Adaptive Models for the Recognition of Human Gesture

by
Andrew David Wilson
B.A., Computer Science, Cornell University (1993)
M.S., Massachusetts Institute of Technology (1995)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2000

Author .....
Program in Media Arts and Sciences,
School of Architecture and Planning
August 10, 2000

Certified by .....
Aaron F. Bobick
Associate Professor
College of Computing, Georgia Institute of Technology
Thesis Supervisor

Certified by .....
Bruce M. Blumberg
Assistant Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by .....
Stephen A. Benton
Chairman, Department Committee on Graduate Students
Program in Media Arts and Sciences

Adaptive Models for the Recognition of Human Gesture

by

Andrew David Wilson

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 10, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Tomorrow's ubiquitous computing environments will go beyond the keyboard, mouse and monitor paradigm of interaction and will require the automatic interpretation of human motion using a variety of sensors including video cameras. I present several techniques for human motion recognition that are inspired by observations on human gesture, the class of communicative human movement.

Typically, gesture recognition systems are unable to handle systematic variation in the input signal, and so are too brittle to be applied successfully in many real-world situations. To address this problem, I present modeling and recognition techniques to adapt gesture models to the situation at hand.

A number of systems and frameworks that use *adaptive gesture models* are presented. First, the parametric hidden Markov model (PHMM) addresses the representation and recognition of gesture families, to extract *how* a gesture is executed. Second, strong temporal models drawn from natural gesture theory are exploited to segment two kinds of natural gestures from video sequences. Third, a realtime computer vision system learns gesture models online from time-varying context. Fourth, a realtime computer vision system employs hybrid Bayesian networks to unify and extend the previous approaches, as well as point the way for future work.

Thesis Supervisor: Aaron F. Bobick
Title: Associate Professor
College of Computing, Georgia Institute of Technology

Thesis Supervisor: Bruce M. Blumberg
Title: Assistant Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Adaptive Models for the Recognition of Human Gesture

by

Andrew David Wilson

The following people served as readers for this thesis:

Read by .....

Allan D. Jepson
Professor of Computer Science
Department of Computer Science, University of Toronto
Thesis Reader

Read by .....

William T. Freeman
Senior Research Scientist
Mitsubishi Electric Research Labs, Cambridge, MA
Thesis Reader

Acknowledgments

I've had the privilege of working with a really swell group of folks at the Media Lab, past and present. Chief among them of course are Aaron's original long-haul students: Lee Campbell, Stephen Intille, Claudio Pinhanez and Jim Davis. Yuri Ivanov has been a great officemate, colleague and friend. Matt Krom, Dave Becker, Josh Wachman, Chris Bentzel, and Teresa Marrin have all been terrific, entertaining officemates. I'm proud to have served with the Old Guard of Vismod, all great friends and colleagues: Baback Moghaddam, Ali Azarbayejani, Irfan Essa, Trevor Darrell, Kris Popat, Alex Sherstinsky, and Thad Starner. They've influenced me in so many ways. My generation: Chris Wren, Sumit Basu, Nuria Oliver, Tony Jebara, and Martin Szummer. Bruce's guys: Chris Kline, Michal Hlavac, Michael P. Johnson, Ken Russell, Zoe Teegarden and Bill Tomlinson.

Thanks to Allan and Bill for serving as readers. Thanks to Bruce for taking me on when Aaron headed off for greener pastures. I owe thanks to Whitman Richards for some good advice, and to Joe Paradiso for letting me play with hardware. Tommy Poggio, Michael Jordan and Ted Adelson have been instrumental in shaping my thinking.

Thanks to Aaron for giving me the freedom to blaze my own trail through graduate school, and for always asking the hard questions.

Thanks to my parents for bringing me into this world and making it an interesting place by leaving cool things around the house to play with. Thanks to my dad for teaching me to think, and to my mom for teaching me to create.

Finally, thanks to my love Ewa. Would never have made it without you.

Contents

Chapter 1

Introduction

1.1 Ubiquitous Computing and Interfaces

The era of *ubiquitous computing* is nearly upon us. Pundits and academics agree that soon the familiar keyboard, mouse and monitor interface will give way to a variety of interfaces that are more specialized to the task at hand, and that nontrivial computing power will be embedded in our homes, offices, cars, phones, television, and perhaps even our refrigerator[53]. The reasons for such a transition are myriad: CPU's and memory chips are cheap enough that it is reasonable to have many computers where once you could only afford one or two; the advent of the network raises the possibility of connecting, distributing and deploying specialized devices in interesting ways; and most importantly, computers have gone far beyond early traditional business and desktop publishing applications to penetrate almost every aspect of our lives.

With respect to the problem of user interface design, there are a few straightforward consequences to the ubiquitous computing revolution:

- The applications of computing technology will be varied and widespread, making it difficult to adopt a single interface paradigm that works in all situations.
- As the machines become more involved in daily live, our interactions with machines will become much richer, more subtle and less symbolic in nature.
- Users will be unwilling to adopt a multitude of entirely new ways of interacting with their machines, preferring instead to adopt modes of interaction they already know.

The question for the designers of the ubiquitous computing systems, is if not the keyboard, mouse and monitor, what then? Speech comes to mind first, if only because we all have grown up on Star Trek. Clearly, speech as an interface is suited to many application domains (for example, the car and the phone), and there is a great deal of research directed to this goal.

It is tempting to think that speech will be the one interface that will work in all situations. Human movement and gesture will also be rich input signal. In the most recent user interface revolution, the transition from command line prompts to the

graphical user interface (GUI), we have seen the mouse exploit human movement to replace many of the functions of the keyboard. The use of human motion will go beyond the mouse however, as the nature of our interaction with computers becomes richer and more subtle.

For example, in some circumstances gesture complements the speech signal; for example, when you are gesturing to emphasize or illustrate a co-occurring utterance, the information conveyed by the gesture is typically not found in the spoken utterance. For example, in the spoken utterance, "the fish was *this* big", the word "this" may be accompanied by a gesture that indicates how big the fish was. In other circumstances gesture may stand alone. Other than sign language, systems that employ *iconic* gestures often do not rely on speech, either because the speech signal is unavailable or too unreliable. For example, crane operators have developed a gesture vocabulary for communicating to the operator from the ground spotter.

Chances are that that while the interfaces around us proliferate, they will tend to work as we do already and not require the years of training needed to use a keyboard efficiently. Take the Palm Pilot, for example. Its novel interface is a product of compromises centered around the need to do away with the traditional keyboard. Its method of input, while stopping short of full handwriting recognition, takes only a few days to learn because it is based on *iconic* gestures: gestures that resemble the desired input. Note that in this case (and likely in many future interfaces) the transition from keyboard to a more "natural" interface involves converting a continuous sequence (pen strokes) to symbols (letters).

1.2 Human Gesture as Communicative Movement

This thesis explores a number of approaches to the automatic recognition of human gesture, one possible component of a ubiquitous computing system interface.

What exactly is meant by the term "human gesture"? There is such a variety of flavors of gesture that it is difficult to delve into a definition of gesture without specifying an application domain. For example, bond traders on the Chicago Board of Trade have a system of symbolic gestures used to communicate bids across the trading floor. The gesture used in sign language is an entirely different scheme of gesture, while the gestures people make naturally while speaking is yet another entirely different kind of gesture. The gestures made by the conductor of orchestra are different still, and there is the mouse, the Palm Pilot, and other devices that exploit human movement.

It is therefore a mistake to think (as has been proposed at a conference on gesture recognition) that researchers may develop a common database of gesture signals with which to compare various recognition algorithms, as was previously done in the fields of speech and face recognition. Similarly it is difficult to compare recognition rates, confusion matrices and other standard metrics of algorithm performance reported in gesture recognition research papers.

I therefore define human gesture in a somewhat pragmatic fashion as *communicative human movement*. Thus the broad goal of the thesis is to design computer

systems that are able to extract meaning from human motion. In this thesis I will address a variety of forms of gesture, and the kind of gesture will be more narrowly specified when it is appropriate to do so. It will be apparent that the question of application domain is largely orthogonal to the concerns of the present work. While this thesis presents specific applications rather than a single general approach, there are several aspects that are common to the approaches used in each application, including the overall emphasis on machine learning and the specific computational techniques used.

One point that I hope will become apparent to the reader is that the deep problems of gesture recognition are found no matter what particular input device is used. The same issues apply whether one is concerned with extracting meaning from signals derived from a mouse or from an image sequence of person gesturing to the camera. For example, a particular issue common to all input devices is the realization that different people will perform the same gesture in different ways. Another issue is how to represent gestures that exhibit meaningful variation from instance to instance. This thesis examines both of these issues.

1.3 Why Adaptive?

Many gesture recognition applications today involve hand-coded strategies to recognize gestures. A more promising approach is that suggested by traditional pattern recognition techniques, where a number of example signals (gestures) are collected and subsequently summarized automatically by a process which fits a compact model to the collection of training signals. Later, when the application is running, the models are matched to the input signal. When a model matches the signal well, the application may conclude that the gesture corresponding to the model occurred. Hidden Markov models are one such framework for the automatic learning of gestures for later recognition.

I used a standard hidden Markov model approach in designing the gesture recognition framework for *Swamped!* an interactive installation in which the user controlled a virtual character by a stuffed doll instrumented with various wireless sensors (see Figure 1-1). Rather than adopt an instantaneous control paradigm whereby each movement of the doll was mimicked by the virtual character, we experimented with the idea of *iconic* control, in which the mimicry operated at the symbolic level: when the user moves the doll in a walking motion (either by rocking the doll side to side or twisting the doll as in shuffling the feet) the system would recognize the movement as the walking gesture, which then directed the virtual character to walk. One motivation for iconic control over direct control is the ability to constrain on-screen animation to be "in character" while exploiting hand-crafted animation. The nature of this interface is described in more detail in [35], and a QuickTime video of the interface may be found at `http://www.media.mit.edu/~drew/movies`.

Examples of a variety of gestures were collected by me and a few other people involved in the project. When *Swamped!* was shown to several hundred users at SIGGRAPH '98, it was clear that some of the gestures were not picked up by the

Figure 1-1: *Swamped!* is an interactive installation in which the user controls a virtual character (a chicken) by manipulating a plush doll instrumented with a variety of sensors.

users, and some of the gestures that we directed the users to try were not recognized by the system. The problem was that different users have different ways of executing the same gesture, such that many times the gesture executed by the user was not recognized by the system. In general it is very difficult to provide a set of examples that spans the variation that one sees in a large number of users. What was needed was an adaptive scheme whereby the system figured out on-the-fly what the user's "kick" gesture looked like.

Also in the *Swamped!* project, there was the lingering desire to know more about *how* the gesture was executed so that animations could be varied continuously or emotion levels in the behavior system could be tweaked.

This thesis addresses the shortcomings of the standard gesture recognition paradigm as embodied by my gesture recognition implementation for *Swamped!*: parameterized models for extracting how a gesture is executed, and online adaptive techniques for adapting to users in an online fashion. I group these ideas under the under the term *adaptive gesture models* to distinguish them from the usual (non-adaptive) gesture recognition paradigm, which in my experience is too brittle to be applied successfully in many real-world situations.

1.4 Adaptive Gesture Models

Adaptive gesture models include parameters that are adapted or specialized on-the-fly to match the particular circumstances at hand. This will be done by leaving some parts of the model dependent on free parameters, and adapting them in response to other constraints of the model, which themselves may be learned in an adaptive manner at an earlier time. With adaptive parameters at its disposal, a model may be specialized to address a wide variety of runtime circumstances.

There are two standard alternatives to the adaptive approach. First we may address the many possible scenarios our gesture recognition may encounter by training as many models as there are unique scenarios. This has the disadvantage that often it is not possible to enumerate the many circumstances that our algorithm will encounter. Furthermore, unless this is done cleverly, we will require more training data than we will likely be able to collect. The second and more common approach is to look for diagnostic features that generalize in exactly the right manner. Feature selection is in general a very difficult problem, however. The manual design of diagnostic features is an art that usually requires a very fine understanding of the domain, while automatic approaches usually result in representations that are opaque to the designer.

The advantages of the adaptive approach are:

- By adapting models to local circumstances, existing models are effectively reused automatically, potentially saving the time of training a new model for a new situation.
- Generalization refers to the ability of the algorithm to perform on previously unseen data. An adaptive model's generalization performance will be related to its adaptive features, and hence, should generalize well along those features by design.
- Because some parameters are left to be fixed during runtime, one might expect an adaptive model to require less offline training.
- The dichotomy of global invariants and local circumstances forces the designer to think hard about what it is that is really being modeled, with the result that it is more likely the case that the "right" things are modeled.
- Models that adapt to the individual users should in theory eventually perform better than models that do not.

The disadvantages of the adaptive approach are:

- By adapting the model to the local circumstances, the model may require more computational effort than a fixed, static model.
- Before adaptation has completed, the algorithm may perform poorly.
- The model may be more complex than a fixed, static model designed for the same task.
- Once deployed, the designer has limited control over the resulting adapted models. The designer determines how the adaptation is to be performed..

System	Adapted	Drives adaptation	Gesture type
Swamped!	N/A	N/A	iconic doll manipulation
PHMM	"how" the gesture was conducted	regularity of gesture	gesture families that vary geometrically
Natural gesture parser	rest state appearance	temporal model of natural gesture	natural spontaneous gesture
Watch and Learn	appearance models at each gesture phase	application context	designed; application specific
Bayes Net Watch and Learn	multiple; appearance, motion model	multiple; context, motion, appearance	designed; application specific

Table 1.1: Comparison of the systems presented in this thesis.

1.5 Architectures and Applications

In this thesis I present a number of systems that explore the adaptive gesture models. They each differ along the dimensions of what is adapted, what information is exploited to drive the adaptation process, the nature of the gestures involved, mathematical techniques used, and so on.

An important consideration in the development of adaptive models for gesture recognition is how to encode constraints in the representation so that learning is possible. In the work presented in this thesis we show a number of ways of encoding a variety of constraints related to gesture. For example, we present systems that incorporate prior knowledge regarding task-level expectations in time, prior knowledge of the form of gestures according to natural gesture theory, and prior knowledge regarding the typical kinds of systematic variation seen in training ensembles. Attributes of the systems are summarized in Table 1.1.

1.5.1 Geometric regularity of gesture families: Parametric Hidden Markov Models

A *gesture family* is a set of semantically similar gestures that vary smoothly in response to smooth changes in some parameter. The Parametric Hidden Markov Model (PHMM) is a variation of the hidden Markov model (HMM) that is able to model gesture families. The gesture recognition process involves adapting the value of the deformation parameter in response to the input data. The resulting algorithm may be used to recover "how" a gesture is executed. The adaptation is driven by no external sources, but rather relies on the geometric regularity of the variation within a gesture family.

Chapter 2 presents the PHMM.

1.5.2 Temporal structure of known behavior: Natural Gesture Models

Natural gesture refers to the spontaneous hand movements made by a person telling a story, for example. An algorithm is constructed to recover the appearance of the subject during rest. The algorithm relies on a simple temporal model of natural gesture to drive the adaptation of rest state models without relying on hand tracking, which in many circumstances is difficult and unnecessary.

The natural gesture parsing system is presented in Chapter 3.

1.5.3 Strong contextual information: Watch and Learn

In the *Watch and Learn* system the use of strong contextual priors to adapt appearance based models of gesture is explored. The idea is that if the user follows the context somewhat closely and the context is constrained, then more precise descriptions of the gesture states may be induced. The Baum-Welch algorithm for training HMM is used to train appearance models in an online fashion.

Watch and Learn builds on the natural gesture parsing system by employing a more probabilistic framework and is implemented to work in real time.

Chapter 4 presents the Watch and Learn system.

1.5.4 Multiple Symmetric Learned Representations: Online Learning of Gesture with Bayesian Networks

Shortcomings of the Watch and Learn system are addressed in the final system, which is based on the mathematical foundation of Bayesian networks. The Bayesian network formulation opens up the possibility for adaptation to be driven in multiple ways, and for different directions of adaptation.

As an example, consider a system which incorporates models of hand movement and separate models of hand appearance. Models of hand movement might be derived from studies of how people gesture while telling a story, and hand appearance models might be trained in an offline fashion. Alternatively, hand appearance models could be inferred by observing the appearance of objects which move in a manner consistent with hand movement models. Once the hand is acquired, new movement models may be learned by observing the movement of the hand. The total gesture signal is modeled as the interaction of two factors: an appearance model and a movement model. Bayesian network models are suited for this style of transfer learning by virtue of the fact that some parts (sets of random variables) of the model may be held constant while some others may be learned.

After introducing some background on Bayesian networks in Chapter 5, in Chapter 6 we argue that the hybrid Bayesian network is an appropriate representation for gesture systems where gesture models are learned incrementally. In particular, we are interested in applying this model to the problem of learning models of gesture on the fly, and adapting previous models to new users. On a higher level, we are also interested in ultimately modeling modeling the act of perception as an active,

adaptive process, where the act of seeing involves understanding a novel input in terms of familiar constructs..

1.6 Contributions

The research areas addressed by this thesis are centered around the automatic recognition of gesture from video and other sensors. It is an investigation of the techniques necessary to exploit gesture as a modality in tomorrow's interfaces. In particular, it is concerned with machine learning techniques for adaptive models of gesture: how do we construct gesture recognition systems that are able to adapt to novel situations and users?

The contributions of this thesis include:

- Parameterized hidden Markov models are a compact representation of gesture families. Parameterized hidden Markov models (PHHMs) characterize systematic variation of gesture. For example, a form of a pointing gesture depends on the direction of pointing. An expectation-maximization (EM) framework is developed for learning the variation from a set of examples as well as recovering the form of the gesture during runtime.

- Temporal structure may be used to classify broad classes of gesture without relying on hand position or velocity information. By way of demonstration, a strong temporal model borrowed from research on natural gesture is used to characterize two broad classes of gesture. Appearance models may be inferred by correlating temporal models with a video sequence.

- Online adaptive learning yields robust, flexible gesture recognition. A computational framework similar to the natural gesture parsing system is used in an online fashion in the Watch and Learn system, which adaptively learns how to recognize gesture given a strong temporal model.

- Hybrid Bayesian networks are a useful computational framework in which to embed an extend the above frameworks. Furthermore, they allow flexibility for multiple adaptation scenarios. A Bayesian Network topology is developed that allows multiple adaptation scenarios, tracking of multiple hypotheses, and a variety of motion models that vary in the nature of the constraints on the problem.

These contributions are developed in each of the subsequent chapters.

Chapter 2

Parametric Hidden Markov Models

2.1 Introduction

Current approaches to the recognition of human movement work by matching an incoming signal to a set of representations of prototype sequences. For example, a typical gesture recognition system matches a sequence of hand positions over time to a number of prototype gesture sequences, each of which are learned from a set of examples. To handle variations in temporal behavior, the match is typically computed using some form of dynamic time warping (DTW). If the prototype is described by statistical tendencies, the time warping is often embedded within a hidden Markov model (HMM) framework. When the match to a particular prototype is above some threshold, the system concludes that the gesture corresponding to that prototype has occurred.

Consider, however, the problem of recognizing the gesture pictured in Figure 2-1 that accompanies the speech "I caught a fish. It was *this* big." The gesture co-occurs with the word "this" and is intended to convey the size of the fish, a scalar quantity. The difficulty in recognizing this gesture is that its spatial form varies greatly depending on this quantity. A simple DTW or HMM approach would attempt to model this important relationship as noise. We call movements that exhibit meaningful, systematic variation *parameterized movements*.

In this chapter we will focus on gestures whose *spatial* execution is determined by the parameter, as opposed to, say, the temporal properties. Many hand gestures that accompany speech are so parameterized. As with the "fish" example, hand gestures are often used in dialog to convey some quantity that otherwise cannot be determined from speech alone; it is the spatial trajectory or configuration of the hands that reflect the quantity. Examples include gestures indicating size, rotation, or direction.

Techniques that use fixed prototypes for matching are not well suited to modeling movements that exhibit such meaningful variation. In this chapter we present a framework which models spatially parameterized movements in a such way that the recovery of the parameter of interest and the computation of likelihood proceed simultaneously. This ability allows the construction of more accurate recognition systems.

Figure 2-1: The gesture that accompanies the speech "I caught a fish. It was *this* big." In its entirety, the gesture consists of a preparation phase in which the hands are brought into the gesture space, a stroke phase (depicted by the illustration) which co-occurs with the word "this" and finally a retraction back to the rest-state (hands down and relaxed). The distance between the hands conveys the size of the fish.

We begin by extending the standard hidden Markov model method of gesture recognition to include a global parametric variation in the output probabilities of the states of the HMM. Using a linear model of the relationship between the parametric gesture quantity (for example, size) and the means of probability density functions of the parametric HMM (PHMM), we formulate an expectation-maximization (EM) method for training the PHMM. During testing, a similar EM algorithm allows the simultaneous computation of the likelihood of the given PHMM generating the observed sequence and estimation of the quantifying parameters. Using visually-derived and directly measured 3-dimensional hand position measurements as input, we present results on several movements that demonstrate the superiority of PHMMs over standard HMMs in recognizing parametric gestures and show improved robustness in estimating the quantifying parameter with respect to noise in the input features.

