the state of the palm, which itself is a complex bundle of joints and tendons and yet appears as a single deforming body. Also, perhaps except for the tips of the fingers, there are no reliable features of the hand to ease tracking.

1.2.1 Anthropomorphization is a feature, not a bug

One feature of motion recognition that makes our task easier is that object recognition may not be a prerequisite to motion recognition. In fact, it may be easier to recognize that an object has human-like motion and then verify that that the object is indeed a person, than to try to identify the object as a person and then verify that it is indeed moving like a human. The reason is that to make a decision about the motion we can integrate a large amount of information over time, while the task of recognizing the object without movement entails making a conclusion from a single image. By integrating information over time, we can postpone the recognition decision until it becomes obvious that only a human would have gone through a long, particular motion. Additionally, after a number of frames we can be more sure that the features we have measured from the image are indeed not in error. By collecting information from a single frame, we risk the possibility that our feature extractor won't perform well in the particular case represented by the image in question.

We can reach two interesting conclusions from these observations. The first is that working with image sequences is actually easier than working with a single image. This becomes more evident as disk space and memory becomes inexpensive and plentiful. The second conclusion is that anthropomorphization, the ascribing of human qualities to something not human, is a feature. In other words, if something exhibits human motion, we should label the object as human and stop there. Doing so avoids the possibility that a prerequisite object recognition step might fail, possibly for some of the reasons mentioned previously.

Furthermore, an anthropomorphizing computer vision system mimics our own vision system in its ability to make the leap of faith necessary to understand, for example, human-like characters we've never seen before in animated films. One very intelligent answer we might expect from a system that considers the motion independently of recognizing the object is "it moves like a human, but it doesn't look like it's human".

1.2.2 Gesture as a class of human motion

Lately gesture recognition has been a popular topic in computer vision. Unfortunately, as yet there does not seem to be an agreed use of the word "gesture." For some, a gesture is a static configuration of a hand [32]. For others, gesture requires the analysis of joint angles and position of the hand over time [49, 51], while others include facial expressions over time [16]. Still others concentrate on how people unconsciously move their hands and bodies in the context of a spoken discourse with another person [31]. Gesture as a static configuration aside, the only common theme seems to be the understanding of the human body in motion; there is no agreed upon set of primitives used to describe gesture. Until a strong set of primitives is shown to exist for a particular domain of gesture, a variety of techniques will be applied.

Some say that gesture recognition will follow the same course of development as speech recognition. Whilte it is tempting to draw strong parallels between speech and gesture, and some techniques of speech recognition have been useful in recognizing gestures, the two domains differ in the existence of strong primitives. Most speech recognition systems analyze speech at the phonetic, word, utterance, and discourse levels. These agreed-upon primitives outline the appropriate features to measure, and allow systems to draw on specialized techniques developed in other fields. While there has been some effort towards arriving at a useful set of features in hand gesture [61] and facial expressions recognition [17], no such set has received universal acceptance.

The set of useful primitives will more likely develop out of novel user interfaces that incorporate gesture recognition, some of which are under development. As products of these individual efforts, the primitives may be task or domain dependent. Look to the ALIVE system [29], for example, which incorporates wireless understanding of the motion of users interacting with virtual worlds. Another interesting domain is that of musical conductor understanding, where the goal is to allow the automatic and natural control of a virtual

orchestra [34]. Additionally, Bolt and Herranz [4] has explored a number of applications exploiting hand, eye and body movements in computer-human interaction.

1.2.3 Goal

For the purposes of this thesis, we define gesture to be the class of human motions (of hands or otherwise) that are intended to communicate. This limitation on human motion that we impose will be important in making the task of identifying human motion tractable. Even given this limitation, this particular class of human motion is one that is quite useful and will likely motivate many interesting computer human interface applications. Additionally, the techniques presented in this thesis are not limited to human gesture, nor even human motion.

Specifically, this thesis addresses the development of a representation of gesture useful in the analysis, coding and recognition from a number of training image sequences of human gesture. Analysis addresses the problem of determining the structure of the gesture, such as how a gesture relates to other gestures or whether a gesture exhibits a certain kind of temporal structure. The coding problem involves the problem of describing what is happening during the gesture, and how best to describe the events. Recognition is the problem of determining if the gesture has been seen before, and if it has, the identity of the gesture.

1.3 Outline of Thesis

After presenting some previous work related to the thesis in Chapter 2, two different approaches to arriving at a representation for gesture will be presented. First, in Chapter 3, a novel method of summarizing trajectories for gesture recognition will be presented. In this method, examples of each gesture are viewed as trajectories in a configuration space. A prototype gesture is derived from the set of training gestures; the prototype is useful in summarizing the gesture as a sequence of states along the prototype curve.

Chapter 4 presents newer work that takes a different approach to the same task. Rather than model the gesture as a trajectory in a configuration space, the state-based description is captured without first computing a prototype. Furthermore, a multiple model approach is taken in which the representations of the gesture themselves are trained concurrently

with the temporal model of states.

Chapter 5 discusses work in progress on the concurrent training of multiple gestures using the framework presented in Chapter 4.

Chapter 6 presents a comparison of the two techniques and concludes the thesis with a number of interesting topics for future work.

Chapter 2

Related Work

2.1 Introduction

There are four classes of existing research that relate to the goal of the automatic understanding of human motion from visual input, and to the specific techniques presented in this thesis:

- Three-dimensional, physical model-based works that seek to recover the pose of the human body
- Works that explore different representations and uses of trajectories for motion recognition
- View-based approaches that use statistical pattern recognition techniques in the recognition of images and image sequences
- Works that use Hidden Markov Models to model the temporal structure of features derived from images

As mentioned previously, for the purpose of the thesis, gesture should be regarded as any motion of the body intended to communicate. Such a broad definition encompasses hand gestures, facial expressions and even larger body movements.

The works are organized primarily by the techniques employed by the authors and not the objects of study, such as facial motion, gait, or hand gestures, primarily since no set of primitives can be used distinguish the domains.

2.2 Human Body Pose Recovery

In these works, the goal is the recovery of the pose of an explicit three-dimensional physical model of the human body. This entails setting the parameters of a physical body model so that they are consistent with the images. Recognition of previously seen motions is then a matter of matching the model parameters over time to stored prototypes. Because they use explicit physical models of the object of interest, these works are of a tradition in computer vision typically called "model-based" vision.

O'Rourke and Badler [43] exploit physical constraints on human movement, such as limits in the values of joint angles, limits on the acceleration of limbs and the constancy of limb lengths. These constraints are used to limit a search in model parameter space in finding a set of parameters that yield a model consistent with the image. While the exact form of these constraints varies across different works, the same constraints are usually used throughout to make the search tractable. O'Rourke and Badler successfully tracked the hands, feet and head of synthetically generated images of humans.

Hogg [23] pioneered the efforts at recognizing people walking across the scene in a fronto-parallel position. Restricting the system to recognize fronto-parallel walking is significant in that discrete features, such as lines along the limbs derived from edge maps, are clearly visible against the background. Modeling the human form as a number of connected cylinders to track walking people against complex backgrounds, Hogg exploits key-frames to model the walking motion, each key-frame having a unique set of constraints. The key-frames are used to limit the search at each stage of the movement. The data for the motion model, including the key-frames, are acquired interactively from a single example of a walking person.

In some of the most recent model-based work, Rohr [51] uses a model similar to Hogg's, but creates a detailed model of walking from medical studies of sixty men. Again, the motion was fronto-parallel to the camera. The model is parameterized by a single pose parameter, varying along a cycle of walking from zero to one. As is common in the search for consistent physical model parameters, Rohr uses the parameters found in the previous frame to constrain the search for updated parameters in the current frame.

So far, the works presented basically proceed in the same way: search among the model parameters to find a projection of the model consistent with the current frame,

using the previous frame's parameters. Lowe [28] generalizes this approach to articulated three-dimensional models.

Chen and Lee [10] enforce constraints over multiple frames of gait data, casting the problem as a graph search to select among candidate postures at each frame. The algorithm thus selects a globally optimal solution of postures. Approaches such as theirs are appealing in that by seeking a globally optimal solution, the method is more robust to failures in a single frame.

A number of researchers have employed three-dimensional models of the hand, usually modeled as an articulated object with many degrees of freedom. Charayaphan [9] and Dorner [14] make the task easier by recovering the position of fiducial marks placed on the hand.

Rehg [50] has implemented a system that tracks the joint angles of the unadorned hand in real time from two cameras. The twenty-seven degrees of freedom of the hand model are recovered using geometric and kinematic constraints similar to O'Rourke and Badler's. The system relies on a high temporal sampling rate so that, again, model parameters from the previous frame effectively initialize the search for new model parameters. Rehg recovers the position of the finger tips by searching along each finger, beginning at the base of the finger. The model is brought into alignment to match the recovered finger tips. Recently the system has been modified to model certain kinds of self-occlusion, again exploiting temporal coherence in reasoning that the occlusion properties should not change drastically from frame to frame.

In summary, the "model-based" approaches appeal in their theoretical ability to recover unconstrained human motion. But seldom is this ability achieved; most systems work in highly constrained situations only. Most systems are designed without regard to the suitability of the sensors used to the task of recovering a fully specified three-dimensional model. In part, deficiencies in the sensors can be made up by exploiting domain knowledge, but in general the systems are fragile.

Unless the goal is to construct a detailed model of the human body for computer graphics applications, an exact recovery of the three-dimensional configuration of the body is unnecessary. In the context of hand gesture recognition, work on the low bit-rate transmission of American Sign Language (ASL) video sequences [37, 55] suggests that the precise recovery of features necessary for a model-based recognition of the moving hand such as

in Rehg's system is unnecessary for at least the comprehension of ASL. In working hard to recover a complete model, the system may risk many mistakes, only to be asked a very simple question that doesn't require such detailed knowledge. The view presented in this thesis is that the model of human motion should be represented in terms of the sensor measurements available. This does not prevent the use of three-dimensional models, but instead allows for the use of weaker, more reliable models.

One interesting variant of the recovery of pose using three-dimensional models is the recovery of internal control parameters. Essa, Darrell and Pentland [16] have formulated the learning of the relationship of the model- and view-based descriptions as a dynamical systems problem for the domain of facial gesture recognition. Proceeding from optical flow images, the activation of a large number of muscle activation levels are inferred. The muscles are in a three-dimensional configuration known beforehand; thus the task is not the recovery of the three-dimensional structure of the face, but the recovery of internal control parameters that result in a facial expression.

2.3 Trajectory Analysis

Another class of related work explores the use of trajectory representations for recognizing human motion from joint angles. The trajectories are typically parameters from model-based descriptions over time, though the changing output of a view-based sensor over time may be a trajectory in its own right.

Gould and Shah [20] consider the recognition of moving objects by their motion alone. They show how motion trajectories can be analyzed to identify event boundaries which are then recorded in their *Trajectory Primal Sketch* (TPS). The computation of the TPS for a number of different kinds of motion, including translation, rotation and projectile motion, are considered in Gould, Rangarajan and Shah [19]. Rangarajan et al. [47] demonstrate 2D motion trajectory matching through scale-space, and mention that the 3D motion trajectory could be stored with the model 2D trajectories. They state a goal of distinguishing between two similar objects with different motions, or two objects with the same motion but different shapes.

For the recognition of classical ballet steps, Campbell and Bobick [5] summarize 3-dimensional motion trajectories in a variety of subspaces of a full *phase space*. Various

2-dimensional subspaces are searched for their ability to accurately predict the presence of a particular dance step in a set of example steps. The particular subspace for recognition of a movement is chosen to maximize the rate of correct acceptance and minimize false acceptance. Campbell and Bobick also address the notion of combining the outputs of these various *predictors* for robust recognition.

Early on Rashid [48] considered the two-dimensional projection of three-dimensional points similar to Johansson's Multiple Light Displays [25]. Points were tracked over a number of frames and clustered based on relative velocities. In some cases, this was sufficient to segment independently moving objects in the scene, as well as points that were rigidly linked in space. Shio [54] exploits a similar technique in segmenting people from a busy street scene by clustering patches of the image that exhibit the same average velocity computed over a few seconds of video.

Allmen and Dyer [1] compute optic flow over the frames of an image sequence and then trace out curves along similar regions of optic flow. These curves are then clustered into coherent motions such as translation or rotation. Allmen and Dyer demonstrate their system by recovering the common and relative motion of a bird flapping its wings. They call their approach "dynamic perceptual organization" for the way motion information is clustered in an unsupervised way to provide a summary of the observed data. The work presented in this thesis proceeds in a similar fashion.

Niyogi has performed similar curve analysis in space-time volumes assembled from image sequences. Niyogi observed that people walking fronto-parallel to the camera trace out distinct periodic curves at certain slices across space of the space-time volume. By fitting contour snakes to these curves, the system can recover the period of the walker and perform simple gait recognition.

Lastly, some work has been done in the real time recognition of simple mouse input device gestures. Tew and Gray [57] use dynamic programming to match mouse trajectories to prototype trajectories. Lipscomb [27] concentrates on filtering the mouse movement data to obtain robust recognition of similarly filtered models. Mardia et al. [30] compute many features of each trajectory and use a learned decision tree for each gesture to best utilize the features for recognition.

One criticism of the approach of analyzing trajectories of human motion is that in most situations the recovery of discrete points on the human body such as the limbs, feet, hands

and head is not robust enough to be practical. In fact it may be the case that only the positions of the extremities of the body may be reliably recovered, and then only if the background is known. But, as will be demonstrated in the thesis, recognition by trajectory is feasible when the features are not the position of tracked points on the body but the smoothly varying output of an image operator on each frame. If the frames in an image sequence vary smoothly in time, then the output of an image operator applied to each image may change smoothly as well.

2.4 View-Based Approaches

The third category of research involves applying statistical pattern recognition techniques to images without deriving an explicit physical model. Sometimes these works are termed "model-less", as they do not incorporate strong three-dimensional models. The term "model-less" is somewhat misleading, since in truth every approach incorporates some way of abstracting and representing what is seen. Thus, every approach relies on some sort of model. Even viewed as a way of simply distinguishing explicit three-dimensional model-based approaches and those that do not rely on three-dimensional models, the term discounts the possibility of using a "weak" physical model. A "weak" physical model might represent the fact that a person's hand is somewhere in front of the person's body, for example, rather than the exact location of the hand in space. This distinction of terms might seem a bit pedantic at first, but it is useful in situating the present work among the works discussed in the chapter.

In a view-based approach no attempt is made to recover parameters of a three-dimensional model from the image. In these studies features of the images alone are used in a statistical formulation. Recognition consists of checking whether the new data match the statistics of the examples. These techniques purportedly allow a degree of generalization and robustness not possible with the model-base approach.

Considering the view-based approach, the Center for Biological and Computational Learning at MIT has been exploring neural networks as function approximators. Poggio and Edelman [40] have demonstrated how a neural net can learn to classify 3D objects on visual appearances alone. Their approach considers each image as a point in the high dimensional space of image pixel values. The value associated with each image (for example,

the pose of the object) is the value of the approximated function at that point. Under some assumptions of smoothness, the technique will generalize to unseen views correctly. Beymer's [2] ongoing work shows how the network can go one step further by synthesizing the actual view associated with an unseen view of a given pose.

Darrell and Pentland [12] have similarly explored a representation based on the interpolation of views with their real time wireless hand gesture recognition system. Model hand gestures are represented as a pattern of responses to matched filters that are acquired during the presentation of examples. New model matched filters, or templates, are acquired during a gesture when no other templates respond vigorously. During recognition, the responses to the previously computed model templates are time warped to stored prototypes. The method of dynamic time warping presented, borrowed from the speech recognition community [46, 21], is a useful and efficient technique to achieve the time shift-invariant matching of two time-varying signals. Darrell's approach is attractive in how both the view models and view predictions are learned during the presentation of the examples.

Murase and Nayar [36] have also studied the automatic learning of object models for the recovery of model parameters. They take the views of an object under all possible rotations about one axis and all angles of illumination. These images are then projected to a lower dimensional *eigenspace* as in [59]. The constraints imposed by the appearance of the object manifest themselves as a hyper-surface in the eigenspace, which is parameterized by object pose and illumination. This they call a *parametric eigenspace.* Recognition consists of projecting the new image into this eigenspace and finding the nearest point on the hyper-surface. The parameters associated with this nearest point return the object pose and angle of illumination. Because the objects used are rigid and the entire view sphere and all illumination conditions are sampled, the hyper-surface is a smooth ring in eigenspace. Furthermore, since all viewing conditions are known and parameterized (the mapping between the view- and model-based representations is "one-to-one" and "onto"), there is no need to summarize the allowable combinations of model parameters. This work by Murase and Nayar relates closely to the second half of the work presented in this thesis.

Pentland et al. [38] take a slightly different approach with their *modular eigenspaces* for face recognition. Rather than use the same eigenspace for all example facial views, a different eigenspace is used for each significantly different value of a parameter. For example, for a data set of example views of a number of people under a number of different

poses, the views under the same pose condition are likely to be more similar to each other than views of the same person under different pose conditions. Therefore the eigenspace representation for all views of a particular pose condition is computed. Recognition consists of determining which eigenspace the example view belongs to, and then projecting the view into that eigenspace for accurate recognition.

In his paper entitled "On Comprehensive Visual Learning" Weng [60] discusses the role of learning in computer vision algorithms. Weng presents a general architecture of attention, but more importantly points out that representations based on eigenvector decomposition are not necessarily optimal for recognition. The approach presented is to build an optimal set of features in the space resulting from the eigenvector decomposition. Cui and Weng [11] demonstrate their approach in learning to recognize hand signs from video images. While only static positions of hands are recognized, the system appears to be robust. Their representation shows only a modest increase in recognition performance over using just the classifications resulting from eigenvector decomposition.

Moghaddam and Pentland [33] cast the classification by eigenvector decomposition of the images into a maximum likelihood framework, performing similar detection activities as in Cui and Weng. Essentially, their work combines a classification based on nearness in the space of eigenvector projection coefficients (termed "distance in feature space") and the reconstruction residual of the image (termed "distance from feature space") to give a complete probabilistic description of images in image-space.

Polana and Nelson [41] have conducted novel research in extracting low level features for the recognition of periodic motion, such as walking and swimming. They compute the Fourier transform of patches of the image over time, and sum the peak frequencies over the spatial domain to obtain a measure of the periodic activity in the sequence. In another work, they showed that by looking at the spatial distribution of movement in the image, they were further able to recognize certain periodic motions. Their technique appears to be limited to periodic motions.

In work of a more applications-oriented flavor Kjeldsen and Kender [26] describe a system in development that is designed to provide control of various window system activities by hand gestures from video. The system is interesting because it has a strong notion of what is required during the different phases of predefined motions, and adjusts the features computed accordingly. For example, during a certain "command" the exact configuration

of the hand may not be important, so only low resolution images are captured during that phase. They exploit Quek's [42] observation that it is rare for both the pose of the hand and the position of the hand to simultaneously change in a meaningful way during a gesture.