Lastly, we present an extension of the framework to handle situations in which the dependence of the state output distributions on the parameters is not linear. Nonlinear PHMMs model the dependence using a 3-layer logistic neural network at each state. This model removes the constraint that the mapping from parameterization to output densities be linear; rather only a smooth mapping is required. The nonlinear PHMM is thus able to model a larger class of gesture and movement than the linear PHMM, and by the same token, the parameterization may be chosen more freely in relation to the observation feature space. The disadvantage of the non-linear map is that closed-form maximization of each iteration of the EM algorithm is no longer possible. Instead, we derive a generalized EM (GEM) technique based upon the gradient of the probability with respect to the parameter to be estimated[1].

[1]Parts of this chapter were previously published in [76, 75, 72, 81]

2.2 Motivation and Prior Work

2.2.1 Using HMMs in gesture recognition

Hidden Markov models and related techniques have been applied to gesture recognition tasks with success. Typically, trained models of each gesture class are used to compute each model's similarity to some novel input sequence. The input sequence could be the last few seconds of data from a variety of sensors, including hand position data derived using computer vision techniques or other position tracking methods. Typically, the classification of the input sequence proceeds by computing the sequence's similarity to each of the gesture class models. If probabilistic techniques are used, these similarity measures take the form of likelihoods. If the similarity to any gesture is above some threshold, then the sequence is classified as the gesture for which the similarity is greatest.

A typical problem with these techniques is determining when the gesture began without classifying each subsequence up to the current time. One solution is to use dynamic programming to match the sequence against a model from all possible starting times of the gesture to the current time. The best starting time is then chosen from all possible starting times to give the best match average over the length of the gesture. Dynamic time warping (DTW) and Hidden Markov models (HMMs) are two techniques based on dynamic programming. Darrell and Pentland [21] applied DTW to match image template correlation scores against models to recognize hand gestures from video. In previous work [9], we represented gesture as a deterministic sequence of states through some configuration or feature space, and employed a DTW parsing algorithm to recognize the gestures. The states were found by first determining a prototype gesture from a set of examples, and then creating a set of states in feature space that spanned the training set.

Rather than model a prototype sequence, HMMs model a stochastic sequence of states to represent gesture. Yamato [84] first used HMMs in vision to recognize tennis strokes. Schlenzig, Hunter and Jain [64] used HMMs and a rotation-invariant image representation to recognize hand gestures from video. Starner and Pentland [66] applied HMMs to recognize ASL sentences, and Campbell et al. [14] used HMMs to recognize Tai Chi movements. The present work is based on the HMM framework, which we summarize in the appendix.

None of the approaches mentioned above consider the effect of a systematic variation of the gesture on the underlying representation: the variation between instances is treated as noise. When it is too difficult to approximate the noise, or the noise is systematic, is often effective to look for diagnostic features. For example, in [80] we employed HMMs that model the temporal properties of movement to recognize two broad classes of natural, spontaneous gesture. These models were constructed in accordance with natural gesture theory [47, 17]. Campbell and Bobick [15] search for orthogonal projections of the feature space to find the most diagnostic projections in order to classify ballet steps. In each of these cases the goal is to eliminate the systematic variation rather than to model it. The work presented here introduces a new method for modeling such variation within an HMM paradigm.

2.2.2 Modeling parametric variations

In many gesture recognition contexts, it is desirable to extract some auxiliary information as well as recognize the gesture. An interactive system might need to know in which direction a user points as well as recognize that the user pointed. In human communication, sometimes *how* a gesture is performed carries significant meaning. ASL, for example, is subject to complex grammatical processes that operate on multiple simultaneous levels [59].

One approach is to explicitly model the space of variation exhibited by a class of signals. In [74], we apply HMMs to the task of hand gesture recognition from video by training an eigenvector basis set of the images at each state. An image's membership to each state is a function of the residual of the reconstruction of the image using the state's eigenvectors. The state membership is thus invariant to variance along the eigenvectors. Although not applied to images directly, the present work is an extension of this earlier work in that the goal is to recover a parameterization of the systematic variation of the gesture.

Yacoob and Black [83] as well as Bobick and Davis [7] model the variation within a class of human movement using linear principle components analysis. The space of variation is defined by a single linear transformation on the whole movement sequence. They apply their technique to show more robust recognition in the face of varying walking direction and style.

Murase and Nayar [49] parameterize meaningful variation in the appearance of images by computing a representation of the nonlinear manifold of the images in an eigenspace of the images. Their work is similar to ours in that training assumes that each input feature vector is labeled with the value of the parameterization. In testing, an unknown image is projected onto the manifold and the parameterization is recovered. Their framework has been used, for example, to recover the camera angle relative to a known object in the field of view.

Recently there has been interest in methods that discover parameterizations in an unsupervised way (so-called *latent* parameterizations). In his "family discovery" paradigm, Omohundro [55], for example, outlines a variety of approaches to learning a nonlinear manifold in some feature space representing systematic variation. One of these techniques has been applied to the task of lip reading by Bregler and Omohundro [12]. Bishop, Svensen and Williams [4] have also introduced techniques to learn latent parameterizations. Their system begins with an assumption of the dimensionality of the parameterization and uses an expectation-maximization framework to compute a manifold representation. The present work is similarly concerned with modeling "families" of signals but assumes that the parameterization is given for the training set.

Lastly, we mention that in the speech recognition community a number of models for speaker adaptation in HMM-based speech recognition systems have been proposed. Gales [24] for example, examines a number transformations on the means and covariances of HMM output distributions. These transformations are trained against a new speaker speaking a known utterance. Our model is similar in that we use constrained transformations of the model to match the data, but differs in that we

are interested in recovering the value of a meaningful parameter as the input occurs, rather than simply adapting to a known input during a training phase.

2.2.3 Non-parametric extensions

Before presenting our method for modeling parameterized movements, it is worthwhile to consider a number of extensions of the standard gesture recognition paradigm that attempt to address the problem of recognizing these parameterized classes.

The first approach relies on our ability to come up with *ad hoc* methods to extract the value of the parameter of interest. For the example of the fish-size gesture presented in Figure 2-1, one could design a procedure to recover the parameter: wait until the hands are in the middle of the gesture space and have low velocity, then calculate the distance between the hands. Similar approaches are used the ALIVE [20] and Perseus [40] systems. The typical approach of these systems is to first identify static configurations of the user's body that are diagnostic of the gesture, and then use an unrelated method to extract the parameter of interest (for example, direction of pointing). Manually constructed *ad hoc* procedures are typically used to identify the diagnostic configuration, a task complicated by the requirement that this procedure work through the range of meaningful variation and also not be confused by other gestures. Perseus, for example, understands pointing gestures by detecting when the user's arm is extended. The system then finds the pointing direction by computing the line from the head to the user's hand.

The chief objection to such an approach is not that each movement requires a new *ad hoc* procedure, nor the difficulty in writing procedures that recover the parameter robustly, but the fact that *they are only appropriate to use when the gesture has already been labeled*. As mentioned in the introduction, a recognition system that abstracts over the variation induced by the parameterization must model such variation as noise or deviation from a prototype. The greater the parametric variation, the less constrained the recognition prototype can be, and the worse the detection results become.

The second approach employs multiple DTW or HMM models to cover the parameter space. Each DTW model or HMM is associated with a point in parameter space. In learning, the problem of allocating training examples labeled by a continuous variable to one of a discrete set of models is eliminated by uniting the models in a mixture of experts framework [33]. In testing, the parameter is extracted by finding the best match among the models and looking up its associated parameter value. The dependency of the movement's form on the parameter is thus removed.

The most serious objection to this approach is that as the dimensionality of the parameter space increases, the large number of models necessary to cover the space will place unreasonable demands on the amount of training data.[2] For example,

[2]In such a situation it is not sufficient to simply interpolate the match scores of just a few models in a high dimensional space since either (1) there will be significant portions of the space for which there is no response from any model or (2) in a mixture of experts framework, each model is called on to model too much of the space, and so is modeling the dependency on the parameter as noise.

to recover a 2-dimensional parameter with 4 bits of accuracy would theoretically require 256 distinct HMMs (assuming no interpolation). Furthermore with such a set of distinct HMMs, all of the models are required to learn the same or similar dynamics (i.e. as modeled by the transition matrix in the case of HMMs) separately, increasing the amount of training data required. This can be embellished somewhat by computing the value of the parameter as the weighted average of all the models' associated parameter values, where the weights are derived from the matching process.

Lastly, we consider the possibility of directly interpolating the parameters of multiple HMMs to arrive at a specialized HMM to match the input. In fact, in a broad sense the current work adopts this approach. The direct interpolation of HMM parameters however does not address how the multiple HMM would be trained when the examples are spread throughout the space of variation, nor does it result in a compact representation in which parameters are trained using all examples. We present a unified approach derived from the usual HMM techniques that incorporates training and testing using a compact representation (a single HMM) and an optimization algorithm to match.

In the next section we introduce parametric HMMs, which overcome the problems with the approaches presented above.

2.3 Parametric hidden Markov models

2.3.1 Defining parameterized gesture

Parametric HMMs explicitly model the dependence on the parameter of interest. We begin with the usual HMM formulation [62] and change the form of the output probability distribution (usually a normal distribution or a mixture model) to depend on the gesture parameter to be estimated.

As in previous approaches to gesture recognition, we assume that a given gesture sequence is modeled as being generated by a first-order Markov finite state machine. The state that the machine is in at time t and its output are denoted q_t and $\mathbf{x}_t$, respectively. The Markov property is encoded by a set of transition probabilities, with $a_{ij} = P(q_t = j \mid q_{t-1} = i)$ the probability of moving to state j at time t given the system was in state i at time $t-1$. In a continuous density HMM an output probability density $b_j(\mathbf{x}_t)$ associated with each state j gives the probability of the feature vector $\mathbf{x}_t$ given the system is in state j at time t: $P(\mathbf{x}_t \mid q_t = j)$. Of course, the actual state of the machine at any given time is unknown or *hidden*.

Given a set of training data — sequences known to be generated by a single machine — the parameters of the machine need to be estimated. In a simple Gaussian HMM, the parameters are the a_{ij}, $\hat{\boldsymbol{\mu}}_j$, and $\boldsymbol{\Sigma}_j$.[3]

We *define* a parameterized gesture to be one in which the output densities $b_j(\mathbf{x}_t)$ are a function of the gesture parameter vector $\boldsymbol{\theta}$: $b_j(\mathbf{x}_t; \boldsymbol{\theta})$. The dimension of $\boldsymbol{\theta}$

[3]The initial state distribution π_j is usually also estimated; in this work we use causal topologies with a unique starting state.

matches that of the degree of freedom of the gesture. For the fish size gesture it would be a scalar; for indicating a direction in space, $\boldsymbol{\theta}$ would have two dimensions.

Note that our definition of parameterized gesture only models the spatial (or more general feature) variation, and not temporal variation. Our primary reason for this is that the Viterbi parsing algorithm of the HMMs essentially performs a dynamic time warp of the input signal. In fact, part of the appeal of HMMs for gesture recognition is its insensitivity to temporal variation. Unfortunately, this property means that it is difficult to restrict the nature of the temporal variation (for example a linear scaling or uniform speed change). Recently, Yacoob and Black [83] derive a method for recognizing global temporal deformations of an activity; their method does not however represent the explicit spatial parameter variation.

Also, although $\boldsymbol{\theta}$ is a global parameter — it affects all states — the actual effect varies state to state. Therefore the effect of $\boldsymbol{\theta}$ is local and will be set to maximize the total probability of the training set. As we will show in the experiments, if some state is best left unperturbed by $\boldsymbol{\theta}$ the magnitude of the effect will automatically become small.

2.3.2 Linear model

To realize the parameterization on $\boldsymbol{\theta}$ we modify the output densities. The simplest useful model is a linear dependence of the mean of the Gaussian on $\boldsymbol{\theta}$. For each state j of the HMM we have:

$$\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}) = W_j\boldsymbol{\theta} + \bar{\boldsymbol{\mu}}_j \tag{2.1}$$

$$P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t, \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}), \Sigma_j) \tag{2.2}$$

where the columns of the matrix W_j span a d dimensional hyper-plane in feature space where d is the dimension of $\boldsymbol{\theta}$. For the fish size gesture, if $\mathbf{x}_t$ is embedded in a six-dimensional space (e.g. the three-dimensional position of each of the hands) then the dimension of W_j would be 6x1, and would represent the one dimensional hyper-plane (a line in six-space) along which the mean of the output distribution moves as $\boldsymbol{\theta}$ varies. For a pointing gesture (two degrees of freedom) of one hand (a feature space of three dimensions), W would be 3x2. The magnitude of the columns of W reflect how much the mean of the density translates as the value of different components of $\boldsymbol{\theta}$ vary.

For a Bayesian estimate of $\boldsymbol{\theta}$ given an observed sequence we would need to specify a prior distribution on $\boldsymbol{\theta}$. In the work presented here we assume the distribution of $\boldsymbol{\theta}$ is finite-uniform implying that the value of the prior $P(\boldsymbol{\theta})$ for any particular $\boldsymbol{\theta}$ is either a constant or zero. We therefore can ignore it in the following derivations and simply use bounds checking during testing to make sure that the recovered $\boldsymbol{\theta}$ is plausible, as indicated by the training data.

Note that $\boldsymbol{\theta}$ is constant for the entire observation sequence, but is free to vary from sequence to sequence. When necessary, we write the value of $\boldsymbol{\theta}$ associated with a particular sequence k as $\boldsymbol{\theta}_k$.

For readers familiar with graphical model representations of HMMs (for example, see [3]), Figure 2-2 shows the PHMM architecture as a Bayes network. The diagram

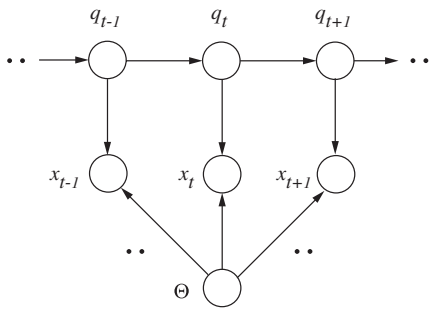

Figure 2-2: Bayes network showing the conditional dependencies of the PHMM.

makes explicit the fact that the output nodes (labeled $\mathbf{x}_t$) depend upon $\boldsymbol{\theta}$. Bengio and Frasconi's [2] Input Output HMM (IOHMM) is a similar architecture that maps input sequences to output sequences using a recurrent neural net, which, by the Markov assumption, needs only consider the current and previous time steps of the input and output. The PHMM architecture differs in that it maps a single parameter value to an entire sequence. Thus the parameter provides a *global* constraint on the sequences, and so the PHMM testing phase must consider the entire sequence at once. Later, we show how this feature provides robustness to noise.

2.3.3 Training

Within the HMM paradigm of recognition, training entails using known, segmented examples of the gesture sequence to estimate the HMM parameters. The Baum-Welch form of the expectation-maximization (EM) algorithm is used to update the parameters such that the probability that the HMM would produce the training set is maximized. For the PHMM training is similar except that there are the additional parameters W_j to be estimated, and the value of $\boldsymbol{\theta}$ must be given for each training sequence. In this section we derive the EM update equations necessary to estimate the additional parameters. An appendix provides a brief description of the Baum-Welch algorithm; for a comprehensive discussion see [62].

The *expectation* step of the Baum-Welch algorithm (also known as the "forward/backward" algorithm) computes the probability that the HMM was in state j at time t given the entire sequence $\mathbf{x}$; the probability is denoted as γ_{tj}. It is convenient to consider the HMM parse of the observation sequence as being represented by the matrix of values γ_{tj}. The forward component of the algorithm also computes the likelihood of the observed sequence given the particular HMM.

Let the set of parameters of the HMM be written as ϕ; these parameters are updated in the *maximization* step of the EM algorithm. In particular, the parameters ϕ are updated by choosing a ϕ', a subset of ϕ, to maximize the auxiliary function $Q(\phi' \mid \phi)$. As explained in the appendix, Q is the expected value of the log probability given the parse γ_{tj}. ϕ' may contain all the parameters in ϕ, or only a subset if several maximization steps are required to estimate all the parameters. In the appendix we

derive the derivative of Q for HMMs:

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')} \tag{2.3}$$

The parameters ϕ of the *parameterized* Gaussian HMM include W_j, $\bar{\boldsymbol{\mu}}_j$, Σ_j and the Markov model transition probabilities a_{ij}. Updating W_j and $\bar{\boldsymbol{\mu}}_j$ separately has the drawback that when estimating W_j only the old value of $\bar{\boldsymbol{\mu}}_j$ is available, and similarly if $\bar{\boldsymbol{\mu}}_j$ is estimated first, W_j is unavailable. Instead, we define new variables:

$$Z_j \equiv \left[\begin{array}{cc} W_j & \bar{\boldsymbol{\mu}}_j \end{array} \right] \quad \Omega_k \equiv \left[\begin{array}{c} \boldsymbol{\theta}_k \\ 1 \end{array} \right] \tag{2.4}$$

such that $\hat{\boldsymbol{\mu}}_j = Z_j \Omega_k$. We then need only update Z_j in the maximization step for the means.

To derive an update equation for Z_j we maximize Q by setting equation 2.3 to zero (selecting Z_j as the parameters in ϕ') and solving for Z_j. Note that because each observation sequence k in the training set is associated with a particular $\boldsymbol{\theta}_k$, we can consider all observation sequences in the training set before updating Z_j. Accordingly we denote γ_{tj} associated with sequence k as γ_{ktj}. Substituting the Gaussian distribution and the definition of $\hat{\boldsymbol{\mu}}_j = Z_j \Omega_k$ into equation 2.3:

$$\begin{aligned}
\frac{\partial Q}{\partial Z_j} &= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \frac{\partial}{\partial Z_j} \left(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k) \right)^T \Sigma_j^{-1} \left(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k) \right) && (2.5) \\
&= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \frac{\partial}{\partial Z_j} \left[\mathbf{x}_{kt}{}^T \Sigma_j^{-1} \mathbf{x}_{kt} - 2 \hat{\boldsymbol{\mu}}_j^T \Sigma_j^{-1} \mathbf{x}_{kt} + \hat{\boldsymbol{\mu}}_j^T \Sigma_j^{-1} \hat{\boldsymbol{\mu}}_j \right] \\
&= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \left[-2 \frac{\partial}{\partial Z_j} \left(Z_j \Omega_k \right)^T \Sigma_j^{-1} \mathbf{x}_{kt} + \frac{\partial}{\partial Z_j} \left(Z_j \Omega_k \right)^T \Sigma_j^{-1} Z_j \Omega_k \right] \\
&= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \left[-2 \frac{\partial}{\partial Z_j} \Omega_k{}^T Z_j{}^T \Sigma_j^{-1} \mathbf{x}_{kt} + \frac{\partial}{\partial Z_j} \Omega_k{}^T \left(Z_j{}^T \Sigma_j^{-1} Z_j \right) \Omega_k \right] \\
&= \Sigma_j^{-1} \sum_k \sum_t \gamma_{ktj} \left[\mathbf{x}_{kt} \Omega_k{}^T - Z_j \Omega_k \Omega_k{}^T \right] && (2.6)
\end{aligned}$$

where we use the identity $\frac{\partial}{\partial M} \mathbf{a}^T M \mathbf{b} = \mathbf{a}\mathbf{b}^T$. Setting this derivative to zero and solving for Z_j, we get the update equation for Z_j:

$$Z_j = \left[\sum_{k,t} \gamma_{ktj} \mathbf{x}_{kt} \Omega_k^T \right] \left[\sum_{k,t} \gamma_{ktj} \Omega_k \Omega_k^T \right]^{-1} \tag{2.7}$$

Once the means are estimated, the covariance matrices Σ_j are updated in the usual way:

$$\Sigma_j = \sum_{k,t} \frac{\gamma_{ktj}}{\sum_t \gamma_{ktj}} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))^T \tag{2.8}$$

as is the matrix of transition probabilities [62] (see also the appendix).

2.3.4 Testing

Recognition using HMMs requires evaluating the probability that a given HMM would generate an observed input sequence. Recognizing a sequence consists of evaluating this probability (known as the *likelihood*) of the sequence for each HMM, and, assuming equal priors, selecting the HMM with the greatest likelihood. With PHMMs the probability is defined to be the maximum probability with respect to the possible values of $\boldsymbol{\theta}$. Compared to the usual HMM formulation, the parameterized HMMs testing procedure is complicated by the dependence of the parse on the unknown $\boldsymbol{\theta}$.

We desire the value of $\boldsymbol{\theta}$ which maximizes the probability of the observation sequence. Again an EM algorithm is appropriate: the expectation step is the same forward/backward algorithm used in training. The estimation component of the forward/backward algorithm computes both the parse γ_{tj} and the probability of the sequence, given a value of $\boldsymbol{\theta}$. In the corresponding maximization step we update $\boldsymbol{\theta}$ to maximize Q, the log probability of the sequence given the parse γ_{tj}. In the training algorithm we knew $\boldsymbol{\theta}$ and estimated all the parameters of the HMM; in testing we fix the parameters of the machine and maximize the probability with respect to $\boldsymbol{\theta}$.

To derive an update equation for $\boldsymbol{\theta}$, we start with the derivative in equation 2.3 from the previous section and select $\boldsymbol{\theta}$ as ϕ'. As with Z_j, only the means $\hat{\boldsymbol{\mu}}_j$ depend upon $\boldsymbol{\theta}$ yielding:

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} = \sum_t \sum_j \gamma_{tj} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{2.9}$$

Setting this derivative to zero and solving for $\boldsymbol{\theta}$, we have:

$$\boldsymbol{\theta} = \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} W_j \right]^{-1} \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} (\mathbf{x}_t - \bar{\boldsymbol{\mu}}_j) \right] \tag{2.10}$$

The values of γ_{tj} and $\boldsymbol{\theta}$ are iteratively updated until the change in $\boldsymbol{\theta}$ is small. With the examples we have tried, less than ten iterations are sufficient. Note that for efficiency, many of the inner terms of the above expression may be cached. As mentioned in the training derivation, the forward component of the expectation step also computes the probability of the observed sequence given the PHMM. That probability is the (local) maximum probability with respect to $\boldsymbol{\theta}$ and is used by the recognition system.

Recognition using PHMMs proceeds by computing for each PHMM the value of $\boldsymbol{\theta}$ that maximizes the likelihood of the sequence. The PHMM with the highest likelihood is selected. As we demonstrate in section 2.4.2 in some cases it may be possible to classify the sequence by the value of $\boldsymbol{\theta}$ as determined by a single PHMM.

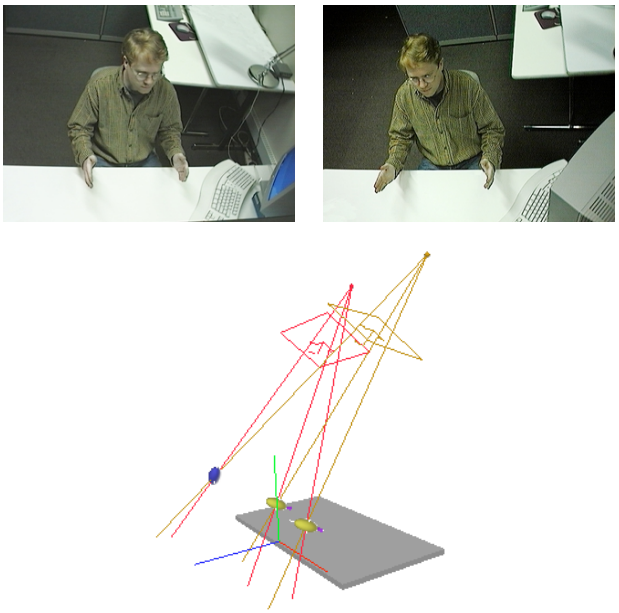

Figure 2-3: The Stereo Interactive Virtual Environment (STIVE) computer vision system used to collect data in section 2.4.1. Using flesh tracking techniques, STIVE computes the three-dimensional position of the head and hands at a frame rate of about 20Hz. We used only the position of the hands for the first two experiments.

2.4 Results of Linear Model

This section presents three experiments. The first — the example discussed in the introduction: "I caught a fish. It was *this* big." — demonstrates the ability of the testing EM algorithm to recover the gesture parameter of interest. The second compares PHMMs to standard HMMs in a to gesture recognition task to demonstrate a PHMM's ability to better model this type gesture. The final experiment — a pointing gesture — displays the robustness of the PHMM to noise in estimating the gesture parameter $\boldsymbol{\theta}$.

2.4.1 Experiment 1: Size gesture

To test the ability of the parametric HMM to learn the parameterization, thirty examples of the type depicted in Figure 2-1 were collected using the Stereo Interactive Virtual Environment (STIVE)[1], a research computer vision system utilizing wide baseline stereo cameras and flesh tracking (see Figure 2-3). STIVE is able to compute the three-dimensional position of the head and hands at a frame rate of about 20Hz. The input to the gesture recognition system is a sequence of six-dimensional vectors representing the Cartesian location of each of the hands at each time step.

The 30 sequences averaged about 43 samples in length. The actual value of $\boldsymbol{\theta}$, which in this case is interpreted the size in inches, was measured directly by finding the point in each sequence during which the hands were stationary and then computing the distance between the hands. The value of $\boldsymbol{\theta}$ varied from 7.7 inches (a small

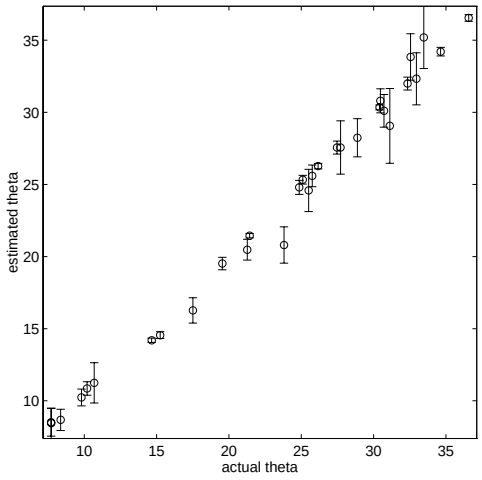

Figure 2-4: Parameter estimation results for the size gesture. Fifty random choices of the test and training sets were used to compute mean and standard deviation (error bars) on all examples. The PHMM was retrained for each choice of test and training set.

fish) to 36.6 inches (a respectable catch). This method of assessing $\boldsymbol{\theta}$ is used as the known value for training examples, and for the "ground truth" in evaluating testing performance. For this experiment, both the training and the testing data were manually segmented; in experiment 3 we demonstrate the PHMMs performing segmentation on an unsegmented stream of data containing multiple gestures.

A PHMM was trained with fifteen sequences randomly selected from the pool of thirty; we used six states as determined by cross validation. The topology of the PHMM was set to be causal (i.e., no transitions to previously visited states, with no "skip transitions" [62]). In this example typically ten iterations were required for convergence, when the relative change in the total log probability for the training examples was less than one part in one thousand.

Testing was performed with the remaining fifteen sequences. As described above, the size parameter $\boldsymbol{\theta}$ was extracted from each of the testing sequences via the EM algorithm that estimates the probability of the sequence. We calculated the difference between the estimated value of $\boldsymbol{\theta}$ and the value computed by direct measurement.

Figure 2-4 shows statistics on the parameter estimation for 50 random choices of the test and training sets. The PHMM was retrained for each choice of test and training set. The average absolute error over all test trials is about 0.16 inches, demonstrating that the PHMM has learned the parameterization accurately. The experiment demonstrates the validity of using the EM algorithm which maximizes output likelihood as a mechanism for recovering $\boldsymbol{\theta}$.

It is interesting to consider the recovered W_j. Recall that for this example W_j is a 6x1 vector whose direction indicates the linear path in six-space along which the mean $\hat{\boldsymbol{\mu}}_j$ moves as $\boldsymbol{\theta}$ varies; the magnitude of W_j reflects the sensitivity of the mean to variation in $\boldsymbol{\theta}$. Table 2.4.1 gives the magnitude of the six W_j vectors for this experiment. The absolute scale of W_j is determined by the units of the feature

measurements and the units of the gesture quantity $\boldsymbol{\theta}$. But the relative scale of the W_j demonstrates that the mean of the middle states (for example, 3 and 4) are more sensitive to $\boldsymbol{\theta}$ than either the initial or final states. Figure 2-5 show how the position of the states depends on $\boldsymbol{\theta}$. This agrees with our intuition: the hands always start and return to the body; the states that represent the maximal extent of the hands need to accommodate the variation in $\boldsymbol{\theta}$. *The system automatically learns which segment of the gesture is most diagnostic of* $\boldsymbol{\theta}$.

State j	1	2	3	4	5	6
$\|W_j\|$	0.062	0.187	0.555	0.719	0.503	0.134

Table 2.1: The magnitude of W_j is greater for the states that correspond to where the hands are maximally extended (3 and 4). The position of these states is most sensitive to $\boldsymbol{\theta}$, in this case the size of the fish.

2.4.2 Experiment 2: Recognition

Our second experiment is designed to illustrate the utility of PHMMs in the recognition of gesture. We compare the performance of the PHMM to that of the standard HMM approach, and demonstrate how the ability of the PHMM to model systematic variation allows it to have smaller (and more correct) estimates of noise.

Consider two variations of a pointing gesture: one in which the hand moves straight away from the body at some angle, and another in which the hand moves from the body with some angle and then changes direction midway through the gesture. The latter gesture might co-occur with the speech "*you*, go over *there*". The first gesture we will call *point* and the second *direct*. *Point* gestures are parameterized by the angle of pointing direction (one parameter), while *direct* gestures are parameterized by the initial pointing angle to select an object and an angle to indicate the object's direction of movement (two parameters). In this experiment we show that two HMM's are inadequate to distinguish instances of the *point* family from instances of the *direct* family, while a single PHMM is able to represent both families and classify instances of each.

We collected 40 examples of each gesture class with a Polhemus motion capture system, recording the horizontal and depth components of hand-position. The subject was positioned at arm's length away from a display. For each *point* example, the subject started with hands at rest and then pointed to a target on the display. The target would appear from between 25° to the left of center and 25° to the right of center along a horizontal line on the display. The training set was collected to evenly sample the interval $\theta = [-25, 25]$. For each *direct* example, the subject similarly pointed initially at a target "X" and then midway through the gesture switched to pointing at a target "O". Each "X" was again presented anywhere from $\theta_1 = 25°$ to the left to 25° to the right on the horizontal line. The "O" was presented at $\theta_2°$, drawn from the same range of angles, but in which the absolute difference between θ_1 and θ_2 was at least 10°. This restriction prevented any *direct* gesture from looking like a *point* gesture.

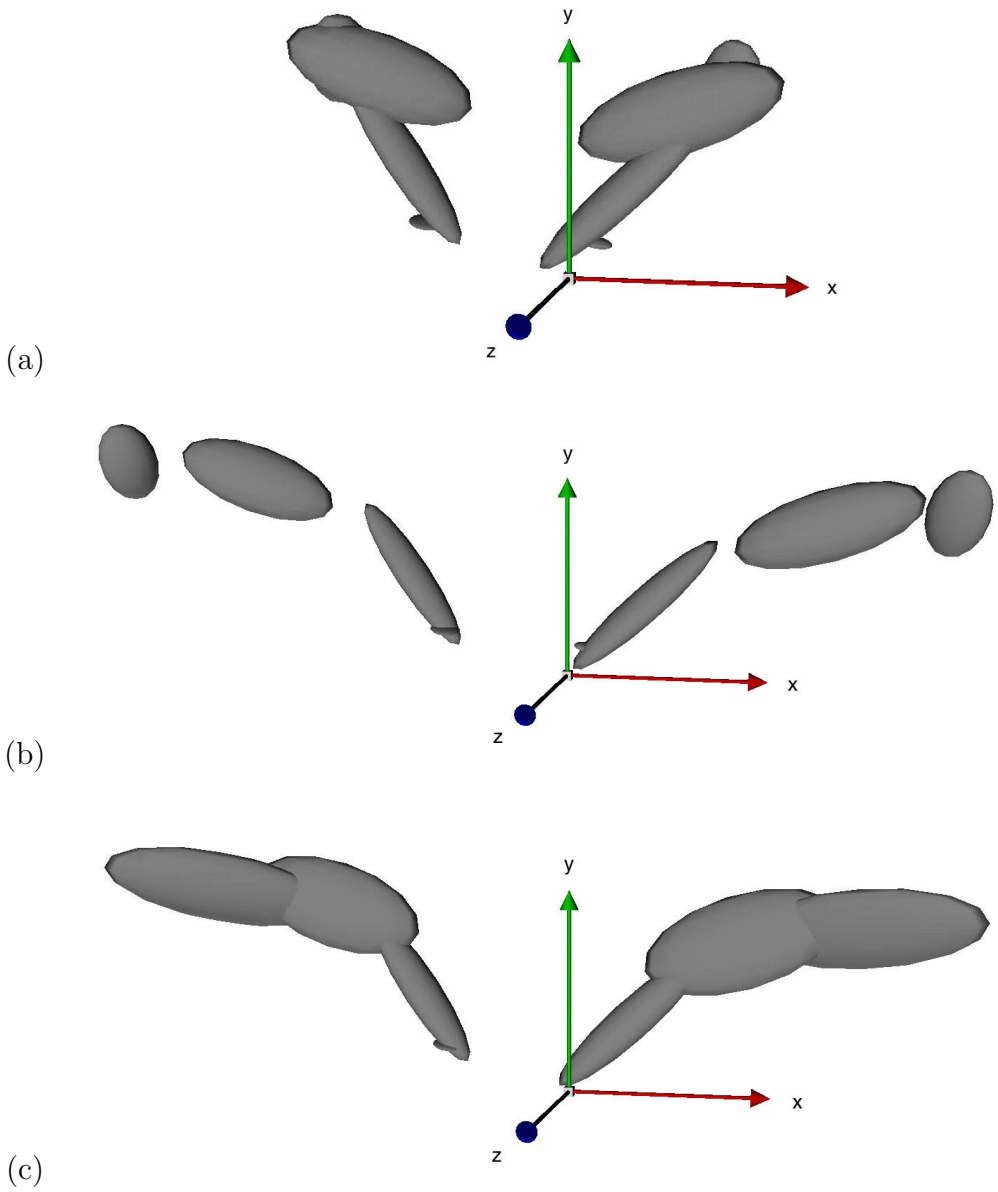

Figure 2-5: The state output density of the two-handed fish-size gesture. Each corresponds to either left or right hand position at a state (for clarity, only the first four states are shown); (a) PHMM, $\boldsymbol{\theta} = 19.0$, (b) PHMM, $\boldsymbol{\theta} = 45.0$, (c) HMM. The ellipsoid shapes for the left hand is derived from the upper 3x3 diagonal block of the full covariance matrices, and the lower 3x3 diagonal block for the right hand. An animation of the fish-size PHMM is located at `http://www.media.mit.edu/~drew/movies`.

Thirty of each set of sequences were used to train an HMM for each gesture class. With 4-state HMMs, a recognition performance of 60% was achieved on the set of 20 test sequences. With 20 states, this performance improved to only 70%.

Next a PHMM was trained using all training examples of both gesture classes. The PHMM was parameterized by two variables θ_1 and θ_2. For each *direct* example, θ_1 and θ_2 were set to equal the angles used in driving the display to collect the examples. For each *point* example, both θ_1 and θ_2 were set to equal the value of the single angle used in collection. By using the same values used in driving the display during collection, the use of an *ad hoc* technique to label the training examples was avoided.

To classify each of the 20 testing examples it suffices to compare the value of θ_1 and θ_2 recovered by the PHMM testing algorithm. We used the single PHMM trained as above to recover parameter values. A training example was classified as a *direct* gesture if the absolute difference in the recovered values θ_1 and θ_2 was more than 5°. With this classification scheme, perfect recognition performance was achieved with a 4-state PHMM, where 2 HMMs could only achieve a 70% recognition rate. The mean error of the recovered values of θ_1 and θ_2 was about 4°. The confusion matrices for the HMM and PHMM models are shown in Figure 2-6.

4-state HMMs

	point	*direct*
actual *point*	8	2
actual *direct*	6	4

20-state HMMs

	point	*direct*
point	10	0
direct	6	4

4-state PHMM

	point	*direct*
point	10	0
direct	0	10

Figure 2-6: Confusion matrices for the *point* and *direct* gesture models. Row headings are the ground truth classifications.

The difference in performance between the HMM and PHMM is due to the fact that the HMM models the systematic variation of each class of gestures as noise. The PHMM is able to distinguish the two classes by recovering the systematic variation present in both classes. Figures 2-7a and 2-7b display the 1.0σ ellipsoids of the Gaussian densities of the states of the PHMM; 2-7a is for $\theta = (15^\circ, 15^\circ)$, 2-7b is for $\boldsymbol{\theta} = (15^\circ, -15^\circ)$. Notice how the position of the means has shifted. Figure 2-7c and d display the 1.0σ ellipsoids for the states of the conventional HMM.

Note that in Figures 2-7c and d the ellipsoids corresponding to each state show how the HMM spans the examples for varying values of the parameter. The PHMM explicitly models the effects of the parameter. It is this ability of the the PHMM to more accurately model parameterized gesture that enhances its recognition performance.

2.4.3 Experiment 3: Robustness to noise, bounds on $\boldsymbol{\theta}$

In our final experiment using the linear model we demonstrate the performance of the PHMM technique under varying amounts of noise, and show robustness in the extraction of the parameter $\boldsymbol{\theta}$. We also demonstrate using the bounds of the uniform distribution of $\boldsymbol{\theta}$ to enhance the recognition capability of the PHMM.

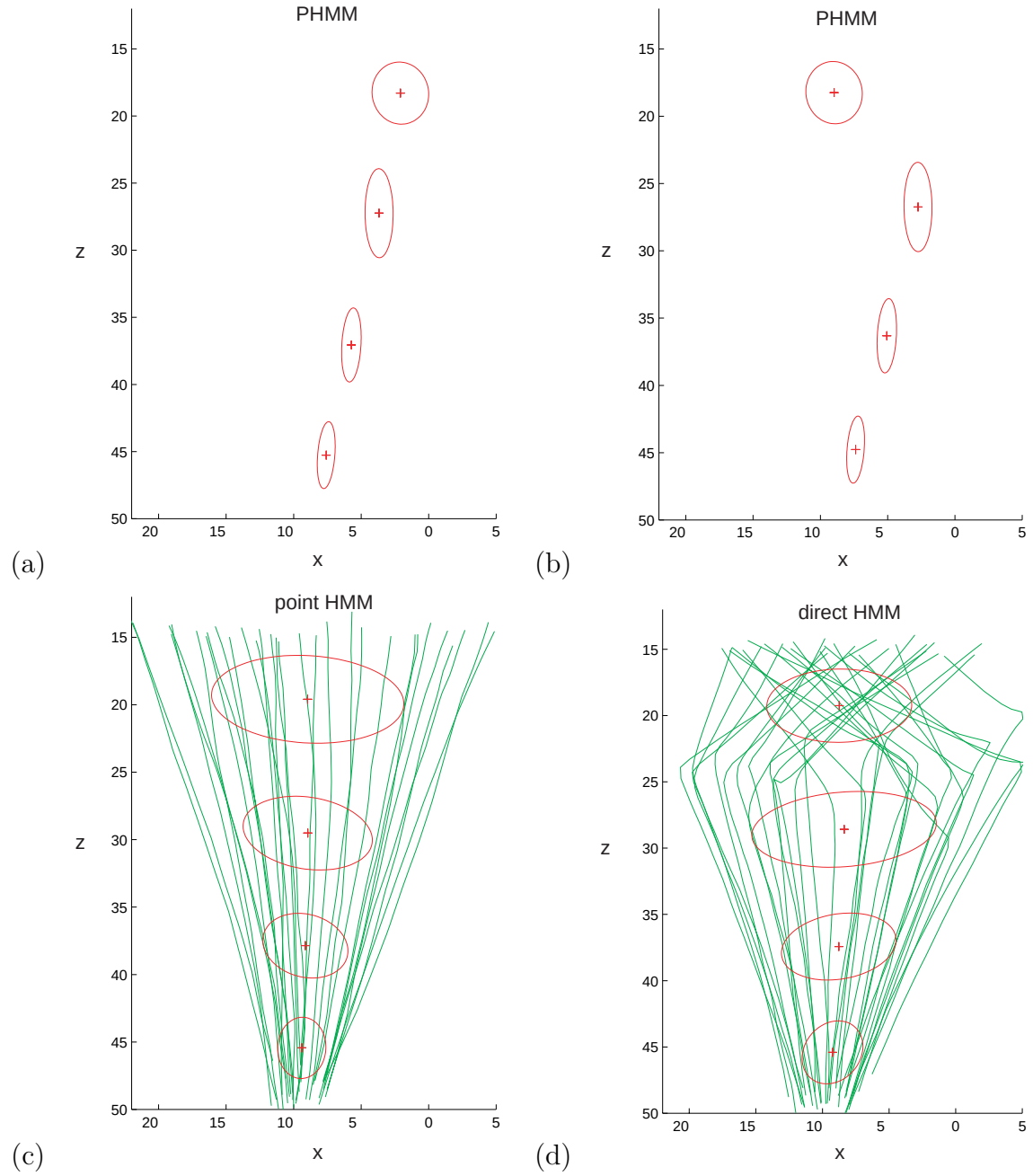

Figure 2-7: The state output densities of the *point* and *direct* gesture models. (a) PHMM $\theta = (15°, 15°)$, (b) PHMM $\theta = (15°, -15°)$, (c) *point* HMM with training set sequences shown, (c) *direct* HMM with training set sequences.