This collection of works offers a variety of approaches, but they all represent what they know in terms of sensor outputs. This more general approach is attractive in the case of human motion, where there is no consensus as to the appropriate primitives of description.

2.5 Hidden Markov Models for Gesture Recognition

Yet another approach is to model the various phases of movements as a state-transition diagram or Markov model. For example, Davis and Shah [13] used a simple finite state machine to model distinctly different phases of a gesture. Recognition is bootstrapped by assuming that the motion starts from a given state.

Hidden Markov Models (HMMs) [45, 44, 24] associate a description of the continuous sensor outputs that are consistent with each of the discrete states in a Markov model. Today the most successful speech recognition systems employ HMMs for recognition. The Baum-Welch algorithm is an iterative algorithm for inferring an unknown Markov model from multiple series of sensor outputs. The technique is a particularly useful representation for gesture because it puts time shift invariance, or dynamic time warping (as in [12]), on a probabilistic foundation. By recovering both the Markov model describing transitions among states and the states coding a representation of the samples local in time, the HMM technique allows us to neatly separate the task of modeling the time course of the signal from the task of representing the signal given a fixed moment in time.

Yamato et al. [64] base their recognition of tennis swings from video on an HMM which infers an appropriate model after sufficiently many examples. A rather arbitrary region-based feature is calculated from each frame; these features are then quantized to a series of discrete symbols to be passed to the HMM. Together with the Markov model, the simple feature computed was enough to distinguish among the different swings.

Schlenzig, Hunter and Jain [52] similarly recognize hand gestures with HMMs by segmenting the hand from the image and computing a bounding box which is then discretized. In later work, Schlenzig et. al [53] compute the Zernicke moments of binary silhouette image of the hand. The Zernicke moments provide a small basis set that provides reconstruction

of the image (as with an eigenvector decomposition) and uses a rotation-invariant representation. In some hand gestures, invariance to specific rotations of the hand is desirable. For example, all pointing gestures may be identified with one rotation-invariant representation of the hand with the index finger extended. The Zernicke moment representation is then passed to a neural network classifier to determine the pose of the hand, one of six static configurations of the hand. The pose is then fed to an HMM describing the legal motions to command a remotely controlled vehicle.

Starner [56] demonstrates that very simple features computed from video are enough to distinguish among a subset of two-handed ASL gestures. His system extracts the position and dominant orientation of the hands wearing colored gloves. Starner couples a language model that specifies the way the gestures may be combined with a number of HMMs coding the motions. The system achieves impressive recognition results given the limited about of information derived from the images.

Yamato's system is limited by the primitive region-based feature computed at every frame. Schlenzig's work is interesting and highly developed, but by classifying the poses with a neural network before invoking the HMM, the system is not allowed to recover the poses that are appropriate for the movement; the relevant poses are provided beforehand. In Starner's work the "poses" are learned in the training of the HMM, but the topology of the HMM is fixed. By using simple features appropriate for two-handed input, Starner's work is limited to his task.

2.6 Summary

Several categories of approaches to the vision-based understanding of human motion were presented. Those that rely on an explicit physical model of the body are attractive because they are able to model unconstrained human motion, and recognition of particular movements can be accomplished in a parameter space that is natural for the object. But it is not clear that such precise knowledge is necessary or even possible for a given domain.

The related work on trajectory analysis offers a number of techniques that may be useful in analyzing human motion. For example, looking at groups of trajectories, we may be able to deduce the common and relative motion present in a moving object. Or, also by looking at a group of trajectories, we may be able to tell if a certain motion is periodic.

The view-based approaches are interesting because, unlike the more model-based approaches, the knowledge about the movements is grounded in the language of the sensor. Thus the systems do not draw conclusions that the sensor readings cannot support. A number of the techniques presented in this category will be used in the present work.

Finally, the HMM-based approaches are valuable in that they begin to address issues such as the temporal structure of the movement. Rather than simply time warping a signal to a stored prototype, HMMs can represent interesting temporal stricture by varying the Markov model topology.

In many ways, the work presented in this thesis is closely related to these last approaches involving HMMs. The techniques and assumptions used in the first half of the thesis work bear a strong resemblance to those of HMMs. In the second half of the work, the HMM technique is used explicitly in moving towards a state-based description of gesture while simultaneously providing dynamic time warping of the input signal.

The existing applications using HMMs however are somewhat limited in the way the representation and temporal structure are determined by each other. The second half of this thesis expands the application of the technique for the purposes of human motion understanding, such that the temporal model and the representation models work together.

Chapter 3

Configuration States for Gesture Summarization and Recognition

3.1 Introduction: Recognizing Gestures

A gesture is a motion that has special status in a domain or context. Recent interest in gesture recognition has been spurred by its broad range of applicability in more natural user interface designs. However, the recognition of gestures, especially natural gestures, is difficult because gestures exhibit human variability. We present a technique for quantifying this variability for the purposes of summarizing and recognizing gesture.[1]

We make the assumption that the useful constraints of the domain or context of a gesture recognition task are captured implicitly by a number of examples of each gesture. That is, we require that by observing an adequate set of examples one can (1) determine the important aspects of the gesture by noting what components of the motion are reliably repeated; and (2) learn which aspects are loosely constrained by measuring high variability. Therefore, training consists of summarizing a set of motion trajectories that are smooth in time by representing the variance of the motion at local regions in the space of measurements. These local variances can be translated into a natural symbolic description of the movement which represent gesture as a sequence of *measurement states.* Recognition is then performed by determining whether a new trajectory is consistent with the required sequence of states.

The state-based representation we develop is useful for identifying gestures from video if

[1] This chapter has been previously published as a conference paper, coauthored with A. F. Bobick [3, 62].

a set of measurements of key features of the gestures can be taken from images. For example, the two-dimensional motion information derived from Moving Light Display images, while far from fully characterizing three-dimensional motion, may serve as key features for many kinds of gesture.

We apply the measurement state representation to a range of gesture-related sensory data: the two-dimensional movements of a mouse input device, the movement of the hand measured by a magnetic spatial position and orientation sensor, and, lastly, the changing eigenvector projection coefficients computed from an image sequence. The successful application of the technique to all these domains demonstrates the general utility of the approach.

We motivate our particular choice of representation and present a technique for computing it from generic sensor data. This computation requires the development of a novel technique for collapsing an ensemble of time-varying data while preserving the qualitative, topological structure of the trajectories. Finally we develop methods for using the measurement state representation to concurrently segment and recognize a stream of gesture data. As mentioned, the technique is applied to a variety of sensor data.

3.2 Motivation for a Representation

If all the constraints on the motion that make up a gesture were known exactly, recognition would simply be a matter of determining if a given movement met a set of known constraints. However, especially in the case of natural gesture, the exact movement seen is almost certainly governed by processes inaccessible to the observer. For example, the motion the gesturer is planning to execute after a gesture will influence the end of the current gesture; this effect is similar to co-articulation in speech. The incomplete knowledge of the constraints manifests itself as variance in the measurements of the movement. A representation for gesture must quantify this variance and how it changes over the course of the gesture.

Secondly, we desire a representation that is invariant to nonuniform changes in the speed of the gesture to be recognized. These shifts may also be thought of as non-linear shifts in time. A global shift in time caused by a slight pause early in the gesture should not affect the recognition of most of the gesture. Let us call the space of measurements that

define each point of an example gesture a *configuration space*. The goal of time invariance is motivated by the informal observation that the important quality in a gesture is how it traverses configuration space and not exactly when it reaches a certain point in the space. In particular, we would like the representation to be time invariant but order-preserving: e.g. first the hand goes up, then it goes down.

Our basic approach to quantifying the variances in configuration space and simultaneously achieving sufficient temporal invariance is to represent a gesture as a sequence of states in configuration space. We assume that gestures can be broken down into a series of "states". In fact there seems to be no *a priori* reason why this should be so for such a continuous domain like motion, other than that we tend to see event or motion boundaries easily (see [37], for example). In some interesting research, Edelman [15] concludes that a low-dimensional space of features may account for our ability to discriminate among 3D objects, and furthermore that this space may be spanned by a small number of class prototypes. Possibly, then, motion may be viewed as smooth movement from one class prototype (or "state") to another, in which case a finite state model of motion is not so unreasonable.

Each *configuration state* is intended to capture the degree of variability of the motion when traversing that region of configuration space. Since gestures are smooth movements through configuration space and not a set of naturally defined discrete states, the configuration states $S = \{s_i, 1 \leq i \leq M\}$ should be thought of as being "fuzzy", with fuzziness defined by the variance of the points that fall near it. A point moving smoothly through configuration space will move smoothly among the fuzzy states defined in the space.

Formally, we define a *gesture* as an ordered sequence of fuzzy states $s_i \in S$ in configuration space. This contrasts with a *trajectory* which is simply a path through configuration space representing some particular motion. A point x in configuration space has a membership to state s_i described by the fuzzy membership function $\mu_{s_i}(x) \in [0, 1]$. The states along the gesture should be defined so that all examples of the gesture follow the same sequence of states. That is, the states should fall one after the other along the gesture. We represent a gesture as a sequence of n states, $G_\alpha = \langle \alpha_1 \alpha_2 .. \alpha_n \rangle$, where states are only listed as they change: $\alpha_i \neq \alpha_{i+1}$.

We can now consider the state membership function of an entire trajectory. Let $T_i(t)$ be the i^{th} trajectory. We need to choose a *combination rule* that defines the state membership of a point x in configuration space with respect to a group of states. For convenience let us

choose `max`, which assigns the combined membership of x, $M_S(x)$, the value $\max_i(\mu_{s_i}(x))$. The combined membership value of a trajectory is a function of time while the assigned state of the trajectory at each time instant is the state with greatest membership. Thus, a set of configuration states translates a trajectory into a symbolic description, namely a sequence of states.

Defining gestures in this manner provides the intuitive definition of a *prototype gesture*: the motion trajectory that gives the highest combined membership to the sequence of states that define the gesture. We can invert this logic in the situation in which we only have several examples of a gesture: first compute a prototype trajectory, and then define states that lie along that curve that capture the relevant variances. In the next section we will develop such a method.

3.3 Computing the Representation

3.3.1 Computing the prototype

Each example of a gesture is a trajectory in configuration space defined by a set of discrete samples evenly spaced in time. At first, it is convenient to parameterize the i^{th} trajectory by the time of each sample: $T_i(t) \in \Re^d$.

Our definition of a prototype curve of an ensemble of training trajectories $T_i(t)$ is a continuous one-dimensional curve that best fits the sample points in configuration space according to a least squares criterion. For ensembles of space curves in metric spaces there are several well known techniques that compute a "principal curve" [22] that attempt to minimize distance between each point of each of the trajectories and the nearest point on the principal curve.

The prototype curve for a gesture removes time as an axis, and is simply parameterized by arc length λ as it moves through configuration space, $P(\lambda) \in \Re^d$. The goal of the parameterization is to group sample points that are nearby in configuration space and to preserve the temporal order along each of the example trajectories.

The problem of computing a prototype curve P in configuration space is how to collapse time from the trajectories $T_i(t)$. Figure 3-1 illustrates the difficulty. If the points that make up the trajectories (a) are simply projected into configuration space by removing time (b), there is no clear way to generate a connected curve that preserves the temporal ordering

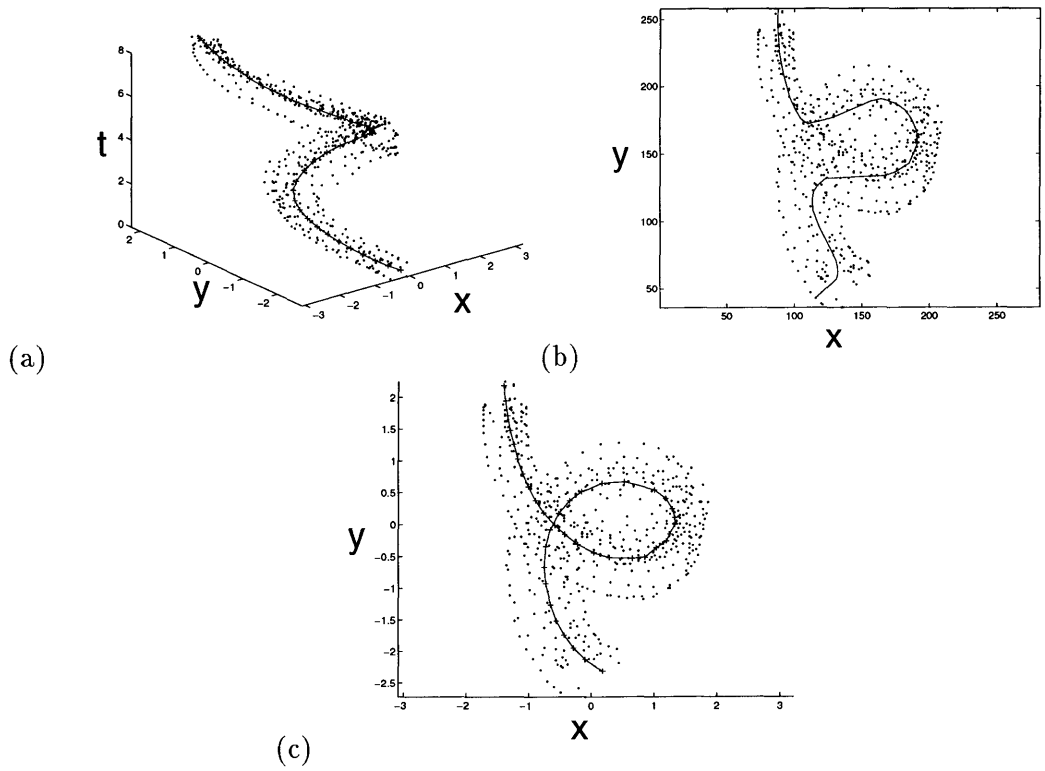


Figure 3-1: (a) Example trajectories as a function of time. (b) Projection of trajectory points into configuration space. Normal principal curve routines would lose the intersection. (c) Prototype curve recovered using the time-collapsing technique (see Appendix).

of the path through configuration space. Likewise, if each of the trajectories is projected into configuration space small variations in temporal alignment make it impossible to group corresponding sections of the trajectories without considering time.

The details of our method are presented in an appendix but we give the intuition here. Our approach is to begin with the trajectories in a time-augmented configuration space. Since a trajectory is a function of time, we can construct a corresponding curve in a space consisting of the same dimensions as configuration space plus a time axis. After computing the principal curve in this space, the trajectories and the recovered principal curve are slightly compressed in the time direction. The new principal curve is computed using the previous solution as an initial condition for an iterative technique.

By placing constraints on how dramatically the principal curve can change at each time step, the system converges gracefully to a prototype curve in configuration space that minimizes distance between the example trajectories and the prototype, while preserving temporal ordering. Figure 3-1(c) shows the results of the algorithm. The resulting prototype curve captures the path through configuration space while maintaining temporal ordering.

An important by-product of calculating the prototype is the mapping of each sample point x_i of a trajectory to an arc length along the prototype curve $\lambda_i = \lambda(x_i)$.

3.3.2 Clustering the sample points

To define the fuzzy states s_i, the sample points of the trajectories must be partitioned into coherent groups. Instead of clustering the sample points directly, we cluster the vectors defining $P(\lambda)$ and then use the arc length parameterization $\lambda(x_i)$ to map sample points to the prototype $P(\lambda)$. The vectors that define $P(\lambda)$ are simply the line segments connecting each point of the discretized $P(\lambda)$, where the length of each line segment is constant.

By clustering the vectors along $P(\lambda)$ instead of all sample points, every point that projects to a certain arc length along the prototype will belong to exactly one cluster. One desirable consequence of this is that the clusters will fall one after the other along the prototype. This ordered sequence of states is recorded as $G_\alpha = \langle \alpha_1 \alpha_2 .. \alpha_n \rangle$.

The prototype curve vectors are clustered by a k-means algorithm, in which the distance between two vectors is a weighted sum of the Euclidean distance between the bases of the vectors and a measure of the difference in (unsigned) direction of the vectors. This difference in direction is defined to be at a minimum when two vectors are parallel and at a maximum

when perpendicular.

Clustering with this distance metric groups curve vectors that are oriented similarly, regardless of the temporal ordering associated with the prototype. If the prototype visits a part of configuration space and then later revisits the same part while moving in nearly the same (unsigned) direction, both sets of vectors from each of the visits will be clustered together. The sample points associated with both sets will then belong to the single state which appears multiply in the sequence G_α. In this way, the clustering leads to a parsimonious allocation of states, and is useful in detecting periodicity in the gesture.

Each cluster found by k-means algorithm corresponds to a fuzzy state. The number of clusters k must be chosen carefully so that there are sufficiently many states to describe the movement in a useful way, but should not be so great that the number of sample points in each cluster is so low that statistics computed on the samples are unreliable. Furthermore, the distribution of states should be coarse enough that all the examples traverse the states in the same manner as the prototype.

3.3.3 Determining state shapes

The center of each of the clusters found by the k-means algorithm is the average location c and average orientation $\vec{v}$ of the prototype curve vectors belonging to the cluster. The membership function for the state is computed from these center vectors and the sample points that map to the prototype curve vectors in each cluster.

For a given state s_i, the membership function $\mu_{s_i}(x)$ should be defined so that it is greatest along the prototype curve; this direction is approximated by $\vec{v}$. Membership should also decrease at the boundaries of the cluster to smoothly blend into the membership of neighboring fuzzy states. Call this membership the "axial" or "along-trajectory" membership. The membership in directions perpendicular to the curve determines the degree to which the state generalizes membership to points on perhaps significantly different trajectories. Call this membership the "cross sectional" membership.

A single oriented Gaussian is well suited to model the local, smooth membership function of a fuzzy state. Orienting the Gaussian so that one axis of the Gaussian coincides with the orientation $\vec{v}$ of the center of the state, the axial membership is computed simply as the variance of the sample points in the axial direction. The cross-sectional membership is computed as the variance of the points projected on a hyper-plane normal to the axis.

It should be noted that these variances are correctly estimated from the points in the original configuration space and not the time-augmented configuration space, since, as mentioned before, the sample point to prototype curve correspondence is slightly in error in the larger space. Thus we cannot simply compute the variances in the larger space and drop the time component.

The inverse covariance matrix Σ^{-1} of the oriented Gaussian can be computed efficiently from the covariance matrix Σ of the points with the center location subtracted. First, a rotation matrix R is constructed, whose first column is $\vec{v}$, the axial direction, and whose remaining columns are generated by a Gram-Schmidt orthogonalization. Next, R is applied to the covariance matrix Σ:

$$\Sigma_R = R^T \Sigma R = \begin{bmatrix} \sigma_v^2 & \cdots \\ \vdots & [\Sigma_{proj}] \end{bmatrix}$$

where σ_v^2 is the variance of the points along $\vec{v}$, and Σ_{proj} is the covariance of the points projected onto the hyper-plane normal to $\vec{v}$. We can scale each of these variances to adjust the cross sectional and axial variances independently by the scalars σ_c^2 and σ_a^2, respectively. Setting the first row and column to zero except for the variance in direction $\vec{v}$:

$$\Sigma_{R'} = \begin{bmatrix} \sigma_a^2 \sigma_v^2 & \cdots 0 \cdots \\ \vdots & \\ 0 & \sigma_c^2 [\Sigma_{proj}] \\ \vdots & \end{bmatrix}$$

Then the new inverse covariance matrix is given by: $\Sigma_S^{-1} = (R \Sigma_{R'} R^T)^{-1}$.