Pointing gesture

Another gesture that requires multi-dimensional parameterization is three-dimensional pointing. Our feature space is the three-dimensional Cartesian position of the wrist as measured by a Polhemus motion capture system. $\boldsymbol{\theta}$ is a two-dimensional vector reflecting the direction of pointing. If the pointing direction is restricted to the hemisphere in front of the user, the movement can be parameterized by the $\boldsymbol{\theta} = (x, y)$ position in a plane in front of the user (see Figure 2-8). This choice of parameterization is consistent with requirement that the parameter be linearly related to the feature space.

The Polhemus system records wrist position at a rate of 30Hz. Fifty pointing gesture examples were collected, each averaging 29 time samples (about 1 second) in length. As ground truth, we again directly measured the value of $\boldsymbol{\theta}$ for each sequence: the point at which the depth of the wrist away from the user was found to be greatest. The position of this point in the pointing plane was returned. The horizontal coordinate of the pointing target varied from -22 to +27 inches, while the vertical coordinate varied from -4 to +31 inches.

An eight state causal PHMM was trained using twenty sequences randomly selected from the pool of fifty; again the choice of number of states was done via cross validation. The remaining thirty sequences were used to test the ability of the model to encode the parameterization. The average error was computed to be about 0.37 inches (combined in x and y, an angular error of approximately 0.5°). The high level of accuracy can be explained by the increase in the weights W_j in those states that are most sensitive to variation in $\boldsymbol{\theta}$. When the number of training examples was cut to 5 randomly selected sequences, the error increased to 0.82 inches (about 1.1°), demonstrating how the PHMM can exploit interpolation to reduce the amount of training data necessary. The approach discussed in section 2.2.3 of tiling the parameter space with multiple unrelated HMMs would require many more training examples to match the performance of the PHMM on the same task.

Robustness to noise

Because of the impact of $\boldsymbol{\theta}$ on all the states of the PHMM, the entire sequence contributes evidence as to the value of $\boldsymbol{\theta}$. For classes of movement in which there is systematic variation throughout much the extent of the sequence, i.e. the magnitude of W_j is non-trivial for many j, PHMMs should estimate $\boldsymbol{\theta}$ more robustly than techniques that rely on querying a single point in time.

To show this ability, we added various amounts of Gaussian noise to both the training and test sets, and then estimated $\boldsymbol{\theta}$ using the direct measurement procedure outlined above and again with the PHMM testing EM procedure. The PHMM was retrained for each noise condition. For both cases the average error in parameter estimation was computed by comparing the estimated value with the value as measured directly with no noise present. The average error, shown in Figure 2-9, indicates that the parametric HMM is more robust to noise than the *ad hoc* technique. We note that while this particular *ad hoc* technique is obviously brittle and does not attempt

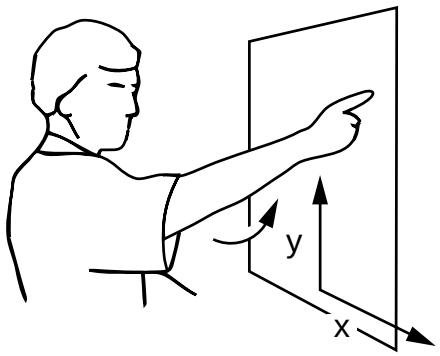

Figure 2-8: The point gesture used in section 2.4.3. The movement is parameterized by the coordinates of the target $\boldsymbol{\theta} = (x, y)$ within a plane in front of the user. The gesture consists of a preparation phase, a stroke phase (shown here) and a retraction.

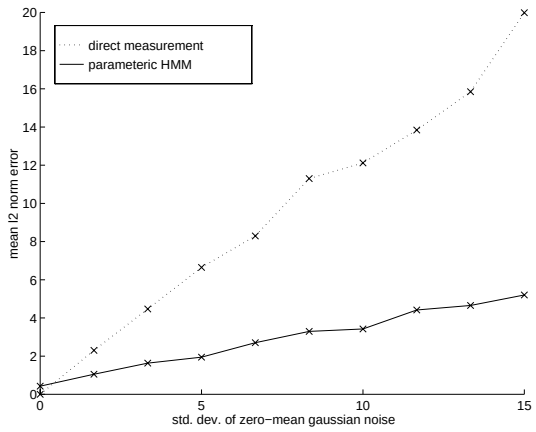

Figure 2-9: Average error over the entire pointing test set as a function of noise. The value of $\boldsymbol{\theta}$ was estimated by a direct measurement and by a parametric HMM retrained for each noise condition. The average error was computed by comparing the estimate of $\boldsymbol{\theta}$ to the value recovered by direct measurement in the noise-free case.

to filter potential noise, it is analogous to techniques used by previous researchers (for example, [40]) for real-world applications.

Bounding $\boldsymbol{\theta}$

Using the pointing data we demonstrate how the bounds on the prior uniform density on $\boldsymbol{\theta}$ can enhance recognition capabilities. To test the model, a one minute sequence was collected that contained a variety of movements including six pointing gestures distributed throughout. Using the same trained PHMM described above, we applied it to a 30 sample (one second) sliding window on the sequence; this is analogous to performing backward-looking causal recognition (no pre-segmentation) for a fixed gesture duration. Figure 2-10a shows the log likelihood as a function of time; the circled points indicate the peaks associated with true pointing gestures. The value of both the recovered and true $\boldsymbol{\theta}$ are indicated for these peaks, and reflect the small

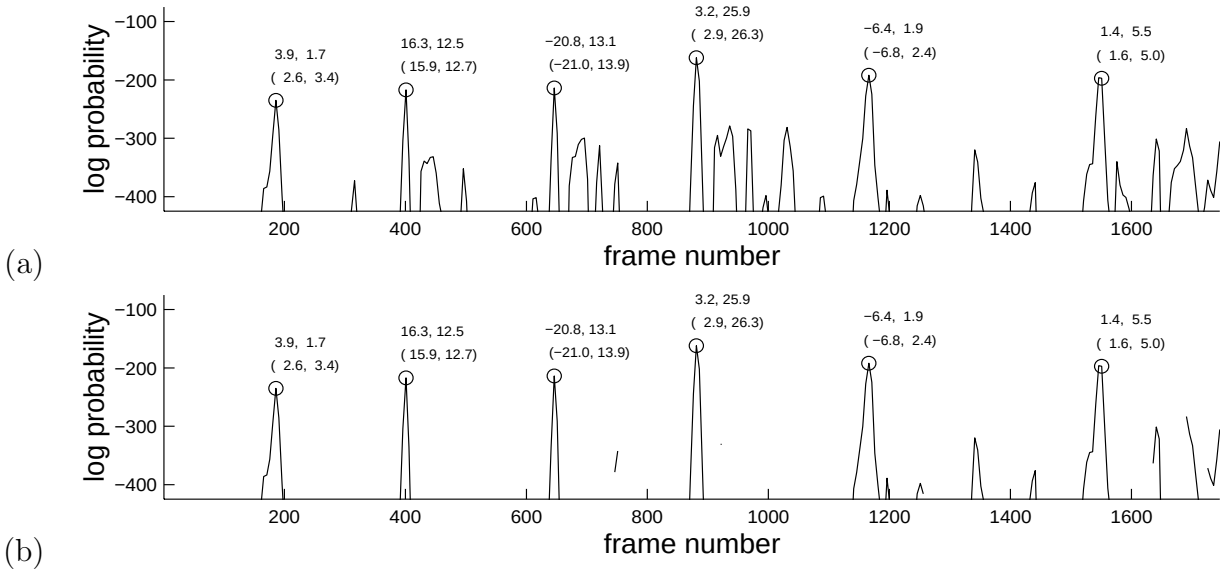

Figure 2-10: Recognition results are shown by the log probability of the windowed sequence beginning at each frame number. The true positive sequences are labeled by the value of $\boldsymbol{\theta}$ recovered by the EM testing algorithm and the ground truth value computed by direct measurement in parentheses. (a) Maximum likelihood estimate. (b) Maximum a posterior estimate, for which a uniform prior probability on $\boldsymbol{\theta}$ was determined by the bounds of the training set. The MAP estimate was computed by simply disallowing sequences for which the EM estimate of $\boldsymbol{\theta}$ is outside the uniform density bounds. This post-processing step is equivalent to establishing a prior on $\boldsymbol{\theta}$ in the framework presented in the appendix.

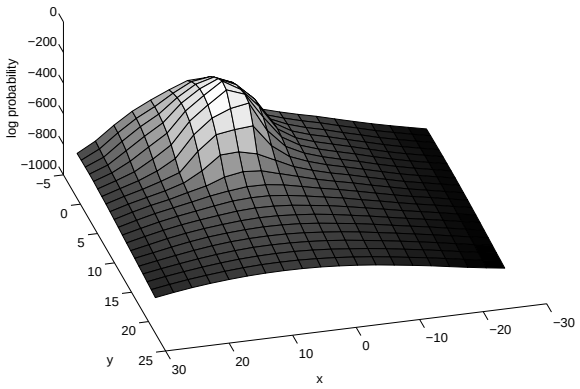

Figure 2-11: Log probability as a function of $\boldsymbol{\theta} = (x, y)$ for a pointing test sequence. The smoothness of the surface makes it possible to use iterative optimization techniques such as EM to find the maximum.

errors discussed in the previous section. Note that although it would be possible to set a log probability threshold to detect these gestures (e.g. -250), there are many false peaks that would approach this value.

However, if we look at the values of $\boldsymbol{\theta}$ estimated for each position of the sliding window, we can eliminate many of the false peaks. Recall that we assume $\boldsymbol{\theta}$ has a uniform prior distribution over some allowed range. We can estimate that range form the training data either by simply taking the extremes of the training set, or by estimating the density using a ML or MAP estimate [13]. Given such bounds we can post-process the results of applying the PHMM by eliminating those windows which select an illegal value of $\boldsymbol{\theta}$. Figure 2-10b shows the result of such filtering using the extremes of the training data as bounds. The improved output would increase the robustness of any recognition system employing these likelihoods.

Local vs. global maxima

One concern in the use of EM for optimization is that while each EM iteration will increase the probability of the observations, there is no guarantee that EM will find the global maximum of the probability surface. To show that this is not a problem in practice for the point gesture testing, we computed the log probability of a testing sequence for all legal values of $\boldsymbol{\theta}$. This log probability surface, shown in Figure 2-11, is unimodal, such that for any reasonable initial value of $\boldsymbol{\theta}$ the testing EM will converge on the maximum corresponding to the correct value of $\boldsymbol{\theta}$. The probability surfaces of the other test sequences in our experiments are similarly unimodal.[4]

The probability surface shown in Figure 2-11 is in fact due to a mixture of NT Gaussians, N the number of HMM states and T the length of the sequence. In order

[4]Given the graphical model equivalent in Figure 2-2 it is possible to exactly solve the for the best value of $\boldsymbol{\theta}$ using the junction tree algorithm [34] and conditional gaussian potentials [18], which model distributions of the form of equations 2.1 and 2.2. The relationship of the PHMM with Bayesian networks is explored in depth in Chapter 5.

to guarantee that EM does not fall in to a local minima, we must show that this superposition of many Gaussians results in a surface with only one local maxima. In fact, this will always be the case if the input sequence is smooth over time, the transformation W at each state changes smoothly from state to state as we move along the Markov model, and the mapping from input sequence to $\boldsymbol{\theta}$ is not inherently ambiguous. Intuitively, this is because any value of $\boldsymbol{\theta}$ which fits at a particular state j will also likely fit for state $j+1$, since the transformation at state $j+1$ is similar to that of state j, and the input also changes smoothly over time. Thus the NT modes in the total probability surface will lie near each other. Note that the change of the transformation W from state to state along the Markov model may be made arbitrarily small by using sufficiently many states. Also, this smoothness assumption will not be violated by a mistaken alignment by the HMM due to an incorrect initial value of $\boldsymbol{\theta}$ since the left-to-right topology of the HMM prevents pathological alignments. In practice, if too few states are used to represent the gesture, EM may not find the global maximum.

Finally, it is the case that for certain gesture families with symmetries, the probability surface over $\boldsymbol{\theta}$ will not be unimodal. For example, consider the family of sine waves, with $\boldsymbol{\theta}$ the phase of the sine wave. In this case the probability surface may have more than one mode, corresponding to the fact that the sine function is not invertible.

2.5 Nonlinear PHMMs

2.5.1 Nonlinear dependencies

The linear PHMM model is applicable only when the output distributions of each state of the HMM are linearly dependent upon $\boldsymbol{\theta}$. When the gesture parameter of interest is a measure of Euclidean distance and the feature space consists of coordinates in Euclidean space, the linear model of 2.3.2 is appropriate.

When this relation does not hold, there are at least three courses of action: (1) find an analytical function which when applied to the feature space makes the dependence of the output distributions linear in $\boldsymbol{\theta}$, (2) find some intermediate parameterization that is linear in the feature space and then use some other technique to map to the final parameterization, and (3) use a more general modeling technique, such as neural or radial basis function networks, to model the parametric variation of the state output densities with respect to $\boldsymbol{\theta}$.

The first option can be illustrated using the pointing example. Suppose the preferred parameterization of direction is a spherical coordinate system. Clearly, one could transform the Cartesian Polhemus data into spherical coordinates yielding a linear, in fact trivial, mapping. The only difficulty with this approach is that such an analytic transformation between feature space and parameter space must exist. When the parameterization is derived, say, from a user's subjective rating, it may be difficult or impossible to represent the feature's dependence on $\boldsymbol{\theta}$ analytically, especially without some insight as to how the user subjectively rates the motion.

The second option involves finding an intermediate parameterization that is linear in the feature space. For example, a musical conductor might convey a dynamic by sweeping out a distance with his or her arm. It may be adequate to model the motion using a parametric HMM with the distance as the parameter, and then use some additional technique to capture the nonlinearity in the mapping from this distance to the intended dynamic. This technique requires a fine knowledge of how the actual physical movement conveys the quantity of interest.

The last option, employing more general modeling techniques, is naturally suited to situations in which the parameterization is nonlinear and no analytical form of the parameterization is known. With a more complex model of the dependence on $\boldsymbol{\theta}$ (for example, a neural network), it may not be possible to solve for $\boldsymbol{\theta}$ analytically to obtain an update rule for the training or testing EM algorithms. In such a case we may perform gradient descent to maximize Q in the maximization step of the EM algorithm (which would then be called a "generalized expectation-maximization" (GEM) algorithm). In the next section we extend the PHMM framework to use neural networks and GEM algorithms to model non-linear dependencies.

2.5.2 Non-linear model

Nonlinear PHMMs replace the linear model of section 2.3.2 with a logistic neural network with one hidden layer. There is one neural network for each state whose function is to map the value of $\boldsymbol{\theta}$ to the output density parameters of that state. As with linear PHMMs, the output of each state is assumed to be Gaussian, with the variation of the density encoded in the mean $\hat{\boldsymbol{\mu}}_j$:

$$P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t, \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}), \Sigma_j) \tag{2.11}$$

The mean $\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})$ is defined to be the output of the network associated with state j:

$$\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}) = W^{(2,j)} g(W^{(1,j)})\boldsymbol{\theta} + b^{(1,j)}) + b^{(2,j)} \tag{2.12}$$

where $W^{(1,j)}$ denotes the matrix of weights from the input layer to the layer of hidden logistic units, $b^{(1,j)}$ the biases at each input unit, and $g(\cdot)$ the vector-valued function that computes the logistic function of each component of its argument. Similarly, $W^{(2,j)}$ and $b^{(2,j)}$ denote the weights and biases for the output layer (see [3]). Figure 2-12 illustrates the network architecture and the associated parameters.

2.5.3 Training

As with linear PHMMs, the parameters of the nonlinear PHMM are updated in the maximization step of the training EM algorithm by choosing ϕ' to maximize the auxiliary function $Q(\phi' \mid \phi)$. In the nonlinear PHMM, the parameters ϕ include the parameters of each neural network $W^{(l,j)}$, $b^{(l,j)}$ as well as Σ_j and transition probabilities a_{ij}.

The expectation step of the nonlinear PHMM is the same as that of the linear

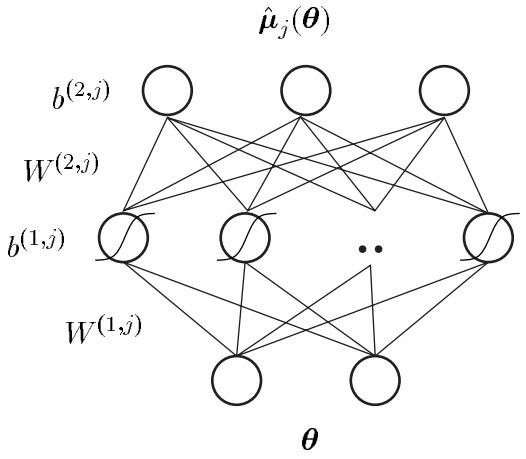

Figure 2-12: Neural net architecture of the non-linear PHMM used to map the values of $\boldsymbol{\theta}$ to $\hat{\boldsymbol{\mu}}$. There is a separate network for each state j for which the weights $W^{(i,j)}$, $i = 1, 2$ and the biases $b^{(i,j)}$, $i = 1, 2$ must be learned in training.

PHMM. In the EM maximization step we maximize Q. From the appendix, we have

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')} \tag{2.13}$$

where we select ϕ' to include the weights and biases of the j neural network. For the Gaussian noise model, we have

$$\frac{\partial Q}{\partial \phi'} = \sum_t \gamma_{tj} \Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j) \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k)}{\partial \phi'} \tag{2.14}$$

There is no way to solve for multi-layer neural network parameters directly (see [4] for a discussion of the *credit assignment problem*); thus we cannot set $\frac{\partial Q}{\partial \phi'}$ to zero and solve for ϕ' analytically. We instead apply gradient ascent to maximize Q. When the maximization step of EM algorithm relies on a numerical optimization the algorithm is referred to as the "generalized expectation-maximization" (GEM) algorithm [22, 46, 52].

Gradient descent applied to a multi-layer neural network may be implemented by the back-propagation algorithm [3]. In such a network we usually have a set of inputs $\{x_i\}$ and outputs $\{y_i\}$. We denote y^* as the output of the network, w_{ij}^l the weight from the ith node at the $l-1$ layer to the jth node at the lth layer, $z_i^l = \sum_j w_{ij}^l x_j^{l-1}$ is the activation, $x_i^l = g(z_i^l)$ is the output. The goal in the application of neural networks for regression is to minimize the total squared error $J = \sum_i (y_i^* - y_i)^2$ by tuning the network parameters w through gradient descent. The derivative of the error with respect to w_{ij} is

$$\begin{aligned} \frac{\partial J}{\partial w_{ij}^l} &= \frac{\partial J}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l} \\ &= \delta_i^l x_j^{l-1} \end{aligned} \tag{2.15}$$

where

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = g'(z_i^l) \sum_j \delta_j^{l+1} w_{ij}^{l+1} \tag{2.16}$$

$$\delta_i^L = y_i^* - y_i \tag{2.17}$$

Back-propagation seeks to minimize J by "back-propagating" the difference $y_i^* - y_i$ from the last layer L through the network. Network weights may be adjusted using, for example, a fixed step-size ρ:

$$\Delta w_{ij}^l = \rho \delta_i^l x_j^{l-1} \tag{2.18}$$

In the case of the nonlinear PHMM we can similarly minimize the expected value of the "error" term $(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j)^T \Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j)$ using back-propagation, thereby maximizing

the likelihood $P(\mathbf{x}_t \mid q_t = j)$:

$$J = \sum_{k,t,j} \gamma_{ktj} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j)^T \Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j) \tag{2.19}$$

From equation 2.16 we may thus derive a new "δ rule":

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = g'(z_i^l) \sum_j \delta_j^{l+1} w_{ij}^{l+1} \tag{2.20}$$

$$\delta_i^L = \gamma_{tj} \Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j) \tag{2.21}$$

In each maximization step of the GEM algorithm, it is not necessary to maximize Q completely. As long as Q is increased for every maximization step, the GEM algorithm is guaranteed to converge to a local maximum in the same manner as EM. In our testing we run the back-propagation algorithm a fixed number of iterations for each GEM iteration.

2.5.4 Testing

In testing we desire the value of $\boldsymbol{\theta}$ which maximizes the probability of the observation sequence. Again an EM algorithm to compute $\boldsymbol{\theta}$ is appropriate.

As in the training phase, we can not maximize Q analytically, and so a GEM algorithm is necessary. To optimize Q, we use a gradient ascent algorithm:

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} = \sum_t \sum_j \gamma_{tj} (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{2.22}$$

$$\frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = W^{(2,j)} \Lambda(g'(W^{(1,j)} + b^{(1,j)})) W^{(1,j)} \tag{2.23}$$

where $\Lambda(\cdot)$ forms the diagonal matrix from the components of its argument, and $g'(\cdot)$ denotes the derivative of the vector-valued function that computes the logistic function of each component of its argument.

In the results presented here, we use a gradient ascent algorithm with adaptive step size [60]. In addition it was found necessary to constrain the gradient ascent step to prevent the algorithm from wandering outside the bounds of the training data, where the output of the neural networks is essentially undefined. This constraint is implemented by simply limiting any component of the step that takes the value of $\boldsymbol{\theta}$ outside the bounds of the training data, established by the minimum and maximum $\boldsymbol{\theta}$ training values.

As with the EM training algorithm of the linear parametric case, for all of our experiments less than ten GEM iterations are required.

2.5.5 Easing the choice of parameterization

In section 2.4.3 we present an example of a pointing gesture parameterized by projection of hand position onto the plane parallel and in front of the user at the moment

that the arm is fully extended. The linear PHMM works well since the projection is a linear operation over the range of angles used in the experiment.

The nonlinear variant of the PHMM just introduced is appropriate in situations in which the dependence of the state output distributions on the parameter $\boldsymbol{\theta}$ is not linear, and cannot be made linear easily with a known coordinate transformation of the feature space.

In practice, a useful consequence of nonlinear modeling for PHMMs is that the parameter space may be chosen more freely in relation to the observation feature space. For example, in a hand gesture recognition system, the natural feature space may be the spatial position of the hand, while a natural parameterization for a pointing gesture is the spherical coordinates of the pointing direction (see Figure 2-13).

The mapping from parameter to observations must be smooth enough to be learned by neural networks with a reasonable number of hidden units. While in theory a 3-layer logistic neural network with sufficiently many hidden units and sufficient data is capable of computing any smooth mapping, we would like to use as few hidden units as possible and so choose our parameterization and observation feature space to give simple, learnable maps. Cross validation is probably the only practical automatic procedure to evaluate parameter/observation feature space pairings, as well as the number of hidden units in each neural network. The computational complexity of such approaches is a drawback of the nonlinear PHMM approach.

In summary, with nonlinear PHMMs we are free to choose intuitive parameterizations but we must be careful that it is possible to learn the mapping from parameters to observation features given a particular observation feature space.

2.6 Results of non-linear model

To test the performance of the nonlinear PHMM, we conducted an experiment similar to the pointing experiment of section 2.4.3 but with a spherical coordinate parameterization rather than the projection onto a plane in front of the user.

We used a Polhemus motion capture system to record the position of the user's wrist at a frame rate of 30Hz. Fifty such examples were collected, each averaging 29 time samples (about 1 second) in length. Thirty of the sequences were randomly selected as the training set; the remaining 20 comprised the test set.

Before training, the value of the parameter $\boldsymbol{\theta}$ must be set for each training example, as well as for each testing example to evaluate the ability of the PHMM to recover the parameterization. We directly measured the value of $\boldsymbol{\theta}$ by finding the point at which the depth of the wrist away from the user was greatest. This point was transformed to spherical coordinates (azimuth and elevation) via the arctangent function. Figure 2-13 diagrams the coordinate system.

Note that for pointing gestures that are confined to a small area in front of the user (as in the experiment presented in section 2.4.3) the linear parametric HMM approach will work well enough, since for small values the tangent function is approximately linear. The pointing gestures used in the present experiment were more broad, ranging from -36° to +81° elevation and -77° to +80° azimuth.

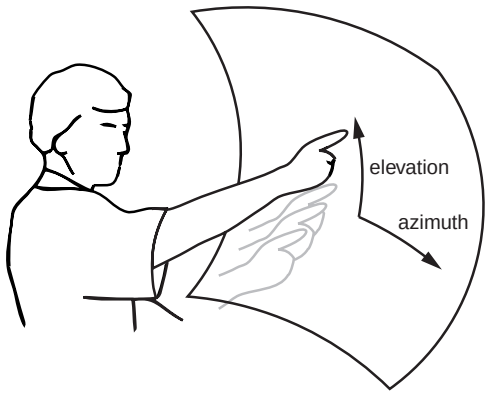

Figure 2-13: The spherical coordinate system is a natural parameterization of pointing direction.

An 8 state causal nonlinear PHMM was trained on the 30 training examples. To simplify training we constrained the number of hidden units of each state to be equal; note that this is not required by the model but makes choosing the number of hidden units via cross validation easier. We evaluated performance on the testing set for various numbers of hidden units and found that 10 hidden units gave the best testing performance.

The average error over the testing set was computed to be about 6.0° elevation and 7.5° azimuth. Inspection of the surfaces learned by the logistic networks of the nonlinear PHMM reveals that as in the linear case, the input's dependence on $\boldsymbol{\theta}$ is most dramatic in the middle of the sequence, the apex of the pointing gestures. The surface learned by the logistic network at the state corresponding to the apex captures the nonlinearity of the dependency (see Figure 2-14). For comparison, an eight state *linear* PHMM was trained on the same data and yielded an average error over the same test set of about 14.9° elevation and 18.3° azimuth.

Lastly, we demonstrate detection performance of the nonlinear PHMM on our pointing data. A one minute sequence was collected that contained a variety of movements including six pointing movements distributed throughout. To simultaneously detect the gesture and recover $\boldsymbol{\theta}$, we used a 30 sample (one second) sliding window on the sequence. Figure 2-10 shows the log probability as a function of time and the value of $\boldsymbol{\theta}$ recovered for a number of recovered pointing gestures. All of the pointing gestures were correctly detected and the value of $\boldsymbol{\theta}$ accurately recovered.

2.7 Discussion

A new method for the representation and recognition of parameterized gesture is presented. The idea is to parameterize the underlying output probabilities of the states of an HMM. Because the parameterization is explicit and analytic, the dependence on the parameter $\boldsymbol{\theta}$ can be learned within the standard EM formulation.

The method is interesting from two perspectives. First, as a gesture or activity

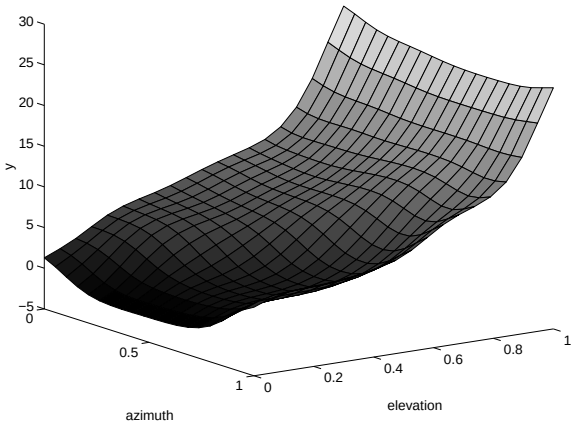

Figure 2-14: The output of the logistic network corresponding to state $j = 5$ displayed as a surface. State 5 is near the apex of the gesture and shows the greatest sensitivity to pointing angle. Only the y coordinate of the output is shown; the x coordinate is similarly nonlinear.

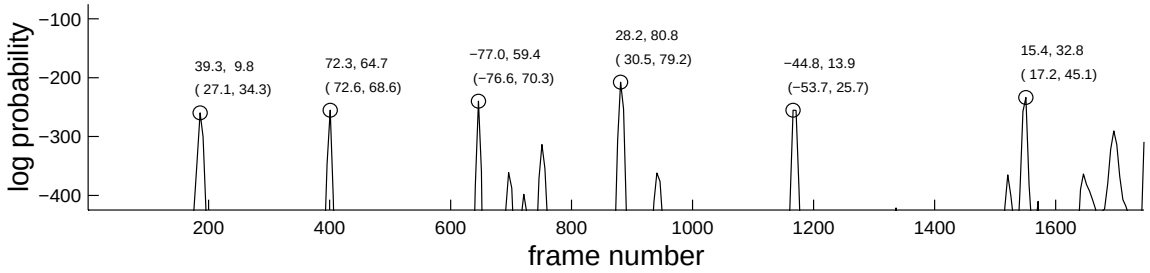

Figure 2-15: Recognition results are shown by the log probability of the windowed sequence beginning at each frame number. The true positive sequences are labeled by the value of $\boldsymbol{\theta}$ recovered by the EM testing algorithm and the value computed by direct measurement (in parentheses).

recognition technique, it is immediately applicable to scenarios where inputs to be recognized vary smoothly with some meaningful parameter(s). One possible application is advanced human-computer interfaces where the gestures indicating quantity must be recognized and the quantities measured. Also, the technique may be applied to other types of movement, such as human gait, where one would like to ignore or extract some component of the style of the movement.

Second, the parameterized technique presented is domain-independent and is applicable to any sequence parsing problem where some context or style ([68]) spans an entire sequence.

The PHMM framework has been generalized to handle nonlinear dependencies of the state output distributions on the parameterization $\boldsymbol{\theta}$. We have shown that where the linear PHMM employs the EM algorithm in training and testing, the nonlinear variant similarly uses the GEM algorithm.

The drawbacks of the of the generalized approach are two-fold: the number of hidden units for the networks must be chosen appropriately during training, and secondly, during testing the GEM algorithm is more computationally intensive than the EM algorithm of the linear approach.

The nonlinear PHMM is able to model a much larger class of parameterized gestures and movements than the linear parametric HMM. A benefit of the increased modeling ability is that with some care the parameter space may be chosen independently of the observation feature space. It follows that the parameterization may be tailored to a specific gesture. Furthermore, more intuitive parameterizations may be used. For example, a family of movements may be parameterized by a subjective quantity (for example, the style of a gait). We believe these are significant advantages in modeling parameterized gesture and movement.

2.8 Unlabelled Training Examples

The PHMM training formulation relies on training examples which are each labelled by a value of $\boldsymbol{\theta}$. Can the PHMM also use unlabelled training examples? If the examples are segmented in time, it is likely that the PHMM testing procedure, in which both $\boldsymbol{\theta}$ and W are estimated simultaneously, could be used in place of the usual PHMM training procedure to use unlabelled training examples. The idea is to use EM to arrive at the value of $\boldsymbol{\theta}$ that each example would have been labelled by while simultaneously estimating W.

Brand [10] has developed a framework similar to the PHMM which exploits a multi-term objective function, including constraints on similarity throughout the gesture family, and parsimony with respect to parameters afforded the system in a fully unsupervised approach.

The success of the unsupervised approach of the PHMM relies on the segmentation of the examples and the constrained nature of the left to right topology that is used in the preceding development of the PHMM. Combined, these two constraints will insure that the examples will be approximately aligned in time throughout the training process.

Furthermore, the constraint that the transformation be linear in the case of linear PHMMs will also ensure a correct solution if the transformation is indeed linear. In the case of the nonlinear PHMM, the situation is complicated by the fact that the neural network may approximate a highly complicated (and possibly unlikely) manifold if many hidden units are explored. The unsupervised nonlinear PHMM will likely be very sensitive to the number of hidden units afforded each neural network.

Note that if all the training examples are unlabelled the actual values of W and $\boldsymbol{\theta}$ will not be unique. If we allow a small fraction of the examples to be labelled, the solution will be unique, and the values of $\boldsymbol{\theta}$ recovered will be interpretable in relation to the labelled examples.

2.9 Conclusion

The PHMM represents the manifold of a gesture family. In the distinction introduced in the previous chapter, the PHMM represents this manifold implicitly. That is, rather than consider a sequence as a vector of concatenated input observations, the PHMM models the manifold piece-wise, with a manifold modeling the variation of each part of the sequence. These models are embedded in a hidden Markov model framework.

In the next two chapters we introduce another framework for handling systematic variation in gesture: temporal models. These techniques are related to PHMMs in that they also encode variation in the output probability distributions of hidden Markov models.

Chapter 3

Learning Gestures from Temporal Structure

3.1 Introduction

As discussed in Chapter 1, in the typical gesture recognition paradigm a gesture model is fit to a set of example sequences in an offline fashion. During runtime, these models are then matched against an input sequence to check which, if any, model best describes the data. In this paradigm learning is conducted entirely offline. The drawback of the offline learning approach is that the testing scenario must closely resemble that of training environment, many training examples must be provided, many gesture models must be computed, or a combination of all three.

In this chapter we begin to explore the idea of online learning of gesture. The idea is to leave some parts of the gesture model unspecified until runtime, at which point the gesture model is adapted to the situation at hand. By leaving some things unspecified until runtime, we have a gesture model that is applicable to particular scenarios that were not manifest in the training set, and so the gesture recognition proceeds more robustly.

The key to constructing gesture models that are adapted in an online fashion to handle novel situations is to encode sufficient constraints in the adaptation process and gesture model such that the adaptation process preserves the semantics of the gesture, such that for example, the "hello" gesture is not adapted to look like "goodbye".

In this chapter we explore the online learning of gesture with constraints related to the temporal structure of the gesture. There is reason to believe that, in many applications, the temporal pattern or signature of a gesture is invariant across many runtime scenarios. For example, even though lighting conditions, clothing, and so on, may change from session to session for a computer vision system, the temporal pattern of moving from some configuration A to B to C and then back again to C remains the same. "Learning" then consists of determining appearance models associated with A, B and C, such that the input sequence makes sense under the given temporal relations.

Consider the following performance animation system, which was shown by the author as part of an exhibit at the SIGGRAPH '96 Digital Bayou exhibit area. The idea was to let the user "fly" a seagull over a virtual landscape. The user controlled the bird's wing motion (and thus the rate of forward flight) by some motion determined in a pre-training phase. This training phase consisted of showing the seagull the user's own "wings down" and "wings up" configuration. This was enough data for the system to build a simple radial basis function (RBF) approximation of the mapping from 3-dimensional hand position as derived from the STIVE vision system, to a flapping phase variable passed to the flight graphics system. Figure 3-1 shows the system in use. A video of the system is located at `http://www.media.mit.edu/~drew/movies`.

While this simple performance animation system is interesting from the standpoint that it was fully customized to each user, the question remained: how to avoid the training phase? Given that "flapping" is such a simple motion it seems reasonable that the system, after a short time watching the user, should be able pick up "wings down" and "wings up". Note that here it would be insufficient to look for just any periodic motion, since there is a distinct semantic difference between "down" and "up" that must be preserved.

At least two sources of information might be used to drive the learning in an updated version of the seagull that learns automatically: first, the temporal structure of the motion, and second, the current state of the virtual character, on the presumption that the user will probably follow the seagull for a bit at first. This last bit of contextual information is enough to remove the ambiguity between "up" and "down".

In section 3.2 we describe a system designed to classify two broad classes of natural gesture that differ primarily in their temporal patterns. While the system runs in an offline fashion, it demonstrates that an online approach is valid given strong temporal models. The next chapter describes a similar framework that learns appearance models for a truly online, realtime computer vision application.

3.2 Natural Gesture from Video

3.2.1 Introduction

One difference between listening to someone relating a story over the phone and listening to the same story in person is the presence of *gesture.* To a listener some gestures seem inconsequential while others clearly have an expository and clarifying purpose. Consider the problem of designing a video coding system that codes the semantically meaningful gestures clearly, perhaps at the expense of other segments of the conversation. Such a system requires the ability to detect when meaningful gestures occur. In the rest of the chapter, we present the development of a theory and algorithm for the visual detection of semantically important natural gestures.

The traditional paradigm for hand gesture recognition involves the construction of a model for each gesture to be recognized. This usually proceeds by collecting a number of examples of the gesture, computing the "mean gesture" and quantifying

(a)

(b)

Figure 3-1: The seagull performance animation system, in which the user controls the virtual seagull. (a) a view of the user interacting with the system. (b) the graphics display. Related work appears in [23].

the variance seen in the examples. The hope is that this description will generalize to the actual test data. Examples of this approach include [64, 8, 74, 19, 66].

This typical pattern recognition approach may be well suited to the recognition of stylized or literal gesture, such as the gestures made by a user navigating aeronautical data in a virtual reality system by contorting their hands. These actions are less gestures than particular literal movements. Others examples are the emblematic gestures substituting for simple linguistic constructs: the ubiquitous OK sign or "giving someone the finger." These situations lend themselves to the construction of sophisticated models capable of representing the variations between people; in the case of the VR-controller, one might even alter the gesture vocabulary to make the recognition more robust.

However, as an approach to *natural gesture* understanding, this methodology seems inappropriate. By "natural gesture" we mean the types of gestures spontaneously generated by a person telling a story, speaking in public, or holding a conversation. The reasons for this skepticism are clear. First, the particular configurations and motions observed in natural gesture are inherently speaker dependent, influenced by cultural, educational, and situational factors [41]. An approach employing fixed, physical descriptions of gesture might find no cross-speaker invariances.

Second, and more important, is that the literal representation of the gesture assumes that the spatial configuration is the most significant aspect of the signal to be extracted. Given that we are observing a sequence, it is plausible that more abstract *temporal* properties are the important elements of a gesture.

In this chapter we develop a method for the detection of the important temporal structure — the *gestural phases* — in natural gesture. We begin by briefly relating some recent developments in the theory of natural gesture which have identified several key *temporal* aspects of gesture important to communication. In particular, gesticulation during conversation can be coarsely characterized as periods of bi-phasic or tri-phasic gesture separated by a rest state. We next present an automatic procedure for hypothesizing plausible *rest state* configurations of a speaker; the method uses the repetition of subsequences to indicate potential rest states. Following, we develop a state-based parsing algorithm used to both select among candidate rest states and to parse an incoming video stream into bi-phasic and tri-phasic gestures. Finally we demonstrate how such characterizations can be used to increase the comprehensibility of a low frame rate coding of video of story-telling speakers[1].

3.2.2 Gesture in Communication

Recent research in the field of natural gesture generation and parsing has identified four basic types of gesture generated during discourse [47, 17]. Three of these are considered to have meaning in a dialog: *iconic*, where the motion or configuration of the hands physically match the object or situation of narration; *deictic*, a pointing gesture; *metaphoric*, where the motion or shape of the hands is somehow suggestive

[1]Portions of this chapter appear in [79, 80]

of the situation. The fourth gesture type, *beats*, is generated to show emphasis or to repair mis-spoken segments.

Characteristic of these gesture types are particular temporal signatures. For example, each is typically bracketed by the hands being in a "rest state." Beat gestures — the simplest — consist only of a small baton-like movement away from the rest state and then back again; these gestures may be termed "bi-phasic." The iconic, metaphoric, and deictic gestures are executed by first "transitioning" from the rest phase into gesture space (the space in front of the speaker), then executing a smaller movement (the "stroke"), remaining at that configuration for a short duration, and then transitioning back to the rest state. Thus, these gestures may be termed "tri-phasic." What it means for a movement of the hands to be a natural gesture is defined, at least in part, by these temporal characteristics. The bi-phasic and tri-phasic distinction is introduced in [16]. The distinction between beats and representational gestures (iconic and metaphoric) is also discussed in [70].

We employ the above descriptions to derive a parsing mechanism sensitive to the temporal structure of natural gesture. Our initial goal is to find possible instances of bi- and tri-phasic gestures in a video sequence of someone telling a story. The motivation is that the tri-phasic gestures encode meaning and need to be segmented from the input gesture stream if they are to be incorporated into any additional interpretation or coding processes.

3.2.3 Detecting candidate rest states

Gesture data

The data presented here are extracted from video of naive subjects relating a story. The subject was led into a closed room and asked to think of a time in which they believed they were in grave danger. The subject was then asked to tell a story of this event. The subject was instructed to look at the experimenter and not the camera, and was also warned that the experimenter would only provide minimal (nonverbal) feedback. Recording proceeded for as long as it took the subject to recount the story.

To reduce the size of recorded imagery, the video was digitized at low spatial (120×160 pixels), temporal (10Hz), and photometric (8-bit gray scale) resolutions. The two sequences used to illustrate the results are 3min38sec and 4min10sec long, for a total of 4700 frames or 90MB of data. A few frames of the first sequence are shown in Figure 3-2.

3.2.4 Feature extraction

To analyze and compare subsequences we require a compact representation for the imagery. Because the focus of our analysis is on the temporal characteristics of the sequences and not the spatial appearance, we select the rather aggressive approach of representing each frame by a small number of coefficients derived from an eigenvector decomposition of the images [71].