A state s_i is then defined by c, $\vec{v}$, and Σ_S^{-1}, with $\mu_{s_i}(x) = e^{-(x-c)\Sigma_S^{-1}(x-c)^T}$. The memberships of a number of states can be combined to find the membership $\mu_{s_i,s_j,..}(x)$ to a set of states $\{s_i, s_j, ..\}$. As mentioned, one combination rule simply returns the maximum membership of the individual state memberships:

$$\mu_{s_i,s_j,..}(x) = \max_{s \in \{si,sj,..\}} \mu_s(x)$$

3.4 Recognition

The online recognition of motion trajectories consists of explaining sequences of sample points as they are taken from the movement. More concisely, given a set of trajectory sample points $x_1, x_2, ..x_N$ taken during the previous N time steps, we wish to find a gesture G and a time t_s, $t_1 \leq t_s \leq t_N$ such that $x_s..x_N$ has an average combined membership above some threshold, and adequately passes through the states required by G.

Given the sequence of sample points at $t_1..t_N$, we can compute t_s and the average combined membership for a gesture $G_\alpha = \langle \alpha_1 \alpha_2 .. \alpha_n \rangle$ by a dynamic programming algorithm. The dynamic programming formulation used is a simplified version of a more general algorithm to compute the minimum cost path between two nodes in a graph.

For the dynamic programming solution, each possible state at time t is a node in a graph. The cost of a path between two nodes or states is the sum of the cost assigned to each transition between adjacent nodes in the path. The cost of a transition between a state α_i at time t and a state α_j at time $t+1$ is

$$c_t(\alpha_i, \alpha_j) = \begin{cases} \infty & \text{for } j < i \\ 1 - \mu_{\alpha_j}(T(t)) & \text{otherwise} \end{cases}$$

That is, we are enforcing forward progress through the states of the gesture and preferring states with high membership.

The dynamic programming algorithm uses a partial sum variable, $C_{t_i,t_j}(\alpha_i, \alpha_j)$ to recursively compute a minimal solution. $C_{t_i,t_j}(\alpha_i, \alpha_j)$ is defined to be the minimal cost of a path between state α_i at a time t_i and α_j at a time t_j:

$$C_{t_i,t_j}(\alpha_k, \alpha_m) = \min_{\alpha_l \in G_\alpha} \left\{ c_{t_i}(\alpha_k, \alpha_l) + C_{ti+1,t_j}(\alpha_l, \alpha_m) \right\}$$

$$C_{t_i,t_i}(\alpha_k, \alpha_m) = 0$$

The total cost associated with explaining all samples by the gesture G_α is then $C_{t_1,t_N}(\alpha_1, \alpha_n)$.

The start of the gesture is not likely to fall exactly at time t_1, but at some later time t_s. Given t_s, we can compute the average combined membership of the match from the total cost to give an overall match score for a match starting at time t_s:

$$\bar{\mu}_{G_\alpha} = 1 - \frac{C_{t_s,t_1}(\alpha_1, \alpha_n)}{(t_N - t_s)}$$

To be classified as a gesture G_α, the trajectory must have a high match score and pass through all the states in G_α as well. For the latter we can compute the minimum of the maximum membership observed in each state α_i in G_α. This quantity indicates the *completeness* of the trajectory with respect to the model G_α. If the quantity is less than a certain threshold, the match is rejected. The start of a matching gesture is then

$$t_s = \arg\min_t Ct, t_N(\alpha_1, \alpha_n), completeness > threshold$$

Causal segmentation of a stream of samples is performed using the dynamic programming algorithm at successive time steps. At a time step t, the highest match score and the match start time t_s is computed for all samples from t_0 to t. If the match score is greater than a threshold η, and the gesture is judged complete, then all points up to time t are explained by a gesture model and so are removed from the stream of points, giving a new value $t_0 = t$. Otherwise, the points remain in the stream possibly to be explained at a later time step. This is repeated for all time steps t successively.

3.5 Experiments

The configuration-state representation has been computed with motion trajectory measurements taken from three devices that are useful in gesture recognition tasks: a mouse input device, a magnetic spatial position and orientation sensor, and a video camera. In each case we segment by hand a number of training examples from a stream of smooth motion trajectories collected from the device. With these experiments we demonstrate how the representation characterizes the training examples, recognizes new examples of the same gesture, and is useful in a segmentation and tracking task.

3.5.1 Mouse Gestures

In the first experiment, we demonstrate how the representation permits the combination of multiple gestures, and generalizes to unseen examples for recognition. Furthermore,

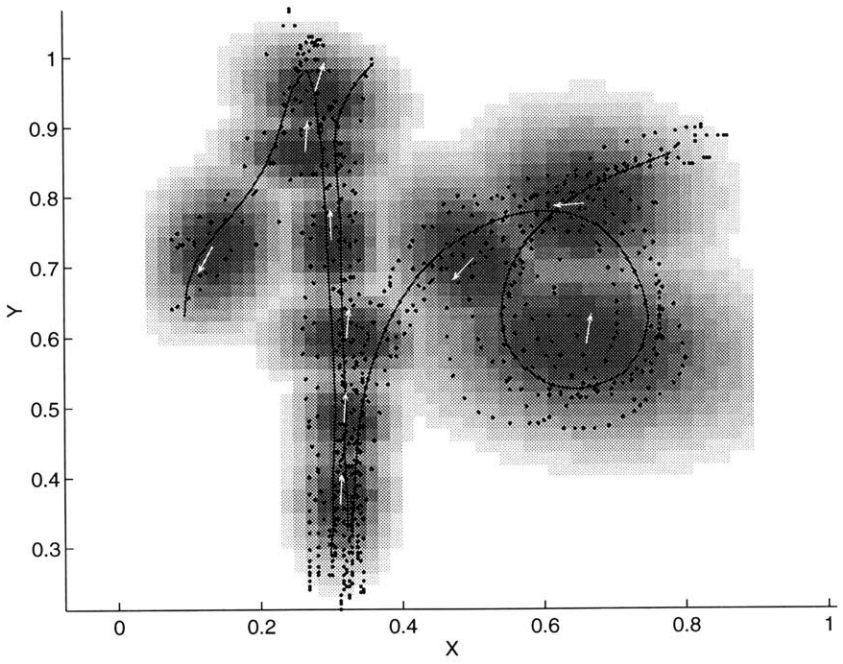


Figure 3-2: The prototype curves (black) for G_α and G_β are combined to find a set of states to describe both gestures. The clustering of the prototype curve vectors for G_α and G_β gives the clustering of all sample points. Each of the ten clusters depicted here is used to define a state centered about each of the white vectors. Darker regions indicate high membership.

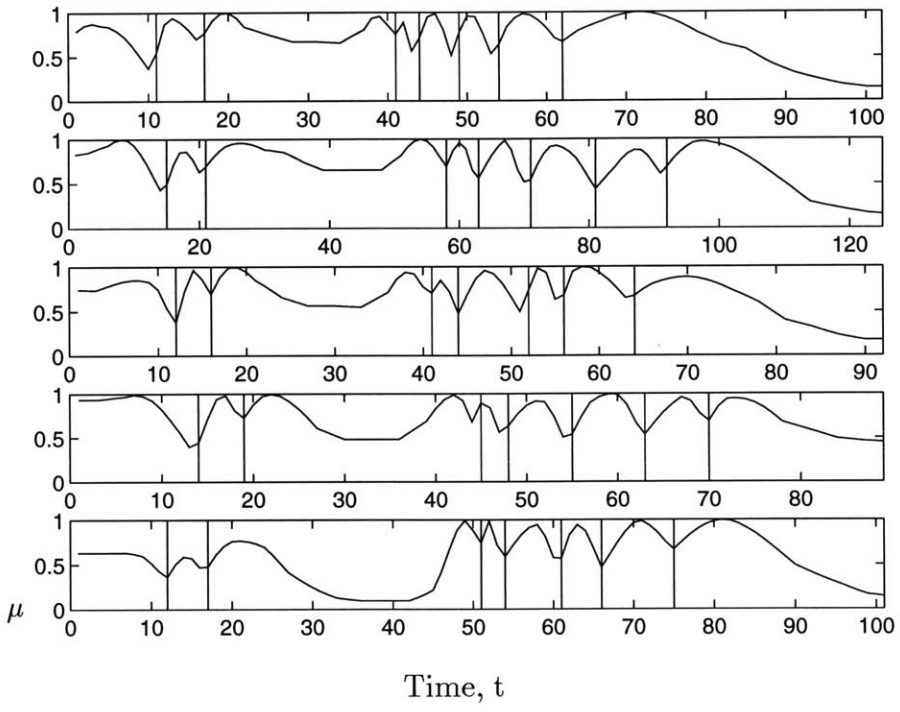


Figure 3-3: The state transition and membership plots for the testing examples for G_α. The transitions (vertical bars) are calculated to maximize the average membership while still giving an interpretation that is complete with respect to G_α. The plot depicts the time-invariant but order-preserving nature of the representation.

by considering two-dimensional data, the cluster points and the resulting states are easily visualized.

The (x, y) position of a mouse input device was sampled at a rate of 20 Hz for two different gestures G_α and G_β. Ten different examples of each gesture were collected; each consisted of about one hundred sample points. Half of the examples were used to compute the prototype curve for each gesture. The vectors along both of the curves were then clustered to find a single set of ten states useful in tracking either gesture. Because the gestures overlap, six of the states are shared by the two gestures.

The shapes of the states computed from the clustering is shown in Figure 3-2. The generalization parameter σ_c had a value of 3.0. The state sequences G_α and G_β were computed by looking at the sequence of state assignments of the vectors along the prototype. The sequences G_α and G_β reflect the six shared states: $G_\alpha = \langle s_1 s_2 s_3 s_2 s_4 s_5 s_6 s_7 \rangle$, $G_\beta = \langle s_3 s_2 s_4 s_5 s_6 s_7 s_6 s_5 s_8 s_9 s_{10} s_8 s_9 \rangle$.

The match scores of the hand-segmented test gestures were then computed using the dynamic programming algorithm outlined in Section 3.4. In computing the maximum match score, the algorithm assigns each point to a state consistent with the sequence G_α. As described, a match is made only if it is considered complete. The state transitions and the membership values computed for each sample are shown for the new examples of G_α in Figure 3-3. The representation thus provides a convenient way to specify a gesture as an ordered sequence of states while also permitting the combination of states shared by multiple gestures.

3.5.2 Position and Orientation Sensor Gestures

For the second experiment, we compute the representation with somewhat sparse, higher dimensional data. We show its use in the automatic, causal segmentation of two different gestures as if the samples were collected in a real time recognition application.

An Ascension Technology Flock of Birds magnetic position and orientation sensor was worn on the back of the hand and polled at 20 Hz. For each sample, the position of the hand and the normal vector out of the palm of the hand was recorded (six dimensions). Ten large wave gestures (about 40 samples each) and ten pointing gestures (about 70 samples each) were collected.

To insure that there were enough points available to compute the prototype curve, each

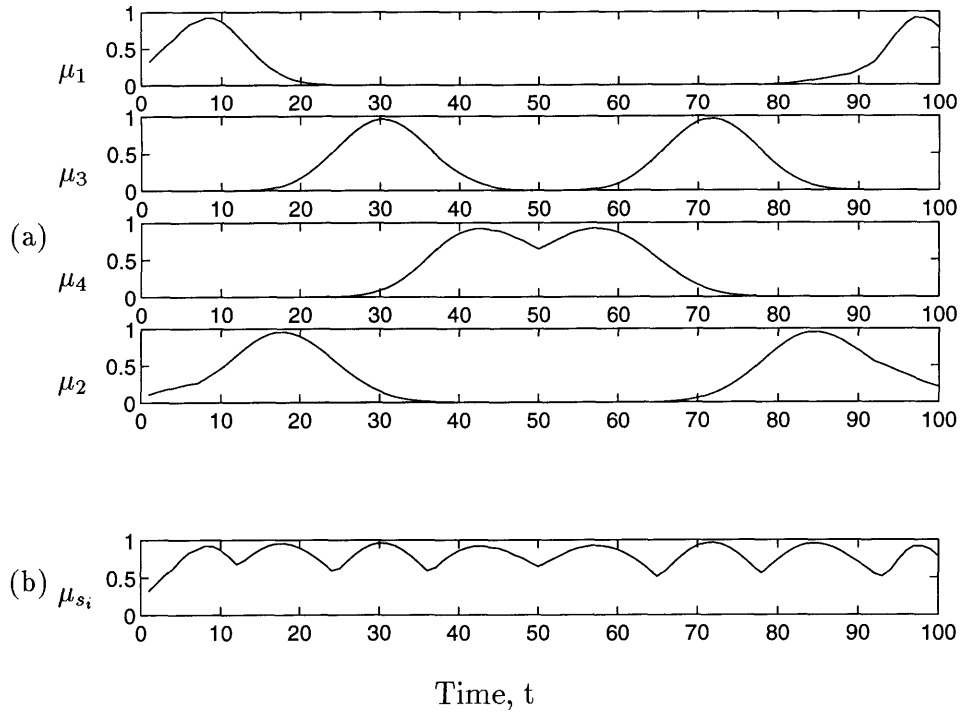


Figure 3-4: (a) The membership plot for the prototype curves for G_{wave} show for each of the configuration states how the states lie along the prototype. (b) Combined membership (maximum) of all states at each point along the prototype. The prototype for G_{point} is similar.

example was up-sampled using Catmull-Rom [6] splines so that each wave gesture is about 40 samples and each point gesture about 70 samples.

The prototype curves for each gesture were computed separately. The membership plots for the prototype wave is shown for each state in Figure 3-4. The gesture sequence G_{wave} is found by analyzing the combined membership function (Figure 3-4b): $G_{wave} = \langle s_1 s_2 s_3 s_4 s_3 s_2 s_1 \rangle$. Similarly, G_{point} (not shown) is defined by $\langle s_5 s_6 s_7 s_8 s_7 s_6 s_5 \rangle$. Note how both the sequences and the plots capture the similarity between the initiation and retraction phases of the gesture.

The state transition and membership plots as calculated by the dynamic programming algorithm are shown for all the example wave gestures in Figure 3-5. Because the example gestures started at slightly different spatial positions, it was necessary to ignore the first and last states in the calculation to obtain good matches. This situation can also occur, for example, due to gesture co-articulation effects that were not observed during training.

A stream of samples consisting of an alternating sequence of all the example wave and point gestures was causally segmented to find all wave and point gestures. For a matching threshold of $\eta = 0.5$, all the examples were correctly segmented.

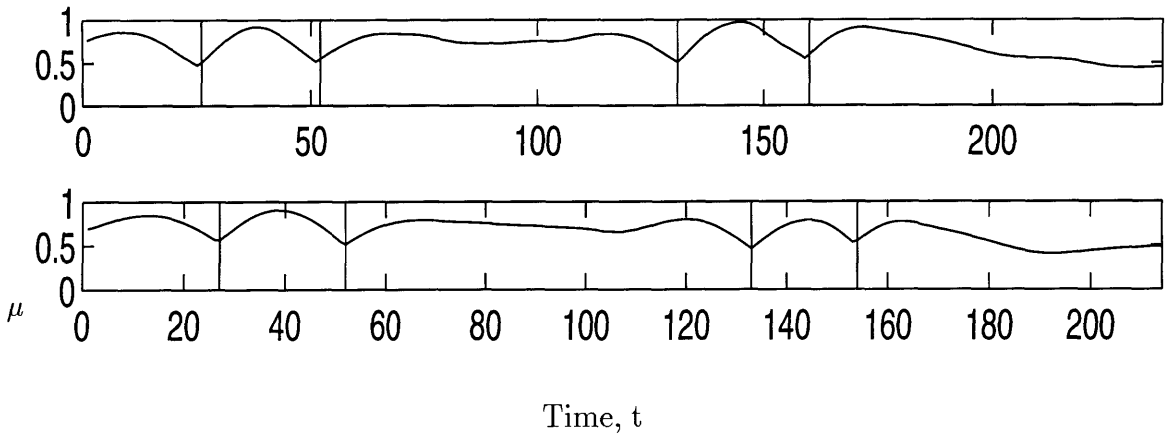


Figure 3-5: The state transition and membership plots for 2 of the 10 testing examples for G_{wave}. The state transitions (vertical bars) were successfully tracked in all 10 wave gestures and 9 of 10 point gestures.

Even with sparse and high dimensional data, the representation is capable of determining segmentation. Additionally, the representation provides a useful way to visualize the tracking process in a high dimensional configuration space.

3.5.3 Image-space Gestures

As a final experiment, we consider a digitized image sequence of a waving hand (Figure 3-6). Given a set of smoothly varying measurements taken from the images and a few hand-segmented example waves, the goal is to automatically segment the sequence to recover the remaining waves. This example shows how the measurements do not require a direct physical or geometric interpretation, but should vary smoothly in a meaningful and regular way.

Each frame of the sequence is a point in a 4800-dimensional space of pixel values, or image-space. If the motion is smooth so that the images vary slowly the sequence will trace a smooth path in image-space. Rather than approximate trajectories in the 4800-dimensional space, we instead approximate the trajectories of the coefficients of projection onto the first few eigenvectors computed from a part of the sequence.

The first five example waves were used in training. The three eigenvectors with the largest eigenvalues were computed by the Karhunen-Loeve Transform (as in [59, 35]) of the training frames, treating each frame as a column of pixel values. The first three eigenvectors accounted for 71% of the variance of the pixel intensity values of the training frames.

The training frames were then projected onto the eigenvectors to give the smooth tra-

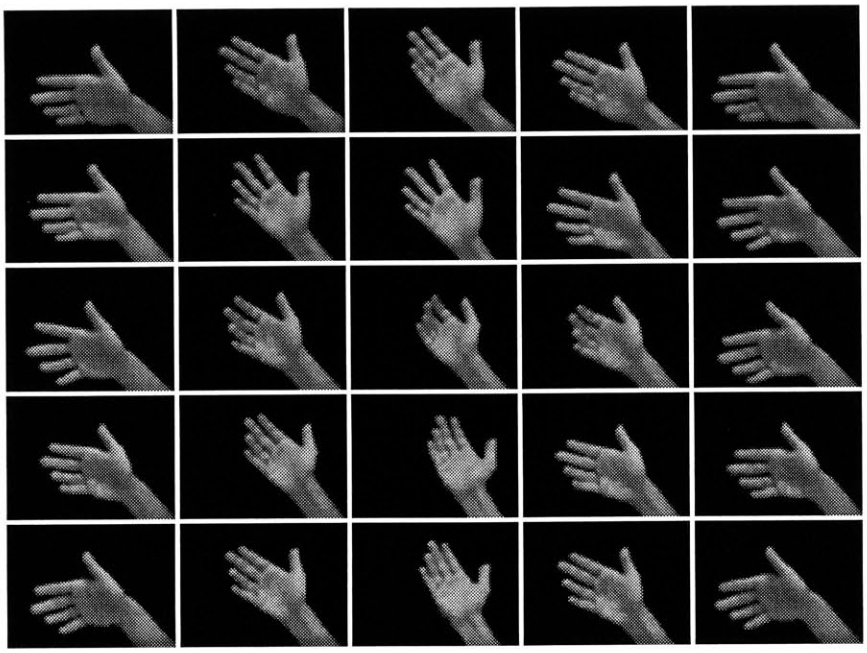


Figure 3-6: Each row of images depicts a complete wave sequence taken from the larger image sequence of 830 frames, 30 fps, 60 by 80 pixels each. Only 5 frames of each sequence is presented. The variation in appearance between each example (along columns) is typical of the entire sequence.

jectories shown in Figure 3-7. The recovered state sequence $G_{wave} = \langle s_1 s_4 s_3 s_2 s_3 s_4 \rangle$ again shows the periodicity of the motion. The state transitions and membership values were computed for ten other examples projected onto the same eigenvectors. Again, the first and last states were ignored in the matching process due to variations in the exact beginning and ending of the motion. One of the examples was deemed incomplete. Lastly the automatic causal segmentation of the whole image sequence was computed. Of the 32 waves in the sequence, all but one (the same incomplete example above) were correctly segmented.

3.6 Summary

A novel technique for computing a representation for gesture that is time-invariant but order-preserving has been presented. The technique proceeds by computing a prototype gesture of a given set of example gestures. The prototype preserves the temporal ordering of the samples along each gesture, but lies in a measurement space without time. The prototype offers a convenient arc length parameterization of the data points, which is then used to calculate a sequence of states along the prototype. The shape of the states is calculated to capture the variance of the training gestures. A gesture is then defined as an

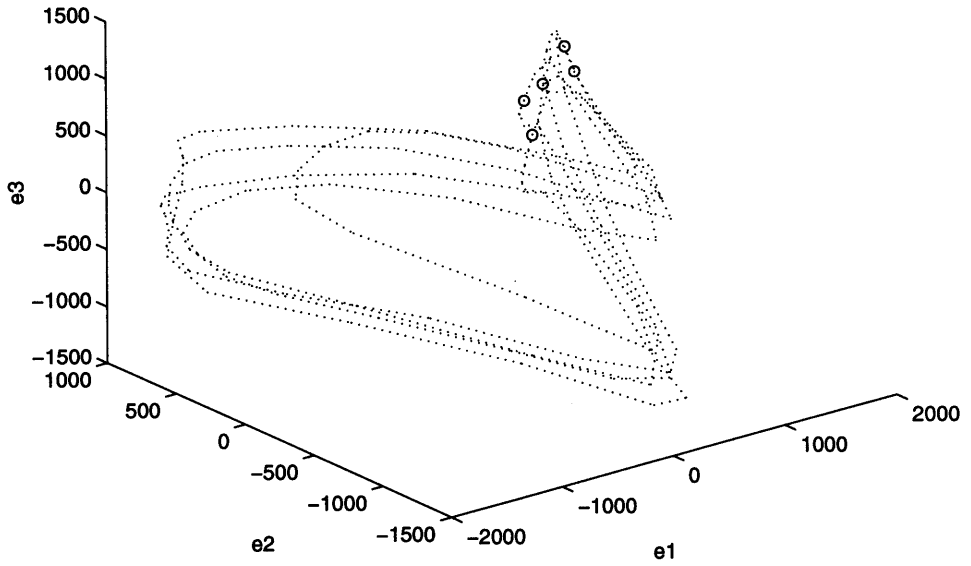


Figure 3-7: Each of the axes e_1, e_2, and e_3 represents the projection onto each of the three eigenvectors. The image sequences for the five training examples project to the smooth trajectories shown here. A circle represents the end of one example and the start of the next. The trajectory plots show the coherence of the examples as a group, as well as the periodicity of the movement.

ordered sequence of states along the prototype.

A technique based on dynamic programming uses the representation to compute a match score for new examples of the gesture. A new movement matches the gesture if it has high average combined membership for the states in the gesture, and it passes through all the gestures in the sequence (it is complete). This recognition technique can be used to perform a causal segmentation of a stream of new samples. Because the particular form of the dynamic programming algorithm we use can be implemented to run efficiently, the causal segmentation with the representation could be useful in a real time gesture recognition application.

Lastly, three experiments were conducted, each taking data from devices that are typical of gesture recognition applications. The variety of inputs addressed demonstrates the general utility of the technique. In fact, our intuition is that there are only a few requirements on the measurements for the technique to be useful; we would like to make these requirements more explicit.

The representation of a gesture as a sequence of predefined states along a prototype gesture is a convenient symbolic description, where each state is a symbol. In one experi-

ment we demonstrated how two gestures can share states in their defining sequences. This description may also be useful in composing new gestures from previously defined ones, in detecting and allowing for periodicity in gesture, and in computing groups of states that are atomic with respect to a number of gestures. In short, the symbolic description permits a level of "understanding" of gesture that we have not explored.

There seems to be little consensus in the literature on a useful definition of "gesture". Part of the problem in arriving at a concise notion of gesture is the broad applicability of gesture recognition, and the difficulty in reasoning about gesture without respect to a particular domain (e.g., hand gestures). The development of the configuration state technique presented is an attempt to formalize the notion of gesture without limiting its applicability to a particular domain. That is, we wish to find what distinguishes gesture from the larger background of all motion, and incorporate that knowledge into a representation.

Chapter 4

Learning Visual Behavior for Gesture Analysis

4.1 Introduction: From human motion to gesture

For all the degrees of freedom available to the human body, we seem to habitually use a only small class of motions that they permit. Even athletes, which as a group use their bodies in ways that most people do not, aspire to repeat motions flawlessly, spending hours upon hours practicing the same motion. In the space of motions allowed by the body's degrees of freedom, there is a subspace that most of us use. For example, if the body is modeled by a series of joints and angles between them, there would be many combinations of joint angles that we would never see.[1]

Gesture is one interesting subspace of human motion. Provisionally, we define gesture to be motions of the body that are intended to communicate to another agent. Therefore, the gesturer and the recipient of the gesture must share a knowledge of gesture to communicate effectively. By way of simplification, this means the gesturer's movement must be one of a predefined set. We do not mean that a given gesture has a fixed geometry; a "wave" might be a single gesture in which the hand can be at any height between the chest and the top of the head. Rather, for now we are assuming that there is a set of gestures and each gesture defines a range of motions that are to be interpreted as being examples of that gesture. Thus gestures are *modal* in the space of possible human motion.

[1] The body of this chapter has been submitted in abbreviated form to the IEEE International Symposium on Computer Vision [63].

Due to the variability of human movement, the behavior of the gesture must be described without regards to precise geometry or precise temporal information. We take *visual behavior* to mean the sequence of visual events that makes a complete action or gesture. We assume that two gestures that have the same visual behavior are in fact the same gesture, thus ignoring the delicate problem of relating the form and meaning of natural gesture [31].

4.1.1 View-based approach

The machine understanding of human movement and gesture brings new possibilities to computer-human interaction. Such interest has inspired research into the recovery of the the complete 3-dimensional pose of the body or hand using a 3-dimensional physical model (e.g. [50, 51]). The presumption behind such work is that a complete kinematic model of the body will be required for useful inferences.

We claim that gestures are embedded within communication. As such, the gesturer typically orients the movements towards the recipient of the gesture. Visual gestures are therefore *viewpoint-dependent.* And the task of gesture understanding is particularly suited to a view-based, multiple model approach in which only a small subspace of human motions is represented.

4.2 Representation of gesture

4.2.1 Multiple models for gesture

We claim that gestures are modal in the space of human motion. But how should a system model human motion to capture the constraints present in the gestures? There may be no single set of features that makes explicit the relationships that hold for a given gesture. In the case of hand gestures, for example, the spatial configuration of the hand may be important (as in a point gesture, when the observer must notice a particular pose of the hand), or alternatively, the gross motion of the hand may be important (as in a friendly wave across the quad). Quek [42] has observed that it is rare for both the pose and the position of the hand to simultaneously change in a meaningful way during a gesture.

Rather than use one model that is only partially effective, the approach here is to allow for multiple models. By *model* we mean a systematic way to describe a set of existing sensor data and a method to determine if it also describes new sensor data. Different models may

interpret the same sensor data in different ways or they may take data from different sensors, in which case *sensor fusion* is the goal. The use of multiple models in a visual classification task is discussed in [39].

One goal is to develop an approach that can exploit multiple models simultaneously, where the type of models might be quite distinct. Model types useful for characterizing images in an image sequence might include eigenvector decomposition of sets of images [59], orientation histograms [18], peak temporal frequencies[41], tracked position of objects in the frame, and optic flow field summaries.

4.2.2 State-based descriptions

In Chapter 3 we defined gesture to be a sequence of states in a configuration space. States were defined on some input space (say the joint angles derived from a DataGlove) and were designed to capture the constraints present in a series of training examples. Membership in a state was governed by probabilistic functions that attempted to capture the natural variability of motion in terms of the variances of these functions.

The temporal aspect of a gesture was incorporated by requiring that the states be defined along a prototype derived from training examples. Once defined, these states would be used to determine if a particular trajectory through the input space was an example of the learned gesture: the trajectory had to sequentially pass through each state attaining sufficient memberships in sequence. The actual time course was not important as long as the sequence was appropriate.

In the work presented here, we continue to consider a gesture as a sequence of states. At each point in time, the observed visual input reflects the current, unobservable state and perhaps the transition to the next state. This state-based description is easily extended to accommodate multiple models for the representation of different gestures or even different phases of the same gesture. The basic idea is that the different models need to approximate the (small) subspace associated with a particular state. Membership in a state is determined by how well the state models can represent the current observation.

4.2.3 Learning visual behavior

In this paper we develop a technique for learning visual behaviors that 1) incorporates the notion of multiple models; 2) makes explicit the idea that a given phase of a gesture is

constrained to be within some small subspace of possible human motions; and 3) represents time in a more probabilistic manner than defined by a prototype approach. In the remaining sections we first derive a state model and membership function based upon *residual*, or how well a given model can represent the current sensor input. We then embed this residual-based technique within a Hidden Markov Model framework; the HMM's represent the temporal aspect of the gestures in a probabilistic manner and provide an implicit form of dynamic time warping for the recognition of gesture. Finally, we demonstrate the technique on several examples of gesture and discuss possible recognition and coding applications.

4.3 Modeling gestures

4.3.1 Model instances and memberships

Suppose we have a set of observations $O = O_1, O_2, \ldots, O_T$ and a collection of states numbered $j = 1 \ldots N$. Assume that for each observation O_t we are given a degree of belief $\gamma_t(j)$ that O_t belongs to a state j; we require that $\sum_{j=1}^{N} \gamma_t(j) = 1$. We can interpret $\gamma_t(j)$ as the *membership* of O_t to state j. In this section we show the combination of multiple models in order to describe the set of observations belonging to a single state; in the next section we consider modeling the time course of states.

A number of *model types* A, B, etc. are selected *a priori* to describe the observations. Each state is then associated with one instance of each model type. Each *model instance* is defined by the set of parameters that limit the model type to match some set of observations. For example, eigenvector decomposition of images may be a model type. An instance of the model type would comprise a particular mean image and set of eigenvectors (eigenimages) that are computed from a set of examples. Let us denote the set of model instances at state j as $\mathcal{M}_j = \{A_j, B_j, \ldots\}$.

The parameters of $\mathcal{M}_j$ are computed from the set of example observations with the highest membership $\gamma_t(j)$. This may be accomplished by weighting each of the examples in the computation of the parameters, or simply by selecting some fixed number of the top observations, ranked by $\gamma_t(j)$. In the examples presented, the latter approach is taken.

For each model instance $m \in \mathcal{M}_j$ and an observation $\mathbf{x}$ we can compute a distance $d_m(\mathbf{x})$ which measures the degree to which the model instance m is unable to match $\mathbf{x}$. In this sense, $d_m(\mathbf{x})$ is a *reconstruction error* or *residual*. If we think of the parameters of $\mathcal{M}_j$

as limiting the model to a subspace of the space of samples, then we may also call $d_m(\mathbf{x})$ a *distance to model subspace.* The distances to each model instance may be combined to give

$$\mathbf{d}_j(\mathbf{x}) = \langle d_{A_j}(\mathbf{x}), d_{B_j}(\mathbf{x}), \ldots \rangle.$$

This quantity is similar to the "distance from feature space" derived in [33].

Next we consider the observation probability distribution $b_j(\mathbf{x})$ which describes the probability of measuring a particular residual for an observation when that observation is really generated by state j. $b_j(\mathbf{x})$ may be estimated from the observations, each weighted by $\gamma_t(j)$. Assuming the independence of each model, we estimate b_j as a normal[2] joint distribution on $\mathbf{d}_j$: $b_j(\mathbf{x}) = \mathcal{N}[\mathbf{d}_j(\mathbf{x}), \mu_j, \Sigma_j]$, with

$$\mu_j = \sum_{t=1}^{T} \frac{\gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \mathbf{d}_j(O_t),$$

and

$$\Sigma_j = \sum_{t=1}^{T} \frac{\gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} (\mathbf{d}_j(O_t) - \mu_j)(\mathbf{d}_j(O_t) - \mu_j)^T.$$

The probability $b_j(\mathbf{x})$ may then be used in the computation of a new $\gamma_t(j)$. One approach to computing $\gamma_t(j)$ that exploits the fact that the observations are not simply a set but a sequence is that of Hidden Markov Models, presented in the next section as a technique for describing transitions between states.

Having updated $\gamma_t(j)$, the estimation of the model instances $\mathcal{M}_j$ described above is iterated. In this way the memberships $\gamma_t(j)$ and the model instances are tuned to define the states in a way that best represents the observations.

Summarizing, we list the following requirements of model instances:

- Model instance parameters are computed using the observations and their membership to the state, $\gamma_t(j)$.
- Each model instance delineates some subspace of the space of observations.
- The distances $\mathbf{d}_j(\mathbf{x})$ must be lowest for observations with a high degree of membership $\gamma_t(j)$.

[2]Since there are no negative distances, the gamma distribution may be more appropriate.

4.3.2 HMM's with multiple independent model subspaces

Following a trend in speech recognition, vision researchers have applied the Hidden Markov Model technique to gesture recognition. Yamato et al. [64] compute a simple region-based statistic from each frame of image sequences of tennis swings. Sequences of the vector-quantized features are then identified by a trained HMM. Schlenzig et al. [53] use a rotation invariant representation of binary images and a neural net to quantize the image to a hand pose before using an HMM. Starner and Pentland [56] apply continuous HMM's with the orientation and position of both hands wearing colored gloves.

HMM's are attractive because they put dynamic time warping on a probabilistic foundation and produce a Markov model of discrete states that codes the temporal structure of the observations [24]. Training an HMM involves inferring a first-order Markov model from a set of possibly continuous observation vectors. Each state is associated with the observation probability distribution $b_j(\mathbf{x})$. The probability of making a transition from a state i to a state j in one time step is denoted as $A(i,j)$. The relationship between the discrete states and $b_j(\mathbf{x})$ is depicted in Figure 4-1(a).

$\gamma_t(j)$, the probability of being in state j at time t given the observation sequence O and the HMM, is computed by the "forward-backward" procedure. $\gamma_t(j)$ is subsequently used by the Baum-Welch algorithm to iteratively adjust $b_j(\mathbf{x})$ and A until the probability of the HMM generating the observations is maximized.

Training the HMM with multiple independent model subspaces proceeds by interleaving iterations of the Baum-Welch algorithm (giving an updated $\gamma_t(j)$ to reflect the updated Markov model A) with reestimating the parameters of the model instances. In this way the representation encoded at each state is trained concurrently with the transition model. The relationship between each discrete state of the Markov model and the *multiple independent model subspaces* is depicted in Figure 4-1(b).

An advantage to this approach is that the representation at each state is designed to match the particular temporal model, and the temporal model is designed for the particular choice of representations as well. Also, by having multiple independent models, we do not rely on any one particular model instance to fit all observations for all states.

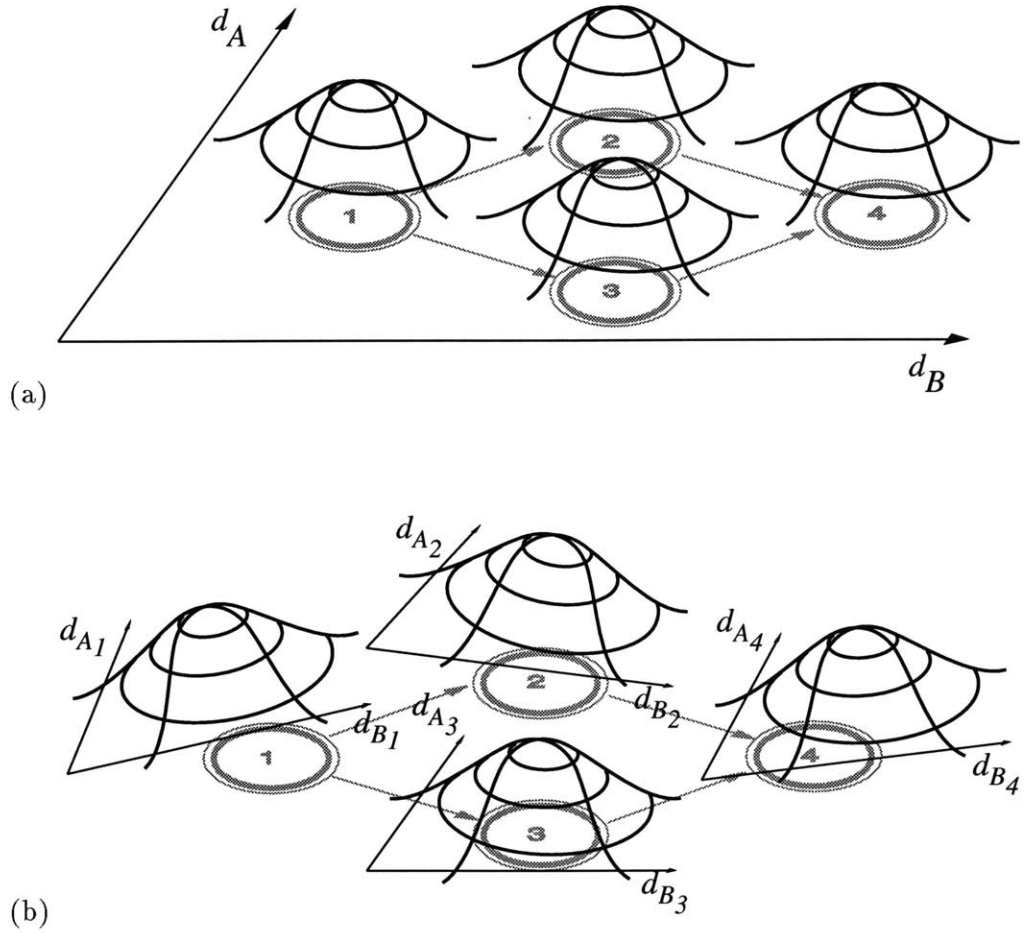


Figure 4-1: Each state of the Markov model (gray) is associated with a unimodal observation pdf. (a) In the conventional HMM framework all observation distributions reside in the same space of measurements from model A and B. (b) In the multiple independent model subspace HMM, each state is associated with an independent space of measurements from model A_j and B_j.