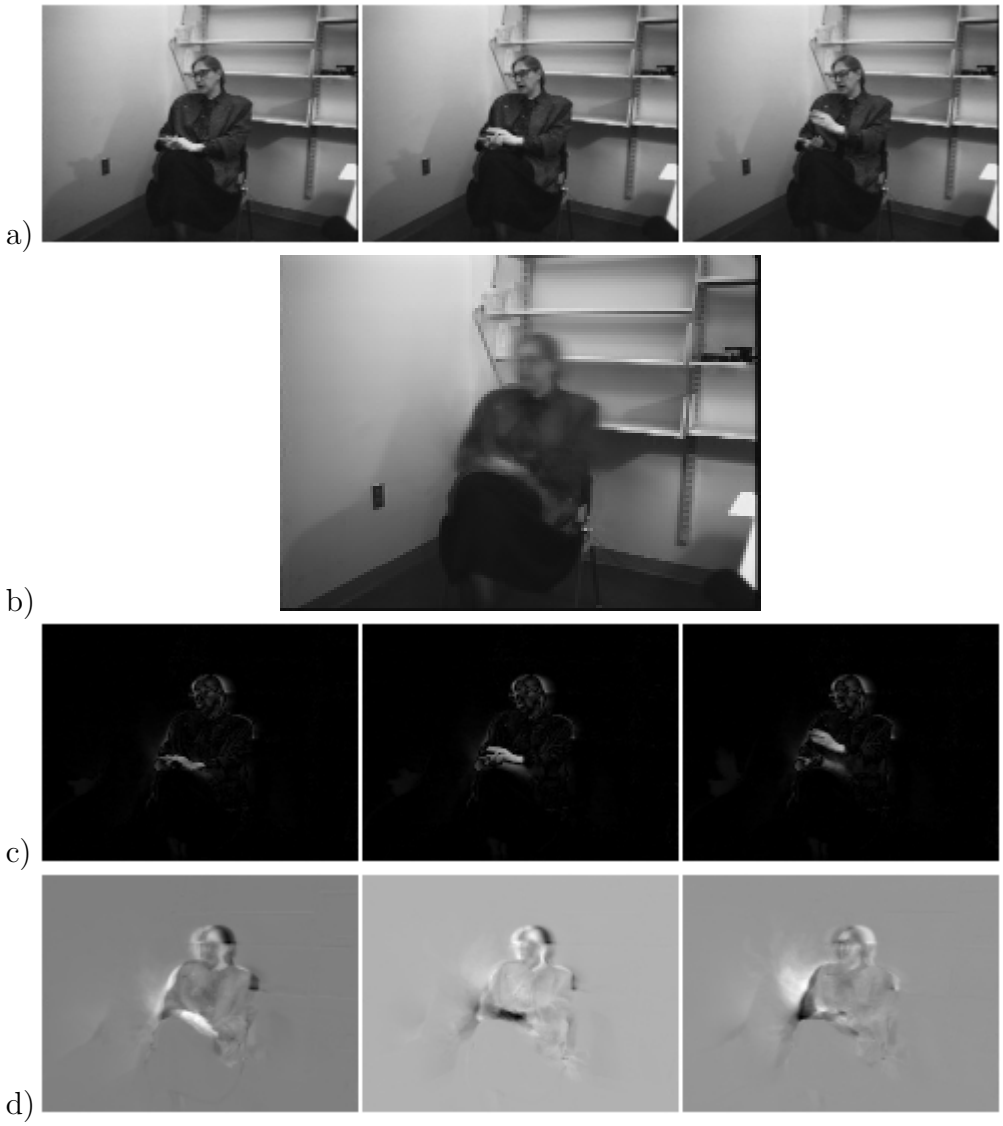

Figure 3-2: Three consecutive frames of the sequence used to illustrate the technique are shown in (a). (c) is the result of computing at each pixel the absolute value of the difference between the images in (a) and the mean image (b) computed from all frames of the sequence. (d) The first 3 eigenvectors of the image sequence.

We apply the technique to image sequences by randomly selecting a few hundred frames, computing the eigenvector decomposition of these frames, and then projecting all frames of the image sequence onto the resulting basis set. Next, the basis set vectors are ordered by how much variance each accounts for in the training frames. Because there is not tremendous variation in imagery of a person telling a story, and since it can be shown that two points that are nearby in the original image space are also nearby in the resulting low-dimensional space [49], we only need retain a small number of coefficients for this work. In the experiments reported here, we use only $n = 10$ coefficients to represent each frame; on average the 10 coefficients account for 55% of the variance. These coefficients are the entire representation used for all further processing.

Note that many of the usual objections to appearance-based principal components — e.g. sensitivity to translation or illumination change — are less relevant in this situation than in others such as face recognition[71]. In section 4 we use absolute distance in coefficient space only to compute a frame's similarity to possible rest state frames. By definition, rest states occupy large numbers of frames and so the set of rest states is well covered by the basis set. For parsing the gestural states, short term *movement* in coefficient space is measured; this signal is unlikely to be dramatically affected by a slightly improper basis that might occur with a change in illumination or translational shift of the speaker. Furthermore, the algorithms presented here are applicable to any feature-based representations of hand configuration.

Subsequence distance matrix

Let $\mathbf{x}_i$ be the n-vector of the eigenvector projection coefficients representing the ith frame of the image sequence. We define $d_{i,j}$ to be the difference between the coefficients of two frames $\mathbf{x}_i$ and $\mathbf{x}_j$ using a distance metric such as the Euclidean norm. Denoting the length L subsequence beginning at frame i and ending with frame $(i + L - 1)$ as $\mathbf{x}_i^L$, we can define the difference between two subsequences $\mathbf{x}_i^L$ and $\mathbf{x}_j^L$ as the total Euclidean distance:

$$d_{i,j}^L = \left[\sum_{k=0}^{L-1} d_{i+k,j+k}^2 \right]^{\frac{1}{2}}$$

By computing $d_{i,j}^L$ for all pairs of i, j we can construct a matrix for all the subsequences.

Figure 3-3 presents the subsequence distance matrix for a central part of one of the test sequences. The diagonal is black, indicating perfect correlation. Black regions off the diagonal indicate points in time where a particular length L subsequence is repeated. For example, beat gestures, which appear in the video as short repeated motions, show up as dark, short parallel lines. Subsequences that are fairly generic (e.g., hands are near the rest position) are likely to have several regions of high similarity. Conversely, motions or configurations that are highly unusual in the sense that they are unlike any other subsequences manifest themselves in the matrix as a bright row (and corresponding column) in which the mean distance is much greater

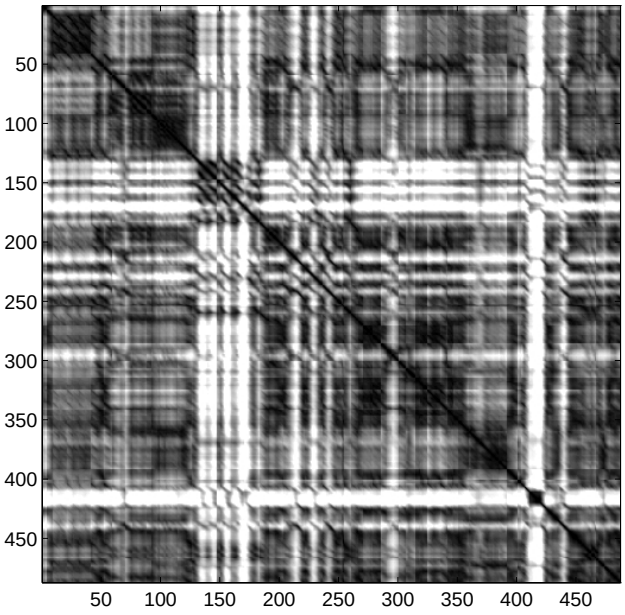

Figure 3-3: Distance matrix of subsequences of length 5, for a 300 frame section of the original video sequence. Dark parallel diagonal lines indicate repeated motions, possibly by "beat" gestures. An interesting sequence in which the subject repeatedly waves her arm at her side in a circular fashion begins at $i = 130$. The white bar around $i = 415$ indicates atypical movement; the subject is waving both arms high above her head.

than the overall mean.

The nature of the distance matrix is sensitive to the subsequence length L. If L is small, we may see spurious similarities. If L is too big, then the matching of "atomic" or primitive subsequences may be prevented. For the results reported here we have set $L = 5$ (one half second at 10Hz); we have not systematically experimented with varying L.

Selecting candidate rest states

Because rest states start and end each bi-phasic and tri-phasic gesture, and because rest states involve little or no motion for a reasonable duration of time, one expects a subsequence corresponding to a rest state to be repeated often. Furthermore, if one were to reconstruct a sequence without regard to the semantic importance of any particular subsequences and using only a small collection of primitive subsequences, then one would want to use the rest state subsequence(s) as (one of) the primitives since it would well describe the imagery.

Our approach to finding candidate rest states is to use the syntactic (as opposed to semantic) reconstruction idea and to select a small set of subsequences which when repeated and assembled in the right order would reconstruct the original sequence as best possible. Of course, finding the optimal set of k subsequences for reconstruction is an exponential problem since the best k does not necessarily contain the best $k-1$ set. However, we expect the reconstruction to be dominated by a few rest states plus

many unrelated motions. Therefore we use a "greedy" algorithm to select a set of reconstructing subsequences.

Let $\mathcal{M}$ be the set of all subsequences (call these models). Let $M \subseteq \mathcal{M}$ be a set of subsequences, where each $m \in M$ specifies the length L subsequence beginning at $\mathbf{x}_m$ (frame m in the original sequence). For each $\mathbf{x}_i$ define

$$y_i = \arg\min_{m \in M} d^L_{m,i}$$

That is, the sequence y_i is the best reconstruction of the sequence $\mathbf{x}_i$ given the models M. The approximation error at frame i is $e_i = \min_{m \in M} d^L_{m,i}$.

The "greedy" procedure is as follows: given the previously selected models M, pick the new subsequence model to add to M such that the decrease in $\sum_i e_i$ is maximized. The algorithm is initialized by choosing the best single subsequence, $M = \{i\}$ where $i = \arg\min_j \sum_k d^L_{j,k}$.

The algorithm can be iterated as many times as there are frames; at that point $\sum_i e_i = 0$. However, each additional decrease in approximation error becomes quite small after a small number of models are included. For the 2200 frame sequence of Figure 3-2 we select only the first 40 subsequences; an additional 60 subsequence would be required to reduce the error only by one half.

Figure 3-4 illustrates the top six ranked (length 5) subsequences. Notice the variety of candidate rest states. The last example (6) is one which will later be rejected by our parsing mechanism: although the subsequence can be used to reconstruct a significant part of the original video, it does not have the right temporal properties to be considered a rest state. Figure 3-5 illustrates the top four candidates from a second example sequence. In this example, notice the radically different rest states recovered.

3.2.5 Detecting gesture phases

Given candidate rest states, we can now simultaneously evaluate them and parse the gesture stream into bi-phasic and tri-phasic gestures. The approach is to use a Markov state description, but with the traditional use of transition probabilities replaced with an explicit model of duration.

Markov states with duration modeling

Although Hidden Markov Models have been a popular technique for the recognition of gesture (see [84, 64, 66, 74]) we note that in our system *the states are not hidden.* In particular, our analysis of natural gesture types in section 2 identifies rest (R), transition (T), and stroke (S) states. The properties of these states are known and can be characterized by similarity in appearance to a rest state, amount of motion exhibited, and the duration during which the state is maintained. Probabilistic densities for these descriptions can be derived directly from training data.

Furthermore, the temporal structure of gesture can be described *a priori* using these states. Beat gestures correspond to moving from R to T and back to R: <R-T-

Figure 3-4: The top six ranked (length 5) subsequences for reconstruction. This selection illustrates the variety of candidate rest states. The last candidate (6) will be rejected by the temporal parsing.

Figure 3-5: The top four ranked subsequences for reconstruction for a second subject.

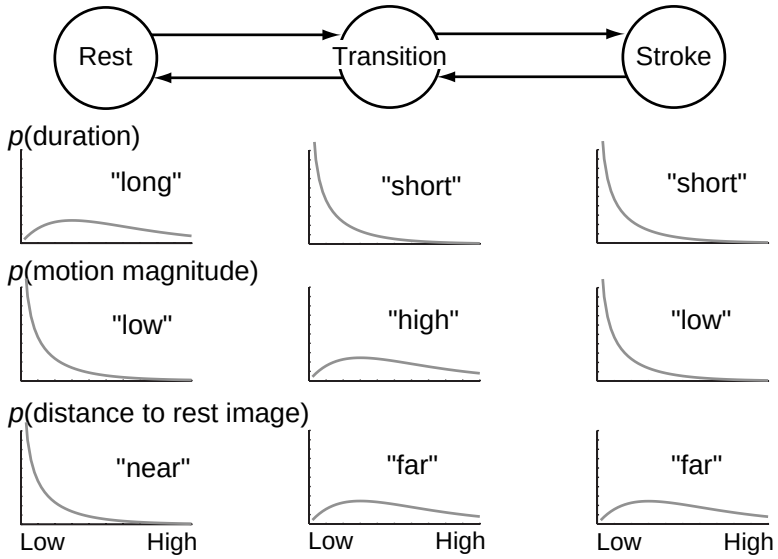

Figure 3-6: The three state machine describing the possible gestures. Below each state is a description of the gamma-density pdf for the given variables. The transitions are unlabeled because we do not use transition probabilities in generating a parse; rather, the duration models drive the temporal relationships.

R>; tri-phasic gestures traverse from R to T to S to T and back to R: <R-T-S-T-R>.[2] The *a priori* knowledge of the structure and properties of the states distinguishes our work from the typical HMM techniques.

Figure 3-6 graphically depicts a gesture phase finite state machine (FSM) and the associated properties of each state. While the exact form and values of the probability densities are not critical (each are modeled by gamma densities) it is important to understand their qualitative nature. The rest state R is modeled as tending to be "near" the rest state's position in eigen-space (using, say, the Euclidean norm), to have "low" motion as measured by the average traversal in eigen-space of the coefficients used to describe each frame, and of "long" duration. Likewise the T state is "far", "high", and "short" while the S state is "far", "low", and "short."

Given these descriptions, one might be tempted to just cluster and classify the image frames using appearance and velocity as features, and ignore any notion of transition. The difficulty with this is the idea of *duration*, which is well modeled using a Markov system ([61]) where a modified Viterbi algorithm exists for parsing input streams with respect to duration models. Duration is fundamental to the idea of being a rest, transition, or stroke phase. The property of duration is much more critical to the gesture-parsing than is the probability of a transition occurring between any two states.

In traditional Markov systems, loopback transitions and their associated probabilities are manipulated in an attempt to alter the duration that a traversal remains

[2]We can also define multi-phasic gestures to be tri-phasic gesture which cycles through the T-S-T sequence more than once: <R-T-[S-T]$^+$-R>; this is sometimes seen when tri-phasic gestures are tightly repeated or overlap.

in a given state. Formally, a fixed loopback transition probability is equivalent to an exponential density on duration, favoring shorter stays within a state. With such systems it is difficult if not impossible to disallow short durations.

To incorporate duration models and to use the Viterbi algorithm[61] to generate the best possible parse, we adopt the framework of a Markov system, but with *no cost for a transition.* The result is a FSM where only the state-output probabilities and the duration the system remains in each state affect the parse. The effect is that instead of using transition probabilities to drive the temporal structure, we use the duration model. Proposing a traversal from state i to state j at time t requires accepting the cost of ending the duration in the first state and starting that of the next.

Identifying rest states

The verification of rest states is accomplished by selecting a candidate subsequence, defining a gesture-phase-FSM using that candidate to define the rest state location in eigenspace, and then parsing the input data. If the tested subsequence is indeed a rest state, then the parsed input should spend a significant amount of time in the rest state R. If it is not, then most of the parse will oscillate between states T and S.

This verification process was applied to each of 40 candidate subsequences, ordered by the reconstruction method of section 3.2.3. Two points are of interest. First, many of the initial candidates (e.g. those ranked 6, 7, and 9) do not satisfy the rest state criteria when considered in a temporal context; their elimination validates the need for the temporal analysis beyond clustering.

Second, many candidate subsequences exhibit good rest state behavior, confirming the idea that there may be several rest states for a given speaker in a given situation. To select a set of rest states adequate to parse the gesture, we again construct a greedy algorithm; here we accumulate rest states according to how many new time-steps are now parsed as rest states if a new candidate is included. For the example of Figure 3-4 we use 20 rest states. Manual thresholding selected this number. However, for the method of detecting the gesture states detailed in the next section, overestimating the number of rest states is much less of a problem than underestimating.

Results: detecting bi-phasic and multi-phasic gestures

To detect gesture phases, we need to construct a gesture phase FSM with the necessary rest states, and then parse the input sequence. To incorporate multiple rest states, we redefine distance to the rest state feature as the minimum distance to all of the chosen subsequences. To then detect the relevant gestures we simply parse the incoming video stream with respect to the gesture-phase-FSM; the parsing algorithm is a duration-modified Viterbi optimization [61].

Figure 3.2.5 shows keyframes taken from four different triphasic gestures correctly labeled automatically by the system. Figure 3-8 illustrates the results for a 100 second long subsequence of one of the two video sequences tested; the other sections have similar results. The top two traces indicate the manually annotated labeling

Figure 3-7: Keyframes from four different triphasic gestures correctly labeled automatically by the system. Compare these images with those of Figure 3-4.

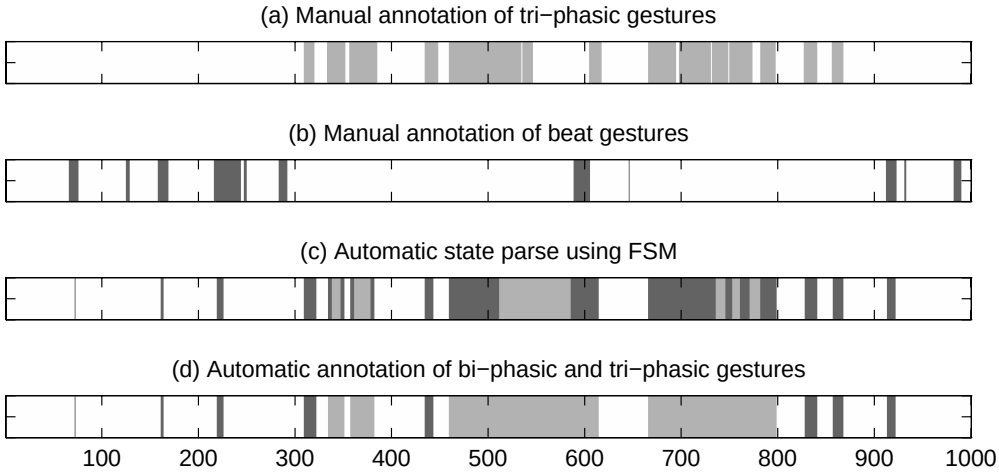

Figure 3-8: Example results of parsing the gesture video. (a) and (b) Visual encoding of a manual annotation of the presence of gesture. The annotation was produced by an expert in gesture communication who had not seen any of the results before viewing the video. (c) The state parse of our gesture-state-FSM and (d) the automatically derived labeling from the state parse (dark grey indicates bi-phasic beats, light grey tri-phasic gestures).

of tri-phasic and beat gestures. These labels were generated by the third author before seeing any of the results. The bottom trace depicts the state-based parse of the incoming video. Notice the overall similarity in the detection. The extent of agreement is difficult to measure quantitatively, and perhaps a bit premature as the gesture community still has difficulties agreeing as to what movements are gestures. Our contribution is the demonstration that the temporal structure coupled with an *a priori* state-based description is adequate to recover most of the gestures present.

We also note that we have tested this procedure on the 4min10sec sequence of a different speaker illustrated in Figure 3-5. Only one parameter of the model needed to be adjusted to generate similar results (to change the gamma pdf for motion magnitude on the rest state), and the parsing agrees with the authors' observations of the gesture. As mentioned, this sequence is interesting because of the radically different rest states recovered; a system must be sensitive to multiple rest states if it is to segment the gestures properly.

3.2.6 Semantically sensitive coding of gesture

Finally we can address the problem introduced at the outset: the development of a coding system that ensures meaningful gestures are clearly coded. We define the problem as follows: Given an original video sequence of someone telling a story, and given a required average frame rate, select the best possible set of frames to accompany the audio signal. The coding problem serves as both a test and an application of our method for detecting semantically meaningful gestures (i.e., tri-phasic gestures) in story-telling.

In evaluating the performance of our technique, there are two difficulties. The first is that by definition there is no signal-oriented method to show that one set of frames is more semantically informative than the next. The question is an empirical one that may be answered by psychology experiments. For now we need to rely on readers' and viewers' own reactions. The second problem is that hardcopy is a poor medium for displaying the results; we hope that interested readers will view the discussed sequences on the Web.

The original data that we use to demonstrate the coding technique has been temporally down-sampled to 10 Hz, running approximately three and one half minutes for a total of 2200 frames. A clip of the original video (and audio) that is particularly rich with gestures is seen at "natural gesture sequence 1"[3]. For this example we chose an average coding frame rate of 1 Hz, requiring we select only one tenth of the frames. The "straw man" coding scheme where we simply include every tenth frame is shown at "natural gesture sequence 2". Notice how the video appears disconnected from the narration, as if the frames were selected by someone not listening to the story.

To generate a semantically sensitive coding of the sequence, we use the distinction between bi-phasic and tri-phasic gestures to make a more informed frame selection. In this experiment, we varied the frame rate according to the presence of a tri-phasic

[3]See the index at `http://www.media.mit.edu/~drew/movies`.

gesture, using a higher frame rate to code tri-phasic gestures and a lower frame rate elsewhere.