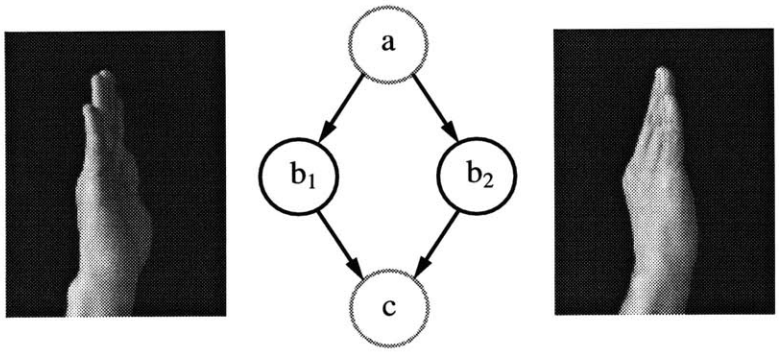


Figure 4-2: A state-based description of gesture must encode the relevant perceptual states. These images of an upright open hand share the same conceptual description, but have very different perceptual descriptions due to a slight change in viewing angle.

4.3.3 HMM topology

Before training the HMM, initial transition probabilities $A(i,j)$ must be provided. The topology of the resulting Markov model is constrained by initially setting some $A(i,j)$ to zero. To ease training, the topology is commonly constrained to be simple (e.g. causal).

The topology of the Markov model has the capability of encoding the temporal structure of the gesture. We choose not to restrict the topology of the Markov model initially and instead recover the topology through training. The reasons for doing so are twofold. First, by not providing a strict initial model we may recover interesting temporal structures that would otherwise escape notice, such as periodicity. In such cases the structure of the recovered transition matrix contributes to our understanding of the gesture.

Second, by providing a strict initial model we make implicit assumptions about the distribution of the sensor outputs (e.g., unimodal along the gesture in the case of a strictly linear Markov model). These assumptions may be unwarranted: while a simple gesture may seem to us a simple sequence of *conceptual states*, the sensors may see the movement as a complicated tangle of *perceptual states*. This may occur, for example, when the sensors used do not embody the same invariances as our own visual system. Figure 4-2 illustrates a single conceptual state (the upright hand) generating grossly different observations. If a single $b_j(\mathbf{x})$ cannot encode both observations equally well, then additional Markov states are required to span the single conceptual state. The addition of these states require the flexibility of the Markov model to deviate from strictly causal topologies.

Initialization:
  set $A(i,j) = \frac{1}{N}$ for all i, j
  initialize $\gamma_t(j)$ randomly, $\sum_{j=1}^{N} \gamma_t(j) = 1$

Algorithm:
  repeat until parameters of $\mathcal{M}_j$ do not change:
    for each state j (parallelizable):
      estimate parameters to models $m \in \mathcal{M}_j$
      compute $\mathbf{d}_j(\mathbf{x})$ for all $\mathbf{x} \in O$
      estimate $b_j(\mathbf{x}) = \mathcal{N}[\mathbf{d}_j(\mathbf{x}), \mu_j, \Sigma_j]$
    end
    update A, $\gamma_t(j)$ by Baum-Welch algorithm
  end

Figure 4-3: Training algorithm

4.3.4 Algorithm

To recover the temporal structure of the gesture and to train the representations at each state to suit the temporal model, we initialize the iterative algorithm sketched above with a uniform transition matrix and a random membership for each observation (Figure 4-3).

In the conventional HMM framework, the Baum-Welch algorithm is guaranteed to converge [24]. By interleaving the estimation of model parameters with the Baum-Welch algorithm, the proof of convergence may be invalidated. However, convergence has not been a problem with the examples tried thus far.

4.4 Examples

4.4.1 Single model

For our first examples, we use the eigenvector decomposition of the image as the single model type. In this case, the parameters associated with a model instance are simply a number of the top eigenimages that account for most of the variance of the training images (as indicated by the eigenvalues) and the mean image of the training images. The training images for a model instance at state j are selected by ranking all samples by $\gamma_t(j)$ and selecting some number of the top samples. Given a model instance E_j and a sample image $\mathbf{x}$, $\mathbf{d}_j(\mathbf{x}) = \langle d_{E_j}(\mathbf{x}) \rangle$ is simply the reconstruction residual (average per pixel) of the image using the precomputed eigenvectors at state j.

The first example is a waving hand against a black background. The second is a twisting hand (as in "turn it up", or "CRANK IT!") [3] against a cluttered background. The last single model example is of the whole body doing jumping-jacks, facing the camera.

For each example, the input consists of about 30 image sequences, each about 25 frames (60 by 80 pixels, grayscale) in length. The top 50 $\gamma_t(j)$-ranked sample images were used, and the number of eigenvectors was chosen to account for 70% of the variance of the selected sample images. The recovered Markov model, mean image at each state and a plot of $\gamma_t(j)$ and d_{E_j} for one training sequence are shown in Figures 4-4, 4-5, and 4-6.

Consider the waving hand example: the recovered Markov model permits the periodicity shown by the plot of $\gamma_t(j)$ over an observation sequence. Some other observation sequences differ in the extent of the wave motion; in these cases the state representing the hand at its lowest or highest position in the frame is not used. The plot of $\gamma_t(j)$ reveals the time warping permitted by the Markov model. For example, the hand must decelerate to stop at the top of the wave, and then accelerate to continue. This is shown by the longer duration of membership to the first (top) state shown in the figure.

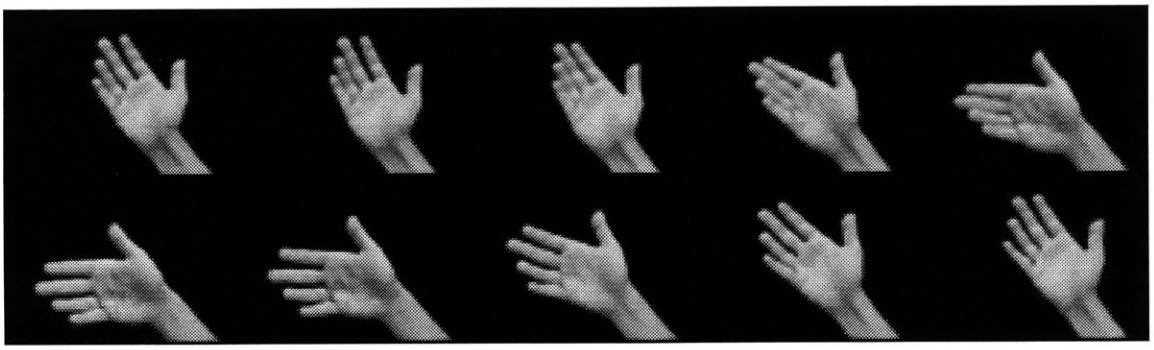
Note that in second and third of these examples, foreground segmentation is unnecessary since the camera is stationary. The mean image corresponds to the static parts of the image; the eigenvector decomposition automatically subtracts the mean image at each state.

4.4.2 Position and configuration

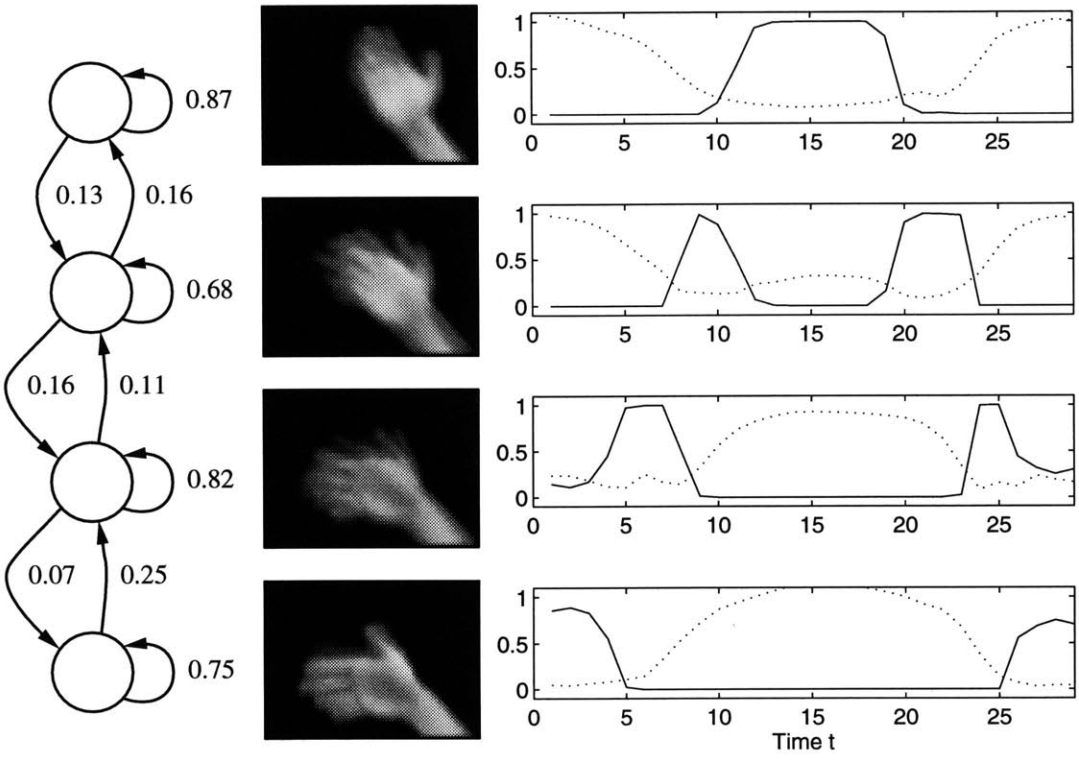
The second example describes the position and configuration of a waving, pointing hand. In each frame of the training sequences, a 50 by 50 pixel image of the hand was tracked and clipped from a larger image with a cluttered background. Foreground segmentation was accomplished using the known background. The configuration C of the hand is modeled by the eigenvector decomposition of the 50 by 50 images. The position P of the hand is modeled by the location of the tracked hand within the larger image; at each state P is estimated as the $\gamma_t(j)$-weighted mean location.

The recovered Markov model is similar to that of the waving hand in the previous example. The mean images and $\mathbf{d}_j = \langle d_{C_j}, d_{P_j} \rangle$ for each state are shown in Figure 4-7.

[3] One interesting experimental question is whether the system is able to distinguish "turn it up" from "turn it down". Presumably, the recovered Markov models would be different for each of these motions. See [18] for the application of gesture recognition to television control.

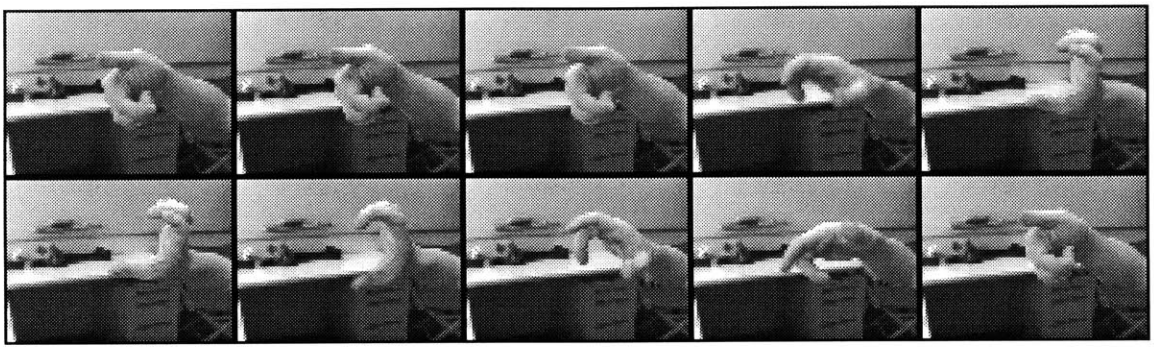


(a)

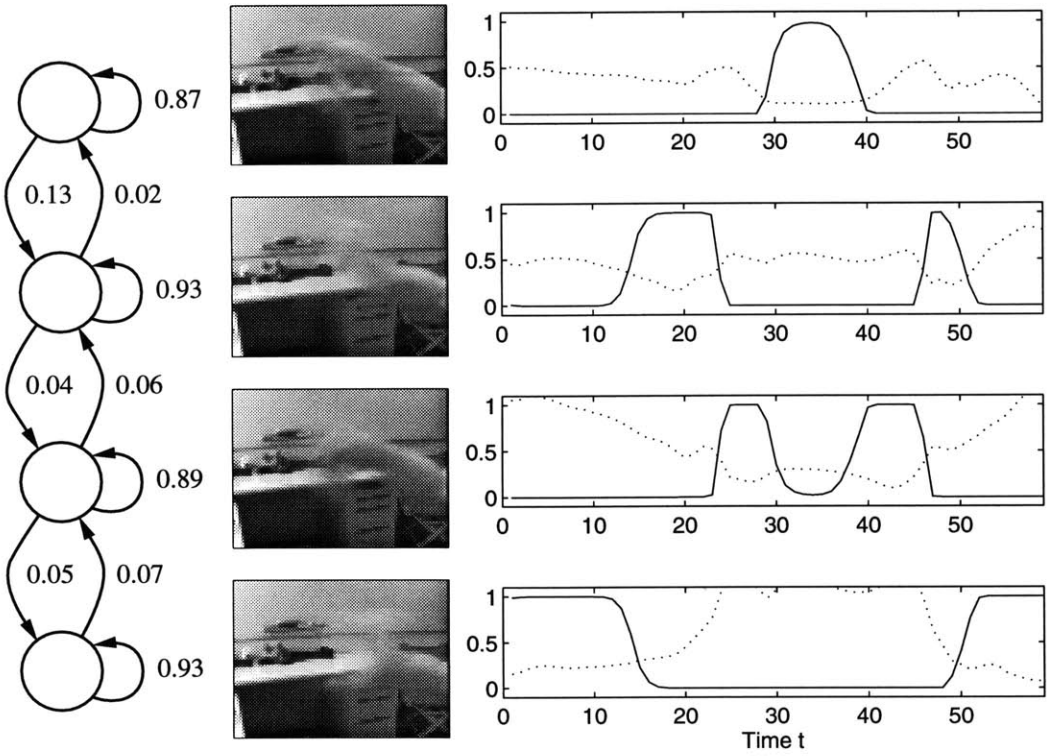


(b)

Figure 4-4: (a) An example wave sequence. (b) The recovered Markov model for all training sequences at left shows the periodicity of the gesture. The $\gamma_t(j)$-weighted mean image for each state is shown in the middle. On the right is a plot of $\gamma_t(j)$ (solid line) and d_{E_j} (dotted line) for each state for one training sequence. The exact shape of the plots varies in response to the variance and length of the sequence. (The plots of d_{E_j} were scaled to fit.)



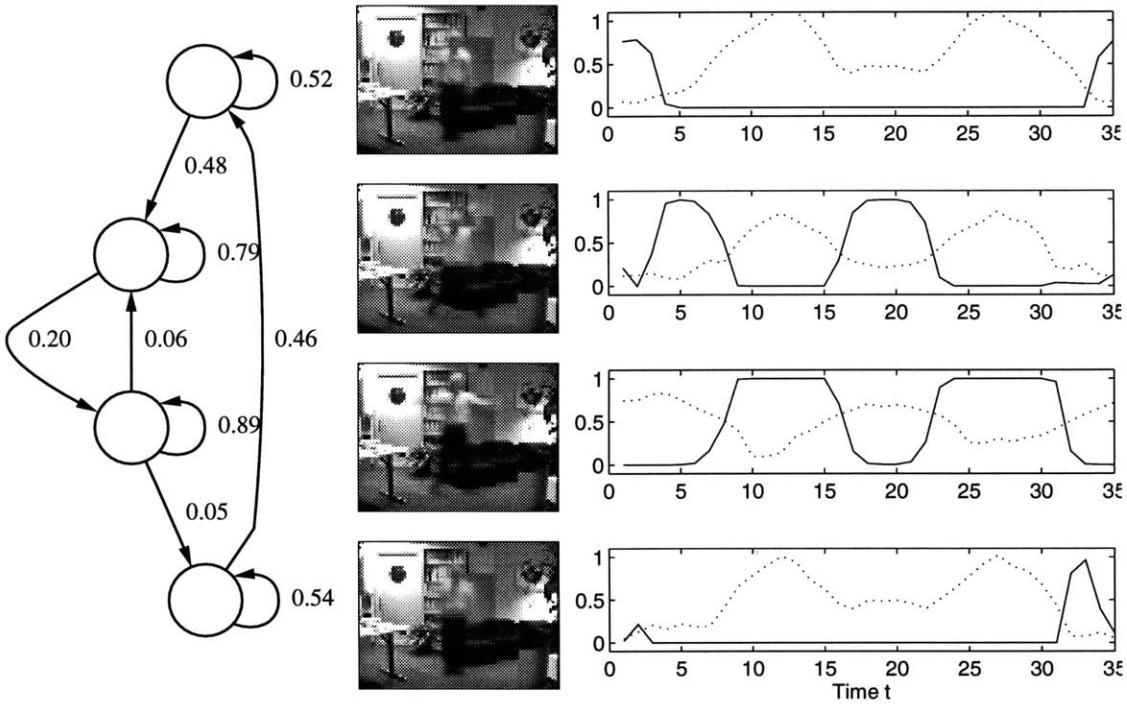
(a)



(b)

Figure 4-5: (a) An example sequence of a twisting hand ("turn it up!"). (b) Recovered Markov model, mean images at each state, plots of $\gamma_t(j)$ (solid line), and d_{E_j} (dotted line) from the sequence above are shown.

(a)



(b)

Figure 4-6: (a) An example jumping-jack. (b) Recovered Markov model, mean images at each state, plots of $\gamma_t(j)$ (solid line), d_{E_j} (dotted line) from the sequence above are shown.

The variance of each feature indicates the importance of the feature in describing the gesture. In this example both the position and configuration of the hand was relevant in describing the gesture. Had the location of the hand varied greatly in the training set, the high variance of the position representation would have indicated that position was not important in describing the gesture.