Our results are shown in two sequences. In both instances the tri-phasic frame rate is set to be ten times the frame rate of the rest of the sequence (rest states or bi-phasic gestures). For purposes of comparison, the average frame rate was held to that used to construct the straw man sequence (1 Hz). The first sequence ("natural gesture sequence 3") is generated using the manually annotated labels described in the previous section. Notice how much more fluid the viewing experience is. Even though there are long stretches with few new frames, the coordination with the story is much tighter.

Finally, we show the computer generated sequence ("natural gesture sequence 4"). Because of the similarity in labeling of the gestures, the video has a similar feel to the sequence generated from the manual annotation. Importantly, the selection of video frames seems to be sensitive to the context of the gesture, even though no content from the speech stream was extracted. The detection of semantically meaningful gestures is purely visual.

Before concluding we mention that the scenario described here is not immediately relevant to the standard one-on-one video conference systems currently available. First, a variable-rate coding scheme such as the one presented works on the premise that bandwidth can be used when it is needed and given back later (or even earlier). Second, the selection of rest states would need to be causal, not batch processed. However, the proposed scheme is immediately applicable to archival coding, where causal encoding is not required. With dynamically estimated rest states, this method is applicable also to multi-user exchanges where bandwidth can be shifted to those who need it the most.

3.2.7 Conclusion

The natural gesture research community has identified fundamental types of natural gesture. In particular the triphasic gestures assist in conveying *meaning* in dialog. We have shown how the temporal structure of a video sequence of someone relating a story can be parsed into states that segment many of the triphasic gestures. We view this work as an initial step toward incorporating gesture sensitivity into dialog understanding. We have also shown how this distinction can be made relevant to video coding.

The present system shows that online learning of gesture models from strong temporal models alone may be possible in situations where the domain is known (for example, using natural gestures in story-telling). Currently, rest state appearance models are derived from the process of fitting the temporal model; the learned rest states bootstrap the segmentation of biphasic and triphasic gestures. A similar technique may be used to learn particular gesture models (for example, a particular kind of iconic gesture that is semantically rich) so that they may be recognized at a later time.

The resulting system is robust to the usual set of transformations that trip up gesture recognition systems that train offline models encoding spatial features: changes

in viewing angle, illumination, etc. In the case of natural gestures this robustness is doubly important in light of the fact that it would be difficult to learn a compact model of, for example, biphasic gestures over a set of spatial features, since classes of natural gesture are often defined in terms of temporal structure rather than spatial characteristics. These gestures often exhibit a bewildering variety of spatial forms, and thus would be difficult to model using a standard set of spatial features (for example, the position of the hands).

However, it is important to note that while there is indeed great variety in a gesture class from session to session, informal observations indicate that the form of the gesture usually doesn't change from instance to instance. In the present work, it was noted, for example, that just a few rest-state appearance models are necessary to cover the full range of rest state frames in a long video sequence; subjects typically change their rest state infrequently or slowly. From these observations it seems reasonable to try to learn gesture models in an online fashion using temporal models to bootstrap the learning process.

In the next chapter we present a closely related realtime system in which the estimation of appearance models and the fitting of the temporal model proceed simultaneously, in much the same way that the PHMM determines the value of its parameters during runtime. The learned gesture models are then exploited later in recognition.

Chapter 4

Watch and Learn

4.1 Introduction

One of the challenges in implementing gesture recognition systems is to design gesture models that work across a wide variety of users and environments. The problem of generalization is particularly acute when computer vision techniques are used to derive features. Lighting conditions, camera placement, assumptions about skin color, even the clothing worn by the user can disrupt gesture recognition processes when they are changed in ways not seen during training.

We argue that rather than attempt to construct training ensembles that cover all possible scenarios, it is preferable to adapt existing models to the situation at hand. This chapter presents preliminary work in developing a system that learns gestures in an online manner. The only knowledge explicitly encoded into the model *a priori* is a Markov model representing the temporal structure of the gesture.

We demonstrate the technique in a simple gesture recognition system based on computer vision as input. We show that with the online adaptive approach, it is possible to use simple features that are not necessarily invariant to the usual set of transformations that disrupt recognition processes[1].

4.2 Motivation: Online Adaptive Learning of Gesture

Typically gesture recognition systems are trained by gathering a number of sequences that serve as examples of a class of gestures, and a model of the class of gestures is constructed automatically from these examples. Hidden Markov models are a popular choice to model gestures because they are easily trained and are efficient in the testing phase [63].

One of the drawbacks of this traditional approach is that the trained models only work well in testing if the situation under which the testing data are collected is

[1]This chapter appears in [77, 78].

typical of the situations in which the training sequences were collected. A common problem that has arisen in our own experiments is if there is a systematic bias in the training data with respect to the testing conditions, such as might easily be the case if the features used were not translation-invariant and the user has moved a bit. HMMs are particularly sensitive to this problem because if there is a (typically artificial) bias that well separates the gestures during training, then the HMM is likely to latch on to that distinction in allocating states that reflect the likelihood that a given machine will produce a given output.

There are two common approaches to this problem: collecting data over many different sessions, and choosing a feature space that generalizes well. By collecting data over many sessions and incorporating them all into the example set, the hope is that the resulting model will encapsulate all the kinds of variation in the gesture that the system is likely to see during runtime. The drawbacks of this approach are two-fold: first, the number of examples that are required may in fact be too great to be practical, and second, as the number of distinguishable situations increase, the model will require more and more degrees of freedom to adequately represent the set of gestures.

The second approach, that of choosing the right feature space, has the chief drawback that it is in general difficult to craft a feature set that at once collapses the variation of the signal so that a manageable number of examples are sufficient, and still allows sufficient detail that the gesture may be recognized among a set of gestures.

We argue that these difficulties may be somewhat eased if we let part of the feature selection process happen during runtime. In the next section we show how an *a priori* model of the temporal structure of the gesture when combined with constraints from context, makes runtime feature selection possible. We call this the *online adaptive learning* of gesture to differentiate it from the usual gesture recognition methodology in which the gesture models are trained off-line.

4.3 Temporal Structure, Context and Control

The idea of the online adaptive gesture learning algorithm presented here is that if the system has a representation of the temporal structure of the gesture in question and this can be combined with real-time information derived from the application context, then the situation is sufficiently constrained that a system may conduct feature selection on the fly. Then later, when context information is unavailable, the system will be able to recognize the gesture via the learned representation.

The need for context arises because typically there is insufficient structure in the temporal model to unambiguously align a given input sequence with a potential traversal of a priori defined states. For example, consider a gesture composed of "up" and "down" phases. The temporal structure of such a gesture would be represented by the periodic Markov model in Figure 4-1. If we try to align an observation sequence to the Markov model in the figure, we find there are two ways to do this. One possible outcome assigns state A a mean feature vector that we call "down" and B a mean vector of "up". The other outcome swaps the assignment of "down" and "up".

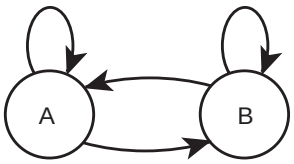

Figure 4-1: The simplest Markov model appropriate for a periodic signal. Given equal transition probabilities from state A to B, the Markov model is symmetric in A and B. Without contextual information, alignment of a signal to this Markov model would yield one of two possible equivalent assignments of semantics to A and B.

If our only concern is recognition then such a transposition is unimportant; the likelihood of the learned HMM producing the observed sequence is the same in either case. However, our goal is to use gesture for *control* of dynamic human-computer interactions. As described in section 4.7 we exploit the temporal-context sensitivity of HMMs by allowing a high likelihood of being in particular states to trigger application events. In this approach, an inversion of, say, "up" and "down" states is unacceptable. Note that the ambiguity may happen not at just the level of single states, but at the level of groups of states, such as whole gesture models.

A way to resolve this ambiguity is to resort to some external information, such as that provided by *application context*. If we can get a hint from the application to differentiate "down" from "up", the ambiguity is removed. Now that the features that correspond to the states has been unambiguously determined, the context information is longer required to perform an alignment which avoids the original ambiguity.

The learning algorithm presented incorporates the real-time learning of a hidden Markov model given application context information.

4.4 Related Work

In [80] we use a approach similar to that presented here to extract two broad classes of the natural, spontaneous gestures that people make when they are telling a story: *biphasic* gestures, which involve moving the hands out of a rest position, into the gesture space, and back to the rest position, and *triphasic* gestures, which consist of an additional stroke phase while the hands are in the gesture space. This classification scheme highlights the temporal differences of an ontology of gestures developed in [47]. A Markov model was hand-designed for each of the two classes of gesture, which differed in their temporal structure. These Markov models were then combined with image-based features derived from a long (5-minute) video sequence to derive the appearance of various rest-states used by the speaker. The present work similarly fits a hand-coded Markov model with a block of video input.

In [76] we introduce the parametric hidden Markov model (PHMM) formalism for representing a family gestures with a hidden Markov model. A PHMM encodes the manner in which the spatial form of the gesture changes as a known parameter changes. For example, the form of the gesture indicating the size of object depends on the size of the object. The PHMM testing phase involves computing the value of the

parameter that maximizes the likelihood of the gesture. PHMM output probability distributions depend on a global vector-valued parameter. The present work uses a similar style of online EM optimization to determine the value of the parameters used in HMM output state distributions.

Oliver, Pentland and Berard [54] use online EM methods to adapt color-class models in real-time for their face and lips tracking system.

4.5 Learning Algorithm

4.5.1 Expectation-Maximization Algorithm for Hidden Markov Models

A hidden Markov model uses the topology of the Markov model and its associated transition probabilities to express the temporal structure of the gesture. For example, a periodic motion may be represented by a simple Markov model with two states and transitions back and forth between them, as in figure 4-1. During testing, the Viterbi or forward/backward algorithms are used to compute the likelihood that an input sequence has the same temporal structure of the HMM, as well as match the state output probability distributions. In the process of calculating this likelihood, the forward/backward algorithm computes the posterior probability $\gamma_{tj} = P(q_t = j \mid O, \lambda)$, the probability that the HMM was in state j at time t, given the observation sequence O and HMM λ. The quantities γ_{tj} represent the parse of the HMM.

If all the values γ_{tj} are known, it is easy to see how to update the output probability distributions. For example, if the output probability distributions are Gaussian with mean μ_j and covariance Σ_j, the update equations are:

$$\mu_j = \frac{\sum_t \gamma_{tj} \mathbf{x}_t}{\sum_t \gamma_{tj}} \tag{4.1}$$

$$\Sigma_j = \frac{\sum_t \gamma_{tj} (\mathbf{x}_t - \mu_j)(\mathbf{x}_t - \mu_j)^T}{\sum_t \gamma_{tj}} \tag{4.2}$$

This is the Baum-Welch update used in training an HMM. The Baum-Welch algorithm is an expectation-maximization (EM) algorithm, where the expectation step involves calculating the γ_{tj} and the maximization step involves updating the parameters of the output probability distributions and transition probabilities.

4.5.2 Controlling the Online Adaptation

In the online adaptive learning of gesture, we use HMMs to represent the gesture we wish to recognize, but instead of running the Baum-Welch algorithm off-line, we run it during runtime to update the output probability distributions. In the present

work, we start with a known Markov model and transition probability matrix that represents the temporal structure of the gesture of interest, while the parameters of the output probability distributions are randomized at the start. As discussed in section 4.3, without some hints from application context, the states of the learned hidden Markov model may not obey the proper semantics required by the application (for example, "down" and "up" may be swapped, or the "up" gesture may be swapped with the "down" gesture).

The modification required to the Baum-Welch algorithm for it to exploit context is basically to bias the γ_{tj} after the initial computation of the expectation. Since the γ_{tj} are used as weights to define the next description of the states, the biased γ's may be thought of as an attention focusing mechanism, permitting exterior knowledge to inform the computational state that is should more aggressively learn its parameters than it would if only using the normal γ membership. In the exposition that follows we give one method to create a lattice of biased γ_{tj}, by defining a new quantity that is a linear combination of γ_{tj} and prior probabilities derived from application context.[2]

The information from application context are assumed to take the form of prior probabilities for each state:

$$\omega_{tj} = P(q_t = j \mid \Omega) \tag{4.3}$$

where Ω represents application context. These posterior probabilities are then combined with the usual HMM posterior probabilities $\gamma_{tj} = P(q_t = j \mid \lambda)$ to obtain a new posterior probability which incorporates the HMM and the application state context:

$$\Gamma_{tj} = \rho_j \gamma_{tj} + (1 - \rho_j)\omega_{tj} \tag{4.4}$$

which is subsequently normalized so that $\sum_j \Gamma_{tj} = 1$. ρ_j is a scalar quantity that is proportional to how much the HMM state j has been tuned during online adaptation.

In the current system, we set ρ_j to be proportional the number of frames for which γ_{tj} is greater than some fixed value (say, 0.7). When beginning the adaptation, ρ_j is at its minimum value, then increases to some maximum value during adaptation. The intuition is that this quantity controls the degree to which the system follows the application context versus the HMM. It also overcomes the fact that when the HMM output distribution function is Gaussian, starting with large covariances to represent uncertainty brings the distributions to zero and so the state is never exploited by the HMM. The effect of ρ_j during runtime is to artificially bias the algorithm to use neglected states.

We also incorporate a global learning rate α to control the adaptation process. The idea is that at the start of the algorithm, when the state output distributions parameters have random values the algorithm should learn aggressively, and that later when the parameters have approached good "final" values the algorithm should change the values less aggressively. This prevents the algorithm from changing the gesture model to match some spurious input.

In the present system α is derived from the confidence value ρ_j described above.

[2]In the present system, we implement this bias by altering the form of the output probability distribution rather than by directly manipulating γ_{tj}.

We currently set the relationship between ρ_j and α in an *ad hoc* manner. For a state which has seen no probability mass γ_{tj}, we would like quantity to be 1.0. It is important that the learning rate always have some value greater than zero, so that the algorithm can continually adapt to slow changes in the gesture signal. Optionally, we normalize α by the frame rate of the online EM process.

The learning rate α is incorporated in the EM update by simply mixing the old value of the parameter with the new value:

$$\mu'_j = (1-\alpha)\mu_j + \alpha \sum_t \Gamma_{tj} \mathbf{x}_t \tag{4.5}$$

The quantities $P(\mathbf{x}_t \mid q_t = j, \lambda)$ are computed over the sequence $\langle \mathbf{x}_{t-T} \ldots \mathbf{x}_t \rangle$, and the EM algorithm is run once over the sequence. At some time $t + \Delta t$, this process is repeated (caching values where possible) over the next window $\langle \mathbf{x}_{t+\Delta t-T} \ldots \mathbf{x}_{t+\Delta t} \rangle$, and so on, throughout the lifetime of the application – there are no distinct training and testing phases.

4.6 Images as Input

4.6.1 Tracking

The *Watch and Learn* system uses the online adaptive algorithm described above with whole images as input. Color images are acquired at a rate of 30Hz from a camera pointed at the user. A body-centric image of the user is derived from the input image by subtracting the pixel values of the image of the scene without the user (the background image) from the current image. This difference image is then binarized to obtain a silhouette image. A simple EM-based tracking algorithm updates the center of the image to follow the center of the silhouette. The body-centric silhouette image is then multiplied by the original image to obtain a color image that is body-centric and does not include the background.

The EM-base tracking algorithm models the spatial distribution of the silhouette pixels over the image as a Gaussian with fixed covariance.

$$h_{x,y} = \frac{1}{\sqrt{2\pi \mid \Sigma \mid}} e^{-\frac{1}{2}\left([\mathrm{x\ y}]^T - \mathbf{c}\right)^T \Sigma^{-1} \left([\mathrm{x\ y}]^T - \mathbf{c}\right)} \tag{4.6}$$

$$\mathbf{c}' = \sum_{D_{x,y} > b} h_{x,y} [\mathrm{x\ y}]^T \tag{4.7}$$

where $D_{x,y}$ is the absolute difference of the current image pixel at (x, y) and the background image pixel at the same location, $\mathbf{c}$ is the mean from the previous time step, b is the threshold for binarizing the image and Σ is the constant diagonal covariance matrix that is chosen to approximate the size of the user in the image.

$h_{x,y}$ is calculated over a window centered about $\mathbf{c}$. If the likelihood of this model falls below a threshold, the algorithm enters a seek mode in which the mean $\mathbf{c}$ is repeatedly assigned a random value until the likelihood $h_{x,y}$ is above the threshold.

Otherwise, the mean of the Gaussian is updated to reflect the translation of the silhouette.

4.6.2 Output Probability Distribution

The pixel values of the cropped color foreground image centered about $\mathbf{c}$ at time t are concatenated to form the feature vector $\mathbf{x}_t$. The last two seconds of the color foreground images are saved in memory. These form the buffer over which the online adaptive EM algorithm updates the output probability distribution parameters. The output probability distributions $b_{jt} = P(\mathbf{x}_t \mid q_t = j)$ take the form:

$$b_{jt} = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2\sigma_j^2 XY}(\mathbf{x}_t - \mu_j)^T(\mathbf{x}_t - \mu_j)} \tag{4.8}$$

where σ_j is a scalar, and X and Y are the dimensions of the image $\mathbf{x}_t$.

The output probabilities $b_{jt}(\mathbf{x})$ are computed over all states j for the current time step only; the values are saved in a buffer of the last T time steps. Updating the output probability distribution parameter μ_j proceeds as equation 4.1, here involving the weighted sum of the images in the image buffer. The update of σ_j is similarly a weighted sum:

$$\sigma = \frac{\sum_t \Gamma_{tj}(\mathbf{x}_t - \mu_j)^T(\mathbf{x}_t - \mu_j)}{\sum_t \Gamma_{tj}} \tag{4.9}$$

After each maximization step, it would be correct to recompute the value of the output probability distributions for each state and each image in the image buffer, since the parameters of the distribution have changed by the update equations. In the present system, however, we do not recompute these likelihoods for the reason that much computation may be avoided by computing the likelihoods for only the newest image. If the buffer is small and the changes in the parameters are continuous, then the outdated values of the likelihood associated with the oldest frames in the buffer will not upset the learning algorithm. Empirically we have found this to be the case.

In the *Watch and Learn* system, computing the weighted sum of images for the maximization step is the most computationally intensive step of the algorithm and need not be executed at every new time step. Thus with the current system, the input image buffer is updated at 30Hz, while the learning algorithm executes at no more than 4Hz. Both the expectation and maximization steps of *Watch and Learn* have been implemented to use MMX single instruction/multiple data (SIMD) instructions available on the Intel Pentium II processor.

4.7 Application: Conducting

One activity in which there is strong contextual information is musical conducting, where both the musicians and the conductor follow a score. The *Watch and Learn* system has been applied to a simplified conducting scenario to demonstrate that a

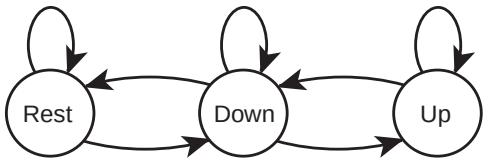

Figure 4-2: The Markov model used to represent the temporal pattern of a beat in the *Watch and Learn* system applied to the simple conducting scenario.

simple beat gesture may be learned by adaptive online learning. A video of the system is located at `http://www.media.mit.edu/~drew/movies`.

The interaction between the user who plays the role of the conductor and the system is as follows. The user steps in front of the camera and waits for the system to play a series of beats (wood block sounds) that establish a tempo. After a few beats, the user begins to follow the beat with his hand. After a few bars of following the system's beat, the system begins to play a piece of music. Having taught the system his own beat gesture, the user is now free to change the tempo of the piece currently playing.

For the simple beat pattern described, a simple three state Markov model is used to model the temporal structure of the gesture (see figure 4-2). The Markov model begins in a rest state, which is learned at first when the user is standing in front of the camera waiting for the system to establish a beat. During the fourth beat generated by the system, the user is supposed to have begun following the beat with his gesture. At this point, contextual priors are changed to match the expectation that at the instant of the system-generated beat, the user should be in the "downbeat" state. Given that at this stage in the learning the rest state has already been learned by the system, the "upbeat" state will be learned correctly because temporal structure provided will lead the system to ascribe the moments before the downbeat to the "upbeat" state, and furthermore, presumably the images during the actual upbeat motion will look different than the "rest" state.

As the user counts out the beats, the appearance models (the means of the output probability distribution) associated with each state gradually appear as reasonable approximations to what an observer might call the "upbeat", "downbeat" and "rest" phases of the gesture. Figure 4-3 shows a typical set of appearance models for the beat Markov model. Figure 4-4 shows γ_{tj} over the last 64 frames (about 2 seconds) of video. The system is continually adjusting these states to reflect the subtle changes in the way the user executes the gesture from instance to instance.

We wish to remind the reader that *Watch and Learn* in no way attempts to track the user's hands; it is purely through the combination of the temporal structure model and the contextual information that gives rise to the semantically correct appearance models. Ultimately, the most important requirement is that the user be *cooperative*, especially during the periods of high learning rate, and *consistent*. It is quite possible to train *Watch and Learn* to recognize foot tapping instead of hand beats, as long as the user consistently does so.