4.4.3 Two camera views

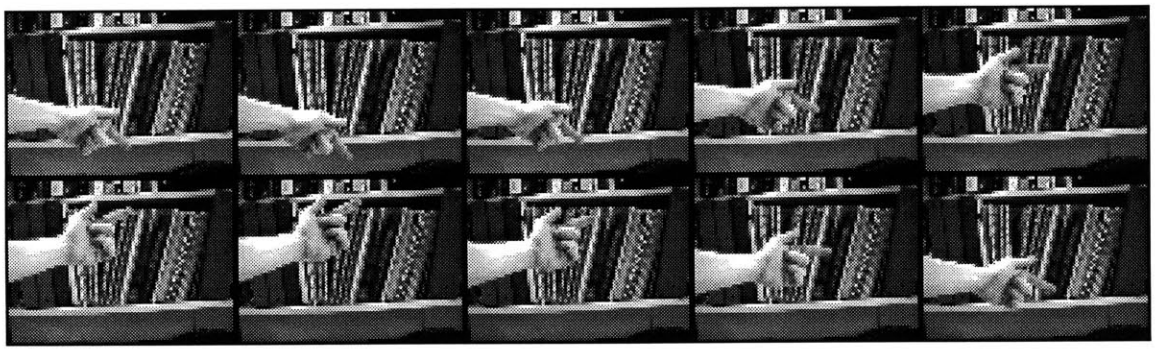
The final example shows how models for two camera views may be combined. Two views may be useful in describing the simultaneous movement of multiple objects (e.g. two hands, or a face and hands), or in describing gestures that exhibit movement in depth. Fifteen examples of a "pushing" gesture (see Figure 4-8(a)) were taken from both a side and front view. Eigenvector decomposition was used to model each view; $\mathbf{d}_j = \langle d_{E_j^{\text{front}}}, d_{E_j^{\text{side}}} \rangle$. The mean image for each view and plots of $\mathbf{d}_j$ are shown in Figure 4-8(b).

Again, because the background is stationary in this example, foreground segmentation is accomplished simply by the subtraction of the mean image from each frame of the sequence.

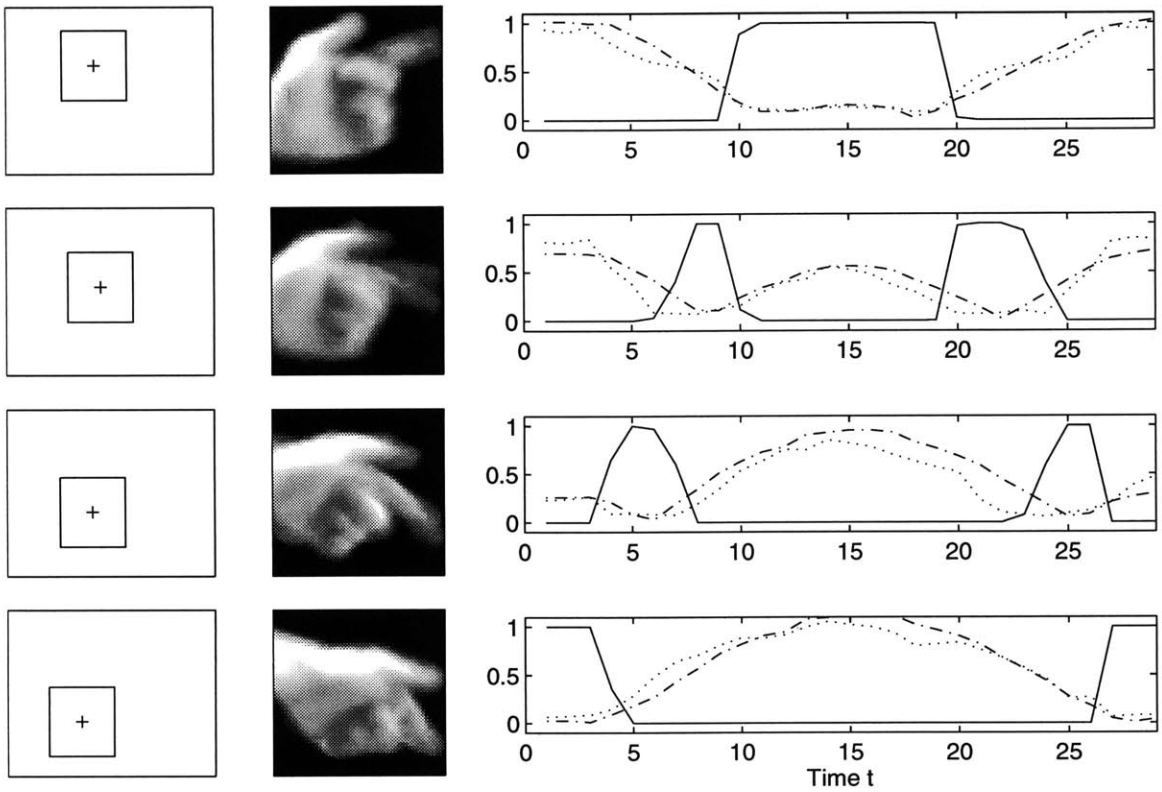
4.5 Representation vs. Recognition

In assigning parameters to model instances so that the models fit the training data well, the models are tuned to represent the training data as best they can. In the case of the eigenvector decomposition of the images, the model represents the training images in an "optimal" way (given some number of eigenvectors). As a series of states, each of which seeks to best represent its associated training images, the multiple model HMM in some sense seeks to represent the training sequences as best it can, both by seeking good representations at each state, and by providing an HMM topology and time warping that covers the time course of the examples as best it can. Subject to the problem of local-minima during training, it would seem that the multiple model HMM is an "optimal" representation of the sequence in some sense, given a certain number of states, model parameters to record, and the first-order limitation on the Markov model.

But a good representation is not necessarily the best for recognition. For example, representations for very similar objects may be identical due to the model limitations, in which case attempts to distinguish one object from the other are hopeless. But there is

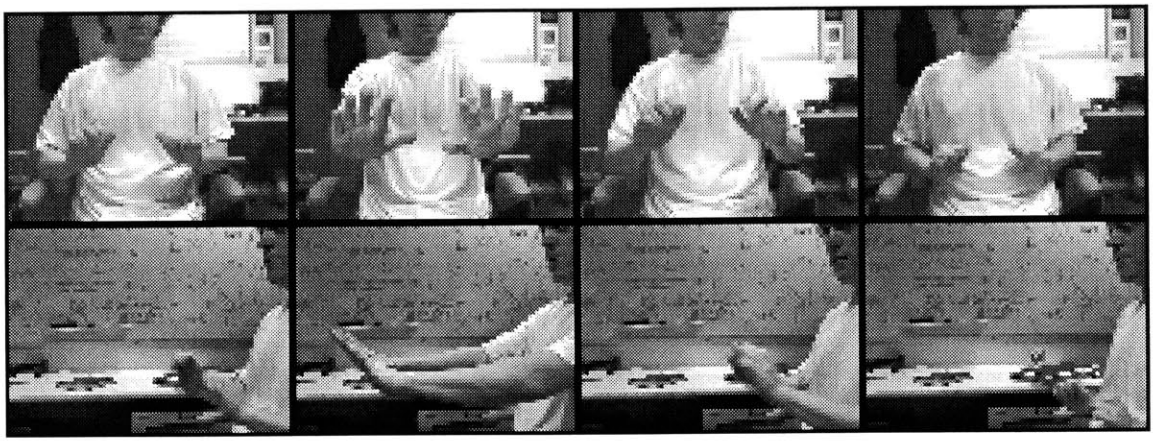


(a)

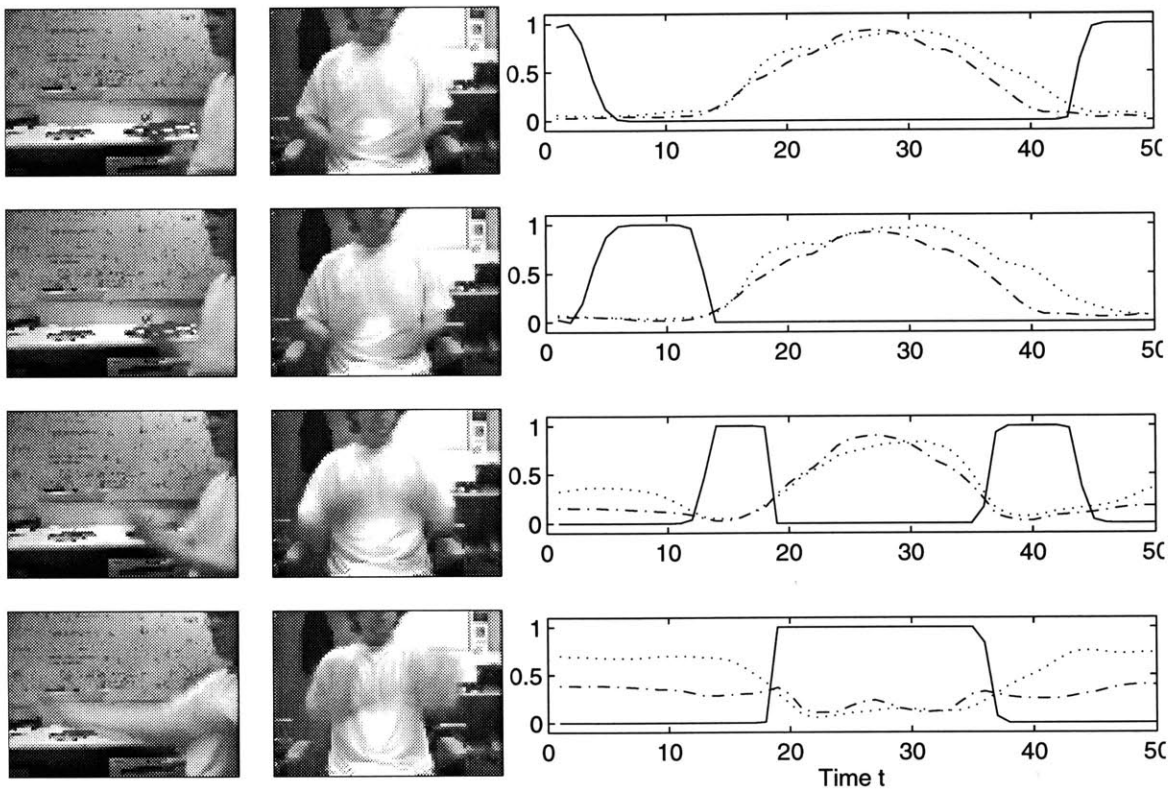


(b)

Figure 4-7: (a) An example pointing wave sequence. (b) The $\gamma_t(j)$-weighted mean location of the tracked hand in the larger image is shown on the left. The mean image for each state is shown in the middle. On the right is a plot of $\gamma_t(j)$ (solid line), d_{C_j} (dotted line), and d_{P_j} (dash-dotted line) for each state for one training sequence. (d_{C_j} and d_{P_j} were scaled to fit.)



(a)



(b)

Figure 4-8: (a) Four representative frames (ordered left to right) are shown from each view of one training "push" gesture. (b) The mean images for both the side view and front view at each state are shown on the left. Plots of $\gamma_t(j)$, $d_{E_j^{\text{front}}}$ (dotted line) and $d_{E_j^{\text{side}}}$ (dash-dotted line) are from one training sequence.

some feature that could be used to distinguish the objects (or they should be labeled as the same object). The problem of building a recognition system is finding the features that distinguish the objects: systems that perform well in recognition tasks perform well because they are designed to.

The multiple model HMM is then suited for a recognition task insofar as a good representation in most cases is good enough for recognition. By virtue of having encoded a good representation, the multiple model HMM is more particularly suited for image coding tasks. For example, in a virtual teleconferencing application, each station might only transmit a symbol identifying the current state seen by the vision system. At this stage, efficient real time coding of the images is necessary. The receiving station then matches symbols to a stored library of model instance parameters and renders an appropriate image. The library of states stored at the receiving station would be computed off-line beforehand by training the multiple model HMM's on previously seen sequences. We believe the real time image coding task is a promising avenue of research for the multiple model HMM framework.

4.6 Summary

A learning-based method for coding visual behavior from image sequences was presented. The technique is novel in its incorporation of multiple models into the well-known Hidden Markov Model framework, so that models representing what is seen are trained concurrently with the temporal model. Multiple model instances are defined at each state to characterize observations at the state. The membership of a new observation to the state is determined by computing a distance measure to each of the *model subspaces* approximated by the model instances. These distance measures are then combined to produce a probability distribution on the observations at the state.

The model instances themselves are trained concurrently with the temporal model by interleaving the estimation of model parameters with the usual Baum-Welch algorithm for updating the Markov model associated with the HMM. Thus the temporal model and the model instances at each state are computed to match one another.

We exploit two constraints allowing application of the technique to view-based gesture recognition: gestures are modal in the space of possible human motion, and gestures are viewpoint-dependent. The examples in Section 4.4 demonstrate that with a practical num-

ber of low resolution image sequences and weak initial conditions, the algorithm is able to recover the visual behavior of simple gestures using a small number of low resolution example image sequences. Future implementations may be useful in novel wireless computer-human interfaces, real time "object-based" coding from video, and studying the relevant features of human gesture.

Chapter 5

Work in Progress: Multiple Gestures

5.1 Introduction

Currently, we are advancing the framework to handle the concurrent training of multiple gestures. One goal of the work presented in the previous chapter is the simultaneous training of the representations coded at each state and the temporal model of the gesture, with the view that by training the two concurrently, the representations will suit the temporal model and the temporal model will suit the representations. By training multiple gestures concurrently, we can extend this idea further by arriving at a set of representations that are defined with respect to a multitude of gesture models.

In one form, this kind of training implies the sharing of observation probability distributions over a number of gesture models, each with its own matrix of state transition probabilities. This relationship between the Markov models and the observation distributions is shown in Figure 5-1.

Instead of updating a single gesture model after the re-estimation of the parameters of M_j, several gesture models would be updated after each re-estimation of M_j. Initially the Markov model associated with each gesture would be define over all available states. As training proceeds, states that are strongly suited to one particular gesture model would fall out of the other gesture models. This is similar in concept to the practice of *parameter tying* [24] in HMM's, in which parameters are shared over a number of states to reduce the

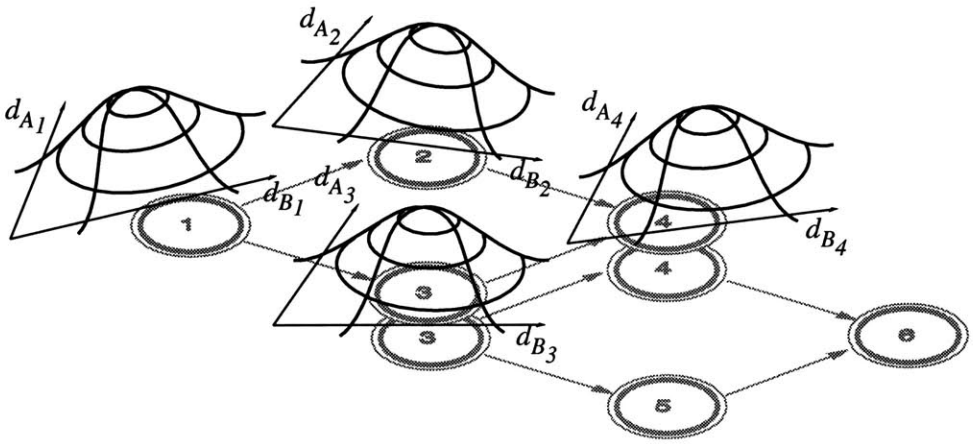


Figure 5-1: Two different gestures are associated with unique Markov models (gray), but share observation distributions at states 3 and 4.

number of degrees of freedom, thus easing the training procedure.

For example, in training two gestures concurrently, we can allocate a total of five states. An five by five state transition probability matrix is associated with each of the two gestures. During the training, the states will be associated with just one or both of the gestures. If one gesture never uses a particular set of the eight states, we would find a smaller transition matrix embedded within the five by five transition matrix associated with the gesture. This relationship of multiple Markov models and observation distributions is depicted in Figure 5-1.

There are a number of reasons for training multiple gestures concurrently:

- Two different gestures will be able to explicitly share states. Thus states will be allocated parsimoniously over the set of gestures. This is purely a computational advantage.

- In allowing gestures to share states, we gain a level of understanding of the gesture; we may find that a particular gesture consists of parts of other gestures. Furthermore, in this way we may find a small number of "atomic" primitives from which most of the gestures we're interested in are composed.

- By allocating the number of states that we can realistically store and by training a large number of gestures concurrently, the problem of deciding on the number of states to allocate to a particular gesture is finessed: during training, more states will

be allocated to the gestures that exhibit complex visual behavior.

A simpler method of training multiple gestures concurrently is to use a single transition matrix for the set of all gestures. However, if the gestures share states to a great degree, such a transition matrix would soon become "polluted" by the multiple gestures due to the limitation of the temporal model's modeling of first-order effects only. By imposing a number of temporal models (state transition matrices) on the same set of states, we avoid the limitations of first-order Markov models that would arise if we used a single temporal model for a number of gestures sharing states.

5.2 Selecting "representative" observations for a state

In the framework presented in Chapter 4, the model instances $m_j \in \mathcal{M}_j$ are computed to match the $\gamma_t(j)$-weighted samples. Presently, this is accomplished in practice by selecting some top number of the $\gamma_t(j)$-ranked samples. Alternatively, the if the model type can accept weighted samples, then $\gamma_t(j)$ may be used directly to weight each sample in the algorithm to compute the parameters of the model instance.

If we use the values of $\gamma_t(j)$ directly to weight each sample, there is the possibility of a state being "squeezed out" by other states, such that it is only given observations with very low $\gamma_t(j)$ values. This is analogous to the problem in k-means clustering where some cluster is assigned no samples. Furthermore, in such a situation it is possible that during the iterative training algorithm, a single state's observation distribution will grow to include all observations, squeezing all other states out.

Alternatively, by selecting some top number of samples as "representatives" of each state, regardless of the actual value of $\gamma_t(j)$, assures that every state has some set of observations that the model instances fit well. In selecting observations in this simple way, we are biasing the algorithm to converge to an interesting result.

Additionally, by considering only some number of the top ranked observations, the model instances avoid the incorporation of outliers; the selection of the top observations is one particular, strict form of robust statistics. In the multiple independent model subspace formulation, the value of $\gamma_t(j)$ does not always correspond to a distance to the state that is consistent with the model instance. Firstly, $\gamma_t(j)$ is computed with regards to the temporal model and is conditional on the fact that the process is actually in one of the

Initialization:
  for each gesture G:
    set $A_G(i,j) = \frac{1}{N}$ for all i, j
    initialize $\gamma_t^G(j)$ randomly, $\sum_{j=1}^{N} \gamma_t^G(j) = 1$
  end

Algorithm:
  repeat until parameters of $\mathcal{M}_j^G$ do not change, for all G:
    for each gesture G:
      for each state j:
        compute $usage_G(j)$
        select $M * usage_G(j)$ top $\gamma_t^G(j)$-ranked observations
        estimate parameters to models $m \in \mathcal{M}_j^G$
        compute $\mathbf{d}_j^G(\mathbf{x})$ for all $\mathbf{x} \in O$
        estimate $b_j^G(\mathbf{x})$
      end
      update A_G, $\gamma_t^G(j)$ by Baum-Welch algorithm
    end
  end

Figure 5-2: The current algorithm for training multiple gestures concurrently uses $M * usage_G(j)$ observations in estimating the model instance M_j^G for gesture G and state j, where M is the total number of observations to select over all gestures. By varying the sample rate in this way, we allow some states to eventually become completely associated with one gesture and also permit a state to be shared among a number of gestures.

states ($\sum_{j=1}^{N} \gamma_t(j) = 1$). Secondly, in the first iteration of the algorithm $\gamma_t(j)$ is assigned a random value: $\gamma_t(j)$ does not correspond to any distance relevant to the model instances. Because the value of $\gamma_t(j)$ is not a metric distance, outliers will have a detrimental effect on the estimation of observation distributions.

5.3 Esimation algorithm

The selection of "representative" observations is relevant in discussing the current algorithm for coding multiple gestures concurrently. As mentioned, by selecting the top number of observations, each state matches some set of observations well. Thus if we have multiple gestures sharing observation distributions, each with their own Markov model, every state of every Markov model will match some set of observations, the net effect being that every state is shared among some part of all gestures.

This is undesirable, since unless the gestures are in fact the same gesture, we should

never have the situation in which all states are shared among all gestures. [1] At the moment, we solve this problem by relying on a statistic that indicates the degree to which a particular gesture is using a state: the $usage_G$ of a state j by a gesture G is defined simply as the average of $\gamma_t(j)$ over the set of observations belonging to the training sequences of the gesture. We then vary the number of representative observations taken from each gesture for a particular state according to the gestures *usage* of the state. The full algorithm for training multiple gestures is shown in Figure 5-2.

For example, suppose we are training two gestures concurrently and we decide to choose $M = 100$ representative observations in calculating a model instance at a particular state. We first calculate the *usage* statistic of the state for each gesture from $\gamma_t(j)$ of the previous iteration of the algorithm. Suppose we find that $usage_A = 0.25$ and $usage_B = 0.75$. Then we select 25 representative observations taken evenly over the examples from gesture A and 75 from B. By varying the sample rate in this way, we allow some states to eventually become completely associated with one gesture and also permit a state to be shared among a number of gestures.

Unfortunately, the current algorithm tends not to share all states among two gestures if all training examples are in fact taken from the same gesture. However, given two different gestures, the algorithm as it stands will perform reasonably with two different gestures. For example, two different gestures were trained concurrently: a waving gesture and a throwing gesture. Plots of $\gamma_t(j)$ and d_m for the single model (eigenvector decomposition) are shown in Figure 5-3. The plots of $gamma_t(j)$ indicate that the states were divided evenly between the two gestures, though some states are show nonzero values for $\gamma_t(j)$ simultaneously, indicating that to some degree states were shared. Given that that the training image sequences for both examples were of the same arm moving in approximately the same location in the frame, the slight sharing of states is not surprising.

Future work with the concurrent training of multiple gestures may involve reevaluating the procedure of selecting representative values based on $\gamma_t(j)$.

[1] One other possibility is that all states are shared among all gestures, but the order of the states differs for each gesture.

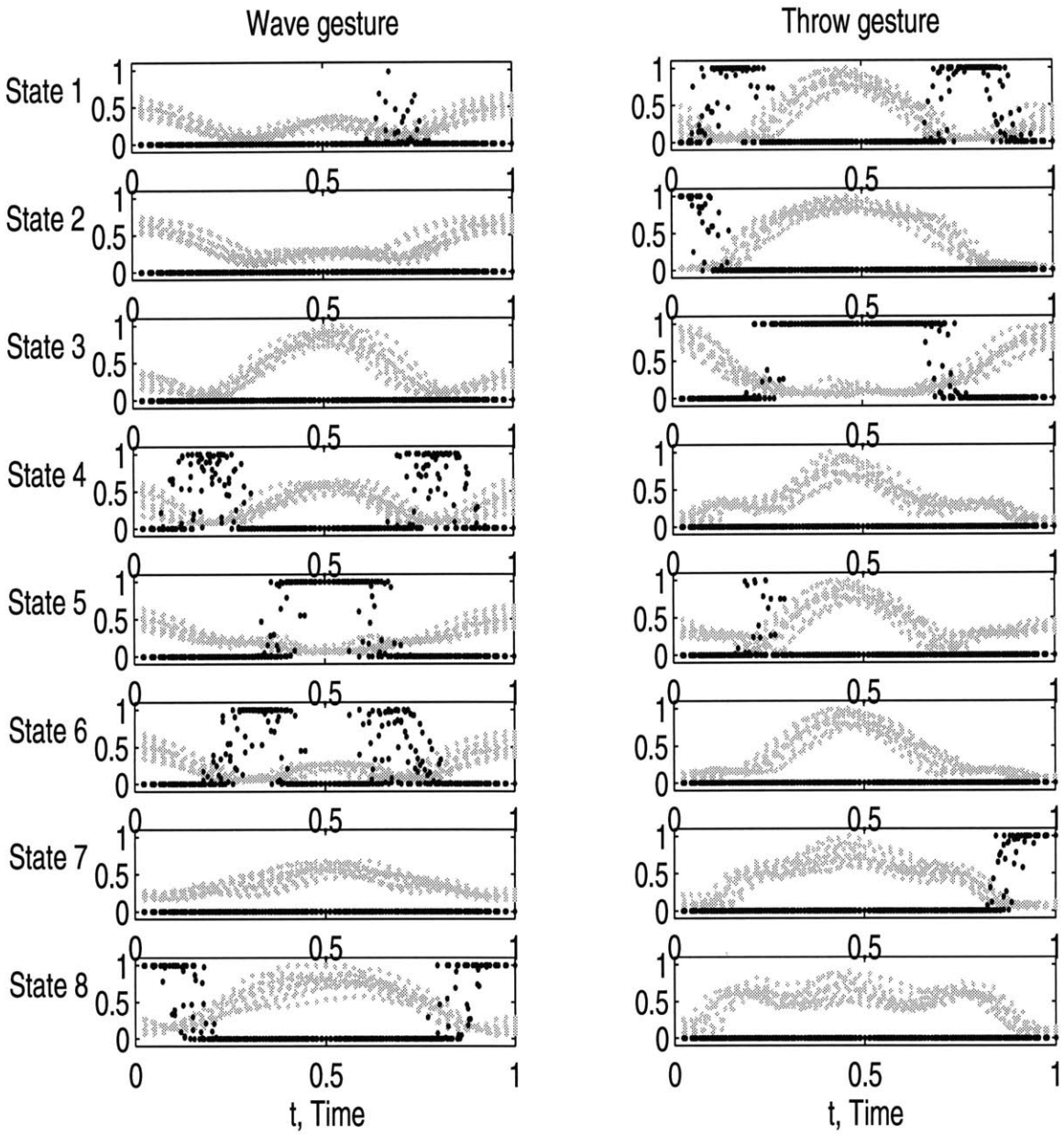


Figure 5-3: Plots of $gamma_t(j)$ (black) and d_m (gray) for two gestures trained concurrently. The plots of $gamma_t(j)$ indicate that the states were divided evenly over the gestures (four states for each gesture), though some states are briefly shared (states 1 and 5). (d_m was scaled to fit.)

5.4 Summary

The motivation for developing the framework presented in Chapter 4 to train multiple gestures concurrently were presented. In general, training in this way allows the establishment of relationships between gestures that the training of a single, independent HMM's for each gesture does not permit. Thus, the concurrent training of multiple gestures gives a new level of gesture understanding.

The current alogrithm presented in the chapter is the result of some preliminary study of techniques to train multiple gestures concurrently.

Chapter 6

Review and Future Work

6.1 Review

The thesis presents two techniques for computing a representation of gesture. The first defines a sequence of states along a prototype through a configuration space. The prototype is used to parameterize the samples and define the states. Dynamic programming may then be used to perform the recognition of new trajectories. The second technique, the multiple independent model subspace HMM, expands on the first by embedding a multiple model approach to the HMM framework. States are no longer computed with respect to a prototype, but arise out of an algorithm that interleaves the estimation of model paramters and the Baum-Welch algorithm.

By way of review, we highlight a number of differences between the techniques:


The configuration states method computes a prototype gesture, while the multiple independent model subspace HMM does not.


The multiple independent model subspace HMM relies on a combination of multiple models at each state; thus each state is associated with its own set of bases or coordinate axes. The configuration states method defines states that all reside in the same measurement space.


The multiple independent model subspace HMM computes the exact form of the measurements to be taken by computing the parameters to several model instances

that are consistent with the temporal model. In the configuration states method, the measurement space is determined in advance.


In the configuration states method, states are computed in a direct fashion, without an iterative method. The multiple independent model subspace HMM uses an iterative method much like a clustering algorithm to compute the states.


The configuration states method defines a gesture as a linear sequence of states, while the multiple independent model subspace HMM generalizes the sequence of states to a Markov model of states.

The configuration states method may be viewed as a particular specialization of the multiple model HMM. In fact, if we exchange the distance metric used for state membership for a probabilistic formulation of membership, the first method comes very close to the HMM framework. The important difference from the HMM framework is the way that assumptions about the task of representing gesture are used to make the task easier (non-iterative). Namely, these assumptions are that a gesture is a *sequence* of states, and that the trajectories that we take as the set of training gestures are smooth and "well-behaved". The smoothness and well-behaved nature of the training samples enables the computation of the prototype (see Appendix) which in turn is used to parameterize the samples and define the states.

Of the two techniques presented, the multiple independent model subspace HMM technique is superior:


The system may be used to "compile" a set of models to produce a description of visual behavior that may be useful as an efficient recognition or coding scheme (a set of model instances, and a Markov model).


The technique develops a representation concurrently with the temporal model, so that both are suited for each other; neither embodies unreasonable assumptions about the other.


The system should be able to make use of a wide variety of models that return distance measurements.


The possibility of the realistic use of multiple models exist, including selection of useful subsets of many models.

In the next section we explore possible future developments of the multiple model HMM technique.

6.2 Future Work

The multiple independent model subspace HMM framework leaves a number of interesting avenues of future research. The directions mentioned in this chapter are:


Real time coding: A robust, low-complexity coding algorithm is needed to make the framework practical for real time implementation.


Diverse models: The multiple model aspect has yet to be thoroughly explored with the use of a diverse collection of models.


Automatic model selection: The possibility exists for the framework to decide which models types to use, in a kind of *feature selection.*


Model combination: More sophisticated ways of combining model outputs may be better able to characterize the observations.


Multiple gestures: The ability to train multiple gestures concurrently may permit a higher level of gesture understanding.

Each of these issues is presented in turn.

6.2.1 Real time coding

In HMM applications, recognition may be performed by the "forward-backward" algorithm, which returns the probability of an observation sequence given a particular HMM. The algorithm requires computing the observation probability at each state, at each time step. When there is a large number of states to consider or the cost of computing $b_j(\mathbf{x})$ is high, the forward-backward algorithm may not scale to a reasonable number of states, especially in a real time recognition or coding application.

In the HMM framework of Chapter 4, the cost of computing the probability of a new observation is quite high, since we must compute d_m for all model instances $m \in \mathcal{M}_j$. For example, if we were to use the eigenvector decomposition of the image as our model with four HMM's with 25 states each, keeping the top 10 eigenvectors at each state, then 1,000 image correlations would be required at every time step.

The development of a robust coding method of low complexity is an important step in a real time implementation of the framework. One possibility is to use a *beam search* in which $b_j(\mathbf{x})$ is computed for a small subset of the available states, chosen to maximize the probability of being in the states given the past observations. This strategy is suboptimal in the sense that it will miss paths through the model that are locally of low probability but globally of high probability. It remains to be seen whether this limitation has practical merit: beam search may return a suboptimal, but qualitatively correct path, especially if the Markov model has many ways of producing the same gesture. In recovering the visual behavior of the gesture, a qualitatively correct solution will be adequate. However, if the beam search is very narrow (consider exploring just one state) and the Markov model is very strict about the sequence of states allowed to produce a completed gesture (consider a strict, linear Markov model), then a beam search may be so fragile as to often return a qualitatively incorrect solution.

6.2.2 Diverse Models

One clear direction of future work is in the use of more diverse models. The combination of models framework has been designed such that any models that return a distance measure may be combined at a state. A very interesting combination of models that we would like to explore is that of shape and motion. For example, we can use the eigenvector decomposition of images for a spatial description of the image and an optic flow field summary to describe the motion. An optic flow field summary might consist simply of the mean flow of the pixels corresponding to the objects of interest, or we may compute an eigenvector decomposition of the optic flow field and use the residual of reconstruction as the distance to model subspace.

The shape and motion pairing invites interesting analyses as to whether the particular spatial information or motion information is important in human gesture. It also may be the case that the particular relevant feature changes between gestures, or even within the same gesture.

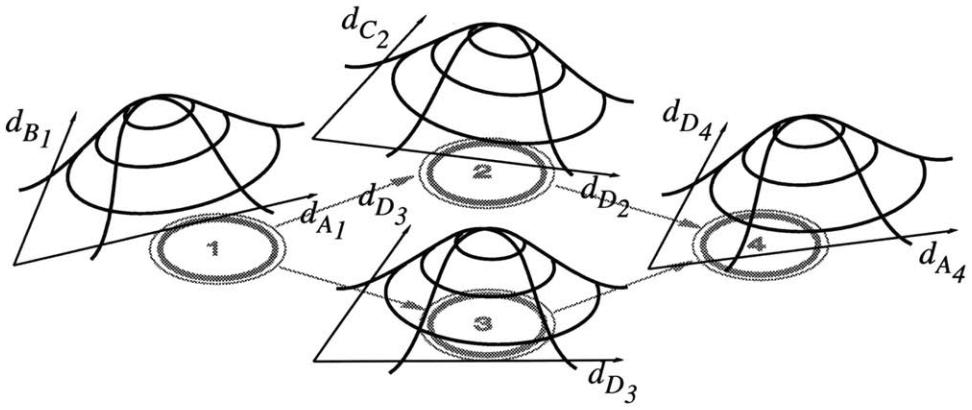


Figure 6-1: Each state of the Markov model (gray) is associated with a unimodal observation pdf, each defined in the space of distances to a number of model instances, drawn from a set of four model types, $\mathcal{M} = \{A, B, C, D\}$.

6.2.3 Automatic Model Selection

Because the observation distribution $b_j(\mathbf{x})$ is a joint distribution on the distances to multiple independent model subspaces, the set of model types at each state does not have to be same across all states. One topic of future work involves the automatic selection of model types at each state. For example, one state may be characterized by motion sufficiently, while in another both motion and shape are required to characterize the observations. This use of different models at each each state is depicted in Figure 6-1.

Automatic selection of models is desirable for a number of reasons. First, a particular model type may be unable to characterize the training examples at a state, in which case erroneous model instances should not contribute to the state and should be removed from the set of models at the state. Secondly, if the appropriate set of features is not known beforehand, a good approach is to use all available models and let the system select the useful ones, possibly subject to a cost constraint. Lastly, the selection of different subsets of model types at each state allows the characterization of behaviors in which the set of relevant features changes over time. Automatic selection of models may provide insight into the degree to which this happens in the case of human gesture.

One criterion for model selection is the minimization of model error. If a model is unable to match even the observations that belong to the state in consideration, it should be removed from the set of model types used at the state. Model error is not a problem in the use of eigenvector decomposition, since there is no set of images that cannot be reconstructed well by some number of eigenvectors. In our current system, the number of eigenvectors

is selected to account for a certain percentage of the variance of the examples, and so the model always fits the examples well. However, one could imagine that a certain model type might fail altogether (the "failure" criterion depending on the model type used) in matching the observations. By detecting model types that fail to characterize the observations at a state, a number of models that are highly specialized to detect certain kinds of observations may be used practically.

Another criterion might be the minimization of the classification error of the examples at each state. Ignoring the temporal aspect of the signals, we can simply compute the degree to which the observation distributions $b_j(\mathbf{x})$ overlap by computing their *divergence* [58]. Note that by ignoring the temporal aspect of the signal, a selection of models that minimizes the overlap of the observation distributions may be more thorough than necessary; for instance, if two states are several transitions away in the Markov model, the degree to which their observation distributions overlap may not be important, since the Markov model removes most of the chance of ambiguity.

The selection may also be done with respect to a cost function. If the goal is the real time implementation of a recognition system, for example, then a good cost constraint might involve the run-time of the computation necessary to compute d_{m_j}. The cost constraints might be designed to take advantage of the hardware available; for example, if two computations may be done in parallel, then the cost function should penalize the system appropriately.

Lastly, there is the question of whether model selection should occur after training is completed, or during the training. Selecting models during training may complicate convergence, but may also result in a more optimal combination of model instances and temporal models as well as lower computational complexity of training. Selection after training has the benefit that the selection of models is always a matter of taking away model types and never one of adding model types, but may result in a very lengthy training session.

6.2.4 Model combination

In the multiple model approach presented in Chapter 4, a particularly simple form of the combination of model outputs has been adopted. Namely, a *conjunctive* combination of model outputs is used. A more general formulation would consider a wider variety of

combination rules. For example, model outputs may be combined by *disjunction* ('OR' or 'XOR'). Additionally, one might model the observation distribution $b_j(\mathbf{x})$ with a distribution with more degrees of freedom, such as a mixture of Gaussians, rather than the simple unimodal multivariate Gaussian used.

To a certain extent, the *lack* of additional degrees of freedom in representing model outputs in the current system should be balanced against the ability of an HMM with many states to assume the same degrees of freedom. For example, it may be the case that an HMM with unimodal output distributions and many states is no less powerful a representation than an HMM with few states and a mixture of Gaussians used to estimate each $b_j(\mathbf{x})$. As an example, consider a single state with its output distribution approximated by a mixture of two Gaussians. The same functionality may be assumed by two states with identical transition probabilites, each with a unimodal output distribution.

In the case of the combination of models in a disjunctive or conjunctive way, the relevant particular combinations might be assumed over a number of states; for example, one state may capture one set of conditions ("either") and another an exclusive set of other conditions ("or").

The recovery of topology from very weak initial conditions, as discussed in the previous chapter, may allow such rich representations to fall out of simple unimodal distribution and transition probability primitives.

6.3 Unsupervised Clustering of Gestures

Another interesting treatment of training multiple gestures involves starting from a number of *unlabeled* training sequences and by a hierarchical clustering technique, arriving at a set of individually defined gestures. For example, the training procedure may begin by assuming that all training sequences are from the same gesture, and proceed as usual with a large number of states. The *Viterbi algorithm* [24] may be used to calculate the most likely sequence of states associated with each training example. Training sequences may the be clustered according to which states of the large Markov model were actually used; the larger model may be split into a number of smaller models and training may proceed with multiple gestures, each with its own Markov model, but sharing observation distributions.

Such a procedure would be appropriate in situations in which the labels of gestures are

unknown. If the techniques presented in the thesis are applied to a domain in which the relationship between the visual appearance of the sequence and the labeling is weak (as in the coding of unconstrained video), such an unsupervised approach may be used to find a set of labels implicitly defined by the sensors used. Or it may be useful if the labels are known, but we wish to determine if the movements are distinct given the sensors.

Again, the treatment of multiple gesture concurrently may provide a higher level of understanding.

6.3.1 Summary of future work

A number of interesting additions to the multiple model HMM framework were presented. The need for a robust, real time coding scheme was discussed briefly. The development of such an algorithm will be driven by the need for a real time implementation, as well as to demonstrate that the complexity of the recognition or coding phase of the framework does not preclude its practical use.

The application of more diverse models is an obvious extension to the current work, and may provide interesting results in how to best to characterize gesture given the current framework.

The automatic selection of model types will enable the framework to be useful in a wide variety of visual scenarios, by enabling the framework to use a large number of models.

The use of more sophisticated model combinations may be an interesting topic that may be useful in helping the framework to use a large number of models more effectively.

The multitude of interesting possibilities for future work attest to the framework's general utility in characterizing visual behavior.

6.4 Conclusion

In conclusion, we believe that the techniques presented in the thesis are a significant step forward in the development of methods that bring "killer demos" of Chapter 1 closer to reality. A number of interesting and significant topics pave the way for a powerful, flexible and practical implementation of the coding of visual behavior for gesture analysis.

Appendix A

Computation of Time-Collapsed Prototype Curve

For each trajectory $T_i(t)$, we have a $\hat{T}_i(t) = T_i(\frac{t}{s})$, where s is a scalar that maps the time parameterization of $T_i(t)$ and $\hat{T}_i(t)$. The time course of all example trajectories are first normalized to the same time interval $[0, s]$. The smooth approximation of the time-normalized sample points gives a rough starting point in determining which of the sample points correspond to a point on the prototype. These correspondences can be refined by iteratively recomputing the approximation while successively reducing the time scale s. If the prototype curve is not allowed to change drastically from one iteration to the next, a temporally coherent prototype curve in the original configuration space will result.

To compute the prototype curve $P(\lambda)$, we use Hastie and Stuetzle's "principal curves" [22]. Their technique results in a smooth curve which minimizes the sum of perpendicular distances of each sample to the nearest point on the curve. The arc length along the prototype of the nearest point is a useful way to parameterize the samples independently of the time of each sample. That is, for each sample x_i there is a lambda which minimizes the distance to $P(\lambda)$: $\lambda(x_i) = \arg\min_{\lambda} \|P(\lambda) - x_i\|$. An example is shown in Figure A-1.

The algorithm for finding principal curves is iterative and begins by computing the line along the first principal component of the samples. Each data point is then projected to its nearest point on the curve and the arc length of each projected point is saved. All the points that project to the same arc length along the curve are then averaged in space. These average points define the new curve. This projection and averaging iteration proceeds until

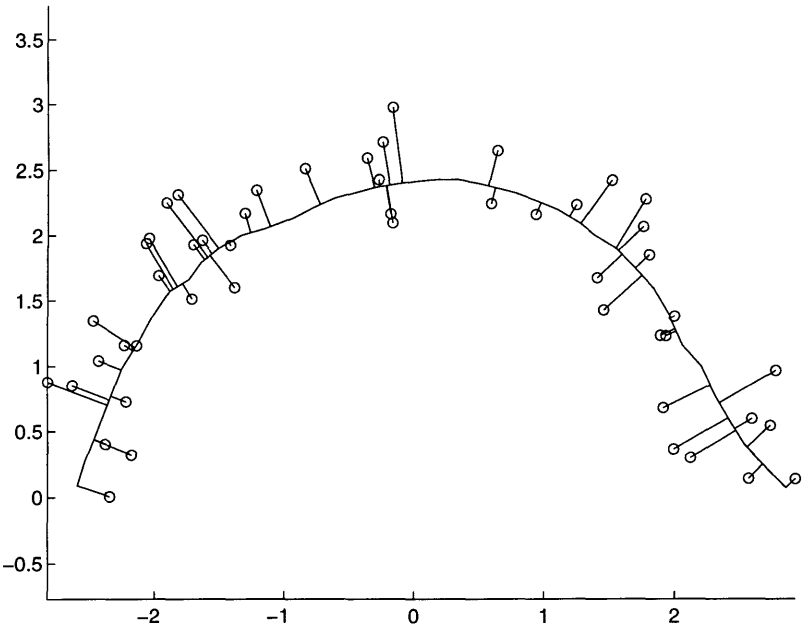


Figure A-1: A principal curve calculated from a number of points randomly scattered about an arc. Only a fraction of the data points are shown; the distances of these points to the closest points on the curve are indicated.

the change in approximation error is small.

In practice, no sample points will project to a particular arc length along the curve. Therefore, a number of points that project to approximately equal arc lengths are averaged. The approach suggested by Hastie and Stuetzle and used here is to compute a weighted least squares line fit of the nearby points, where the weights are derived from a smooth, symmetric and decreasing kernel centered about the target arc length. The weight w for a sample x_i and curve point $p = P(\lambda)$ is given by

$$w = \left(1.0 - (\frac{|\lambda(p) - \lambda(x_i)|}{h})^3\right)^3$$

where h controls the width of the kernel.

The new location of the curve point is then the point on the fitted line that has the same arc length. For efficiency, if the least squares solution involves many points, a fixed number of the points may be selected randomly to obtain a reliable fit.

By starting with a time scaling s which renders the trajectories slowly varying in the configuration space parameters as a function of arc length, the principal curve algorithm computes a curve which is consistent with the temporal order of the trajectory samples.

Then the time scale s can be reduced somewhat and the algorithm run again, starting with the previous curve. In the style of a continuation method, this process of computing the curve and rescaling time repeats until the time scale is zero, and the curve is in the original configuration space. To ensure that points along the prototype do not coincide nor spread too far from one another as the curve assumes its final shape, the principal curve is re-sampled between time scaling iterations so that the distance between adjacent points is constant.

The inductive assumption in the continuation method is that the curve found in the previous iteration is consistent with the temporal order of the trajectory samples. This assumption is maintained in the current iteration by a modification of the local averaging procedure in the principal curves algorithm. When the arc length of each point projected on the curve is computed, its value is checked against the point's arc length computed in the previous iteration. If the new arc length is drastically different from the previously computed arc length ($|\lambda_t(x_i) - \lambda_{t-1}(x_i)| > threshold$), it must be the case that by reducing the time scale some other part of the curve is now closer to the sample point. This sample point to prototype arc length correspondence is temporally inconsistent with the previous iteration, and should be rejected. The next closest point on the curve $P(\lambda)$ is found and checked. This process repeats until a temporally consistent projection of the data point is found.

By repeatedly applying the principal curve algorithm and collapsing time, a temporally consistent prototype $P(\lambda)$ is found in configuration space. Additionally, the arc length associated with each projected point, $\lambda(x_i)$, is a useful time-invariant but order-preserving parameterization of the samples. An example of this time-collapsing process is shown in Figure A-2.

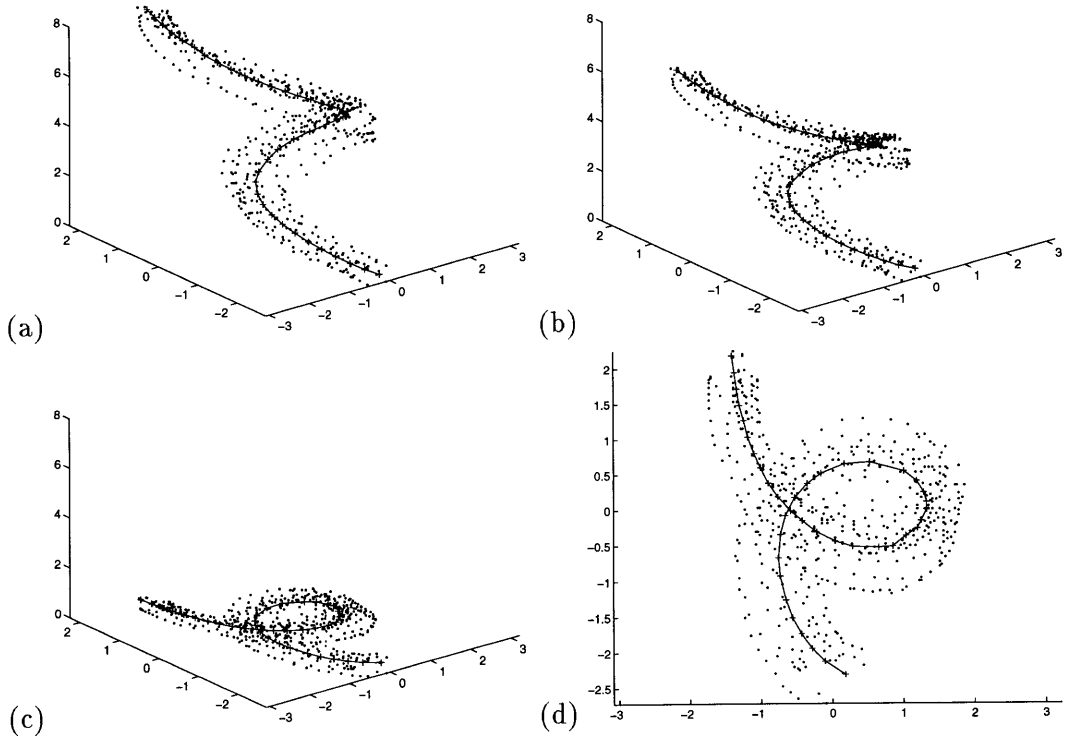


Figure A-2: The principal curve is tracked while time is slowly collapsed in this series: (a) $s = 8$, (b) $s = 5$, (c) $s = 0$. In each of these graphs, the vertical axis is time. (d) shows the final, temporally consistent curve.

Bibliography

[1] Mark Allman and Charles R. Dyer. Towards human action recognition using dynamic perceptual organization. In *Looking at People Workshop, IJCAI-93*, Chambery, Fr, 1993.

[2] D. Beymer, A. Shashue, and T. Poggio. Example based image analysis and synthesis. Artificial Intellgence Laboratory Memo 1431, Massacusetts Institute of Technology, 1993.

[3] A. F. Bobick and A. D. Wilson. A state-based technique for the summarization and recognition of gesture. *Proc. Int. Conf. Comp. Vis.*, 1995.

[4] R. A. Bolt and E. Herranz. Two handed gesture in multi-modal natural dialog. In *Proc. of UIST '92, Fifth Annual Symp. on User Interface Software and Technology*, Monterey, CA, 1992.

[5] L. W. Campbell and A. F. Bobick. Recognition of human body motion using phase space constraints. In *Proc. Int. Conf. Comp. Vis.*, 1995.

[6] E. Catmull and R. Rom. A class of local interpolating splines. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 317–326, San Francisco, 1974. Academic Press.

[7] C. Cedras and M. Shah. Motion-based recognition: a survey. Computer Vision Laboratory Technical Report, University of Central Florida, 1994.

[8] C. Cedras and M. Shah. A survey of motion analysis from moving light displays. *Proc. Comp. Vis. and Pattern Rec.*, pages 214–221, 1994.

[9] C. Charayaphan and A.E. Marble. Image processing system for interpreting motion in American Sign Language. *Journal of Biomedical Engineering*, 14:419–425, September 1992.

[10] Z. Chen and H. Lee. Knowledge-guided visual perception of 3-D human gait from a single image sequence. *IEEE Trans. Sys., Man and Cyber.*, 22(2), 1992.

[11] Y. Cui and J. Weng. Learning-based hand sign recognition. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*, Zurich, 1995.

[12] T.J. Darrell and A.P. Pentland. Space-time gestures. *Proc. Comp. Vis. and Pattern Rec.*, pages 335–340, 1993.

[13] J. W. Davis and M. Shah. Gesture recognition. *Proc. European Conf. Comp. Vis.*, pages 331–340, 1994.

[14] B. Dorner. Hand shape identification and tracking for sign language interpretation. In *IJCAI Workshop on Looking at People*, 1993.

[15] S. Edelman. Representation of similarity in 3D object discrimination. Department of Applied Mathematics and Computer Science Technical Report, The Weizmann Institute of Science, 1993.

[16] I. Essa, T. Darrell, and A. Pentland. Tracking facial motion. In *Proc. of the Workshop on Motion of Non-Rigid and Articulated Objects*, Austin, Texas, Nov. 1994.

[17] I. Essa, T. Darrell, and A. Pentland. A vision system for observing and extracting facial action parameters. In *Proc. Comp. Vis. and Pattern Rec.*, pages 76–83, 1994.

[18] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*, Zurich, 1995.

[19] K. Gould, K. Rangarajan, and M. Shah. Detection and representation of events in motion trajectories. In Gonzalez and Mahdavieh, editors, *Advances in Image Analysis*, pages 393–426. SPIE Optical Engineering Press, 1992.

[20] K. Gould and M. Shah. The trajectory primal sketch: a multi-scale scheme for representing motion characteristics. *Proc. Comp. Vis. and Pattern Rec.*, pages 79–85, 1989.

[21] P. M. Grant. Speech recognition techniques. *Electronics and communication engineering journal*, pages 37–48, February 1991.

[22] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[23] David Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1(1):5–20, Feb 1983.

[24] X.D. Huang, Y. Ariki, and M. A. Jack. *Hidden Markov Models for Speech Recognition.* Edinburgh University Press, 1990.

[25] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14(2):201–211, 1973.

[26] R. Kjeldsen and J. Kender. Visual hand gesture recognition for window system control. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*, Zurich, 1995.

[27] J.S. Lipscomb. A trainable gesture recognizer. *Pattern Recognition*, 24(9):895–907, 1991.

[28] D. G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *Int. J. of Comp. Vis.*, 8(2):113–122, 1992.

[29] P. Maes, T. Darrell, B. Blumberg, and A. Pentland. The ALIVE system: full-body interaction with animated autonomous agents. MIT Media Lab Perceptual Computing Group Technical Report No. 257, Massachusetts Institute of Technology, 1994.

[30] K. V. Mardia, N. M. Ghali, M. Howes T. J. Hainsworth, and N. Sheehy. Techniques for online gesture recognition on workstations. *Image and Vision Computing*, 11(5):283–294, 1993.

[31] D. McNeill. *Hand and Mind: What Gestures Reveal About Thought.* Univ. of Chicago Press, Chicago, 1992.

[32] B. Moghaddam and A. Pentland. Maximum likelihood detection of faces and hands. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*, Zurich, 1995.

[33] B. Moghaddam and A. Pentland. Probabalistic visual learning for object detection. In *Proc. Int. Conf. Comp. Vis.*, Cambridge, MA, 1995.

[34] H. Morita, S. Hashimoto, and S. Ohteru. A computer music system that follows a human conductor. *Computer*, 24(7):44–53, July 1991.

[35] H. Murase and S. Nayar. Learning and recognition of 3d objects from appearance. In *IEEE 2nd Qualitative Vision Workshop*, New York, June 1993.

[36] H. Murase and S. K. Nayar. Parametric eigenspace representation for visual learning and representation. Department of Computer Science Technical Report CUCS-054-92, Columbia University, 1992.

[37] D. H. Parish, G. Sperling, and M.S. Landy. Intelligent temporal subsampling of american sign language using event boundaries. *Journal of Experimental Psychology: Human Perception and Performance*, 16(2):282–294, 1990.

[38] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. *Proc. Comp. Vis. and Pattern Rec.*, 1994.

[39] R. W. Picard and T. P. Minka. Vision texture for annotation. *Journal of Multimedia Systems*, 3:3–14, 1995.

[40] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343(6225):263–266, 1990.

[41] R. Polana and R. Nelson. Low level recognition of human motion. In *Proc. of the Workshop on Motion of Non-Rigid and Articulated Objects*, pages 77–82, Austin, Texas, Nov. 1994.

[42] F. Quek. Hand gesture interface for human-machine interaction. In *Proc. of Virtual Reality Systems*, volume Fall, 1993.

[43] J. R. and N. I. Badler. Model based image analysis of human motion using constraint propagation. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 2(6), November 1980.

[44] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–285, February 1989.

[45] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

[46] L. R. Rabiner and B. H. Juang. *Fundamentals of speech recognition.* Prentice Hall, Englewood Cliffs, 1993.

[47] K. Rangarajan, W. Allen, and M. Shah. Matching motion trajectories using scale-space. *Pattern Recognition*, 26(4):595–610, 1993.

[48] R. Rashid. Towards a system for the interpretation of moving light displays. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 2:574–581, 1980.

[49] J. M. Rehg and T. Kanade. DigitEyes: vision-based human hand tracking. School of Computer Science Technical Report CMU-CS-93-220, Carnegie Mellon University, December 1993.

[50] J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. *Proc. European Conf. Comp. Vis.*, 2:35–46, 1994.

[51] K. Rohr. Towards model-based recognition of human movements in image sequences. *Comp. Vis., Graph., and Img. Proc.*, 59(1):94–115, 1994.

[52] J. Schlenzig, E. Hunter, and R. Jain. Recursive identification of gesture inputs using hidden markov models. *Proc. Second Annual Conference on Applications of Computer Vision*, pages 187–194, December 1994.

[53] J. Schlenzig, E. Hunter, and R. Jain. Vision based hand gesture interpretation using recursive estimation. In *Proc. of the Twenty-Eighth Asilomar Conf. on Signals, Systems and Comp.*, October 1994.

[54] A. Shio and J. Sklansky. Segmentation of people in motion. In *Proc. IEEE Workshop on Visual Motion*, pages 325–332, 1991.

[55] G. Sperling, M. Landy, Y. Cohen, and M. Pavel. Intelligible encoding of ASL image sequences at extremely low information rates. *Comp. Vis., Graph., and Img. Proc.*, 31:335–391, 1985.

[56] T. E. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*, Zurich, 1995.

[57] A.I. Tew and C.J. Gray. A real-time gesture recognizer based on dynamic programming. *Journal of Biomedical Eng.*, 15:181–187, May 1993.

[58] C. W. Therrien. *Decision Estimation and Classfication.* Wiley, New York, 1989.

[59] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[60] J. Weng. On comprehensive visual learning. In *Proc. NSF/ARPA Workshop on Performance vs. Methodology in Computer Vision*, pages 152–166, Seattle, 1994.

[61] A. Wexelblat. A feature-based approach to continuous-gesture analysis. Master's thesis, Masschusetts Institute of Technology, 1994.

[62] A. D. Wilson and A. F. Bobick. Using configuration states for the representation and recognition of gesture. MIT Media Lab Perceptual Computing Group Technical Report No. 308, Massachusetts Institute of Technology, 1994. Available at http://www-white.media.mit.edu/vismod.

[63] A. D. Wilson and A. F. Bobick. Learning visual behavior for gesture analysis. In *Proc. IEEE Int'l. Symp. on Comp. Vis.*, 1995. submitted for publication.

[64] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. *Proc. Comp. Vis. and Pattern Rec.*, pages 379–385, 1992.