Changes in tempo are made in a very simplistic manner according to the rise and

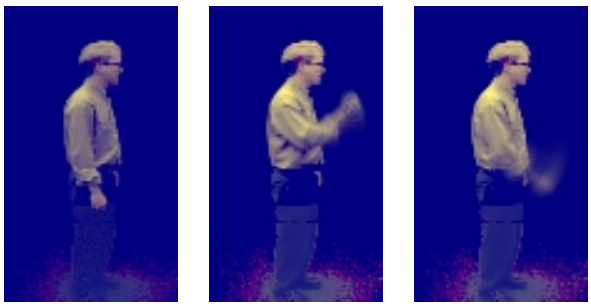

Figure 4-3: The appearance models (images) associated with each state of the beat HMM, after online adaptation. When the algorithm is started, the pixels values of the images are randomized.

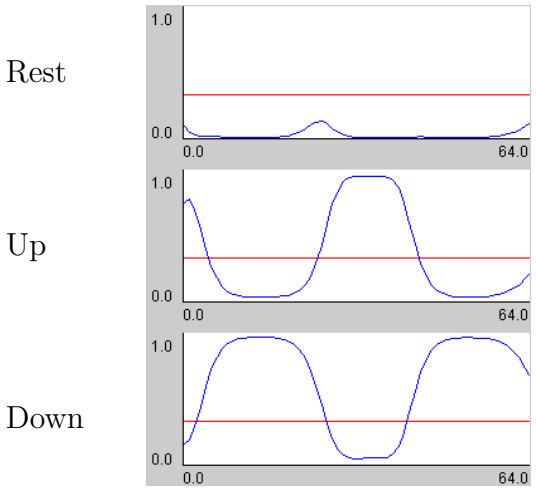

Figure 4-4: Plots of γ_{tj} for the beat gesture, after online adaptation.

fall of $\gamma_{t,up}$: when $\gamma_{t,up}$ falls below a certain threshold, a "beat" event is generated. The time between the current beat and the last beat is calculated, converted to MIDI clock units and passed on to the MIDI time-keeper running on the host computer. Presently, no attempt is made to synchronize where the particular downbeat falls with the downbeat in the score. If at some point the user returns to the rest state, tempo changes are not made and the piece continues playing at the last tempo.

4.8 Discussion and Future Work

An online adaptive learning algorithm for learning gestures has been presented. The approach differs from the usual train/test paradigm in that much of the training process may occur online. The algorithm requires a Markov model that represents the temporal structure of the gesture to be learned. This is combined with contextual information to train the output probability distributions during runtime. *Watch and Learn* succeeds in learning a simple beat pattern, and in another configuration has been applied to learning a mapping to various drum sound patches with a slightly more complex temporal model (see figure 4-5).

We argue that the problem of generalization by feature selection is eased with the online adaptive learning algorithm presented above. By delaying the estimation of the output probability density parameters to runtime, the online algorithm is free to choose only those values which fit the current set of data. Thus any particular bias in the features present in runtime that would have upset an off-line approach is absorbed in the online learning process.

The net result is that with the online algorithm, feature selection is not as crucially important as with the off-line algorithm in obtaining generalization performance. As long as the features are consistent over the set of learned states, the online algorithm will set the output probability distribution parameters appropriately. For example, image space itself may make an extremely poor feature space for many gesture applications because many of the usual desired invariants are absent.

Although in general computer vision has been dismissed as a technique useful to computer music on the grounds that the techniques are too computationally complex to run quickly on today's hardware, we note without hard justification that *Watch and Learn* is quite responsive. One reason for this is the fact if events are triggered from γ_{tj}, the temporal model enables the system to anticipate events: for example, a MIDI note-on event may be generated at the moment that $\gamma_{t,up}$ begins to fall below a threshold, which is in a moment in advance of the peak of $\gamma_{t,down}$ (see figure 4-4). Also recall that $\gamma_t j$ is being updated at 30Hz.

There are two drawbacks to the *Watch and Learn* system as it is currently implemented. First, the system assumes that the user is being cooperative at all times. This drives the learning initially, but can be a problem once gesture models have been learned. For example, once the beat gesture is learned reliably, if the user does a completely different gesture, this new movement should not be incorporated into the model. However, if the gesture appears to have the same temporal structure as the original beat, and occurs at the moment in time during which the system expects

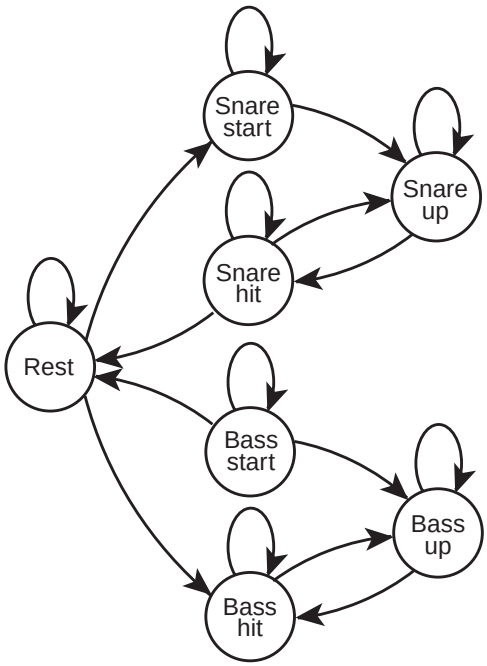

Figure 4-5: The Markov model used for a drum-set configuration of *Watch and Learn*. MIDI events are generated when the Markov model enters the "Snare hit" and "Bass hit" states. "Snare start" and "Bass start" states capture the preparation of the gesture and do not generate MIDI events. The appearance models of the start states may adapt to resemble those of the hit states, thus they are necessary to prevent spurious MIDI events during the gesture's preparation phase.

a beat, the system should incorporate the new information.

The second drawback to the *Watch and Learn* system is the *ad hoc* manner in which the confidence values ρ_j and the learning rate α is determined. We expect to incorporate more principled ways of controlling the adaptation.

The next chapter outlines a framework that in part is designed to address these drawbacks.

4.8.1 Learning a Saliency Map

Watch and Learn employs a very coarse notion of a focus of attention by tracking the user and deriving body-centered images from video input. This broad focus of attention does not address the situation in which the user, for example, trains the system on a hand gesture, and then shifts his leg to a new position. As discussed earlier, if this change is slight in the image, the online adaptation process may allow the incorporation of this new pose into the appearance model without disrupting the gesture recognition process. If the change is large in the image, however, the input image may not be similar to any stored appearance models, and gesture recognition and adaptation processes may be lost. A narrower focus of attention that picks out parts of signal to attend to is desirable.

Watch and Learn may be extended to incorporate learned *saliency maps* which indicate which parts of the image to attend to in its computation. A variety of approaches may be investigated, but the most computationally feasible and motivated approaches will involve using image motion as a cue for saliency over the image, rather than the computationally more expensive discriminant-analysis techniques. The following outlines an approach based on image motion.

During the parsing of gesture, the underlying cause of moving from one gesture state to another is change in the image. Intuitively, in order to see transitions from state to state, the system need only then pay attention to regions of the image that tend to change over the course of the gesture. During the adaptation process, a saliency map over image may be computed which indicates which pixels were changing while the gesture model was in a particular state j. This may be accomplished in a variety of ways; one approach is to learn the variance of the pixel color at each location throughout the image, using the usual M-step equations:

$$\sigma_j^2(x,y) = \frac{\sum_t \gamma_{tj}(I(x,y) - \mu_j(x,y))^T(I(x,y) - \mu_j(x,y))}{\sum_t \gamma_{tj}}$$

where $I(x,y)$ is the vector-valued pixel color at (x,y), and $\mu_j(x,y)$ is the value of μ_j corresponding to image coordinates (x,y).

This variance may then be incorporated into a probabilistic saliency map, where $s = \text{ON}$ and $s = \text{OFF}$ indicates whether the pixel is salient (ON) or not (OFF).

$$\begin{aligned} P((x,y) \mid s_j(x,y) = \text{OFF}) &= \mathcal{N}(0, \sigma_j^2(x,y)) \\ P((x,y) \mid s_j(x,y) = \text{ON}) &= 1 - P((x,y) \mid s_j(x,y) = \text{OFF}) \end{aligned}$$

The usual output probability distribution function may incorporate the likelihood of the pixel belonging to the map:

$$b'_{jt}(x,y) = b_{jt}(x,y)P((x,y) \mid s_j(x,y) = \text{ON})$$

Note there are many possible ways to incorporate $P((x,y) \mid s_j(x,y))$ but that incorporating it directly into the original b_{jt} form is complicated by the fact that s_j is not a measure of uncertainty in the usual interpretation of Gaussian distributions; this behavior is opposite of what is desired. Furthermore, by keeping the saliency distribution separate from the usual output distribution, we reserve the ability to differentially weight the two quantities.

A simple variant on this approach is to build a saliency map based on image motion independent of the HMM state j. That is, image motion is treated as a saliency cue independent of the gesture state. Note that in an implementation, this could amount to modifying the tracking algorithm of Section 4.6.1 to follow regions of the image which exhibit movement rather than regions which are sufficiently different from the background. Additionally, the window size of Section 4.6.1 could be made to change according to the size of the region. If the window size and location is constrained to change slowly, the system will effectively ignore extraneous movement outside the gesture space.

Chapter 5

Hybrid Bayesian Networks

5.1 Introduction

Bayesian networks, or *graphical models* or *probabilistic independence networks* as they are also known, have become increasingly popular probabilistic modeling techniques. In the past, discrete-valued Bayesian networks had been popularized for their application in expert systems [57], but more recently their relationship with other familiar techniques such as hidden Markov models, Kalman filters and factor analyzers has been established. Consequently, much recent work in machine learning and computer vision has applied Bayesian networks to extend and combine the previous approaches (see [56] for example).

In this chapter I briefly outline the methodology of a particular flavor of Bayesian networks that will be useful in uniting the mathematical structures of the previous chapters. The hybrid Bayesian network will prove its worth in subsuming hidden Markov models and allowing various extensions and combinations with other types of structures. In particular I will consider the interpretation of the Parametric Hidden Markov Model (PHMM) as a hybrid Bayesian network. Finally, the presentation of an implementation of dynamic Bayesian networks will serve to lay the groundwork for the final work presented in the next chapter.

5.2 Bayesian Networks Basics

5.2.1 Random Variables

Suppose we would like to model the probability distribution over a set of random variables A, B, C, D, E, F[1]. We may chose any number of techniques to model this joint probability $P(A, B, C, D, E, F)$ directly, such as kernel estimation, but for anything but a toy problem the modeling of the full joint probability will be computationally intractable. By our knowledge of the phenomenon we are trying to model we may know something about the conditional independence of the random variables.

[1]This section borrows examples and outline from [39].

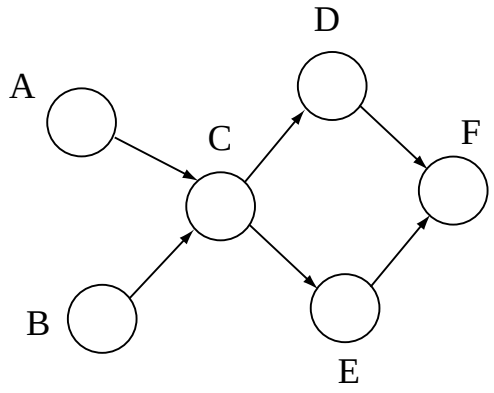

Figure 5-1: Graph corresponding to equation 5.1.

With this knowledge we may factor this distribution into a product of conditional probabilities. For example, we may have

$$P(A, B, C, D, E, F) = P(A)P(B)P(C \mid A, B)P(D \mid C)P(E \mid C)P(F \mid D, E) \quad (5.1)$$

Graphically, we may depict this factorization as a graph with a node for each random variable and where the *lack* of an arc from A to B expresses the conditional independence of B on A. The graph corresponding to the above distribution is shown in Figure 5-1.

Furthermore we may without loss of generality rewrite the distribution as a product of *potentials* over subsets of variables:

$$P(A, B, C, D, E, F) = \phi(A, B, C)\phi(C, D, E)\phi(D, E, F)$$

where

$$\phi(A, B, C) = P(A)P(B)P(C \mid A, B)$$

and

$$\phi(C, D, E) = P(D \mid C)P(E \mid C)$$

and

$$\phi(D, E, F) = P(F \mid D, E)$$

The link between the potentials and the corresponding graphical diagram may be established by drawing an undirected arc between each pair of parents of every node (this process has been coined *moralizing* the graph). If we then change all directed arcs into undirected arcs, notice that each clique in this new graph shown in Figure 5-2 corresponds to a potential above. The potentials are often called *clique potentials*.

5.2.2 Inference

Starting from the graph in Figure 5-2 we may construct a new graph which has a node for each clique in the moralized undirected graph and an arc between each pair of nodes V and W for which the intersection of the variables in cliques V and W,

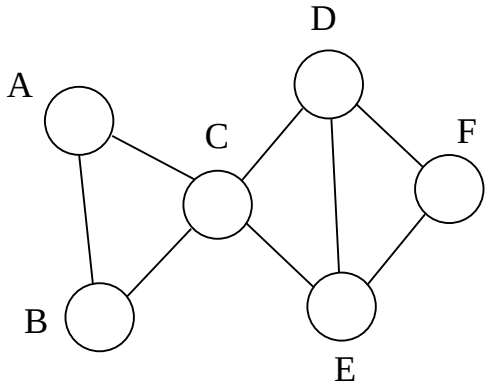

Figure 5-2: The moralized, undirected graph.

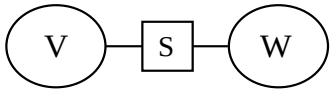

Figure 5-3: Cliques V and W, with separator S

$S = V \cap W$ is non-empty. We may construct a further graph which places a node for this intersection S between the connected nodes V and W, as in Figure 5-3. The resulting graphs are shown in Figure 5-4. Each node S is termed a *separator*.

Recall that with each clique there is a clique potential which assigns a value to each configuration of variables in the clique. Put another way, the potentials make assertions on the values of configuration of the variables. Intuitively, the separators are added to the graph to indicate graphically that since cliques V and W share the variables in S, their potentials must be made to "agree" on the value of the variables in S.

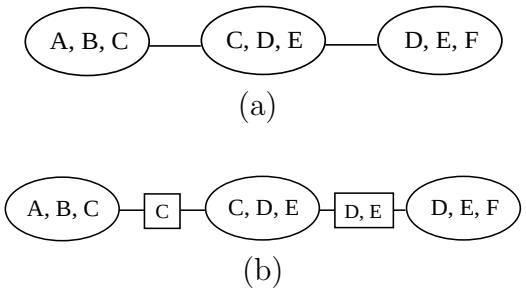

Figure 5-4: (a) A clique graph for the graph in Figure 5-2, and (b) clique graph with separators.

The clique potentials V and W are made to agree on the distribution on the variables in S by *marginalizing* and *rescaling*:

$$\phi_S^* = \sum_{V \setminus S} \phi_V$$

$$\phi_W^* = \phi_W \frac{\phi_S^*}{\phi_S}$$

where $V \setminus S$ denotes the elements of V not in S, and ϕ_S denotes a potential on S. This process is sometimes called *absorption*. W is said to *absorb* from V, giving updated potentials ϕ_{W^*} and ϕ_{S^*}.

When the potentials agree on the distribution of variables in each separator S, the network is said to be *globally consistent*. The inference process on Bayesian networks amounts to ensuring global consistency. This may be achieved by having each node absorb from its neighbors in a parallel, distributed fashion throughout the net. In practice, this may be achieved in a serial fashion in a two stage process: pick a root R in the clique tree; successively absorb from the leaves of the tree up to the root and then successively absorb from the root to the leaves. After this process has completed, the network will be globally consistent[34]. Note that in the process of insuring global consistency only the successive application of *local* algorithms is employed.

This two-stage process is termed *probability propagation*, *belief propagation* or the *message passing* algorithm for Bayesian networks. Typically the propagation algorithm is run after evidence on the value of some set of nodes is collected. Evidence on some variable A is incorporated in the network by picking any clique which contains A, changing its associated potential to reflect our revised knowledge about the value of A, and running the propagation algorithm to restore global consistency. The posterior distribution on some other node B may then be computed by picking any clique that contains B and marginalizing over all variables in the clique other than B. Likewise, the likelihood of the total network may be computed after propagation by picking any clique and marginalizing the corresponding potential over all variables in the clique. In the marginalization process, the potentials must be normalized so that the resulting marginal is a probability distribution.

5.2.3 Triangulation

There is one unfortunate complication to this simple local message-passing algorithm that must be addressed. Figure 5-5 illustrates a directed graph and its moralized, undirected graph. Two possible clique trees may be constructed from the undirected graph.

Note that in the clique tree of Figure 5-5(c) the random variable C appears in two non-neighboring cliques. Unfortunately, there is no guarantee that after the propagation algorithm the distribution over C in each of the cliques will agree. Similarly, the clique tree of Figure 5-5(d) has E in two non-neighboring cliques.

The solution is to *triangulate* the moralized, undirected graph. A triangulated graph has no cycles with four or more nodes without a chord. A graph may be

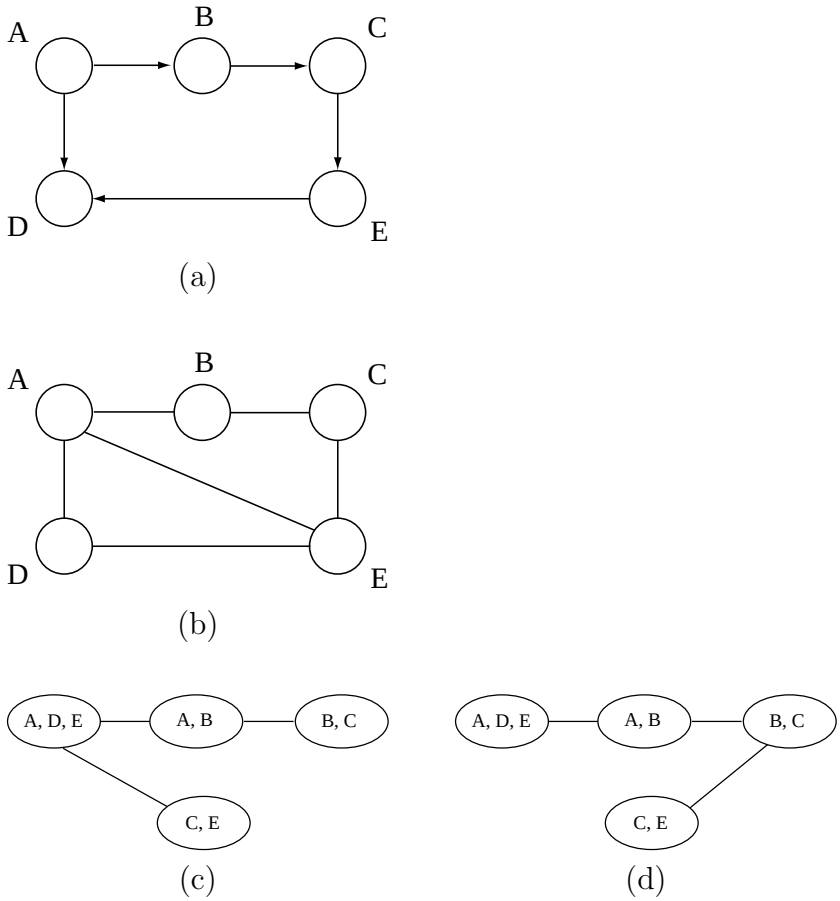

Figure 5-5: The following illustrates the construction of a clique tree on a simple graph: Starting from a directed graph (a) on A, B, C, D, E, the moralized undirected graph (b) is constructed. Two clique trees (c) and (d) may be constructed from the moralized undirected graph.

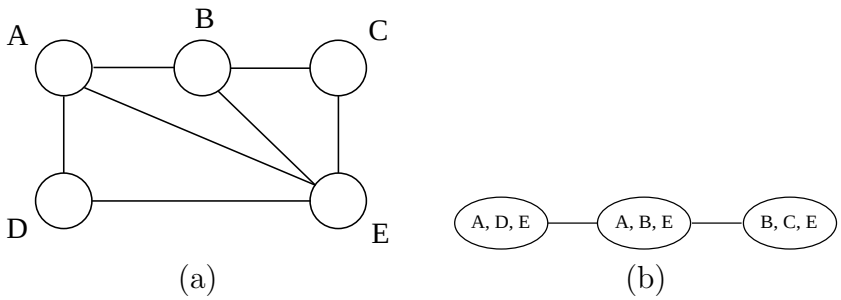

Figure 5-6: (a) The triangulated, moralized version of the graph of Figure 5-5 and (b) its clique tree.

triangulated by adding undirected edges until the graph is triangulated.

Propagation on the clique-tree made from the triangulated graph is guaranteed to insure global consistency [34, 18]. To understand why, consider that the clique tree computed from the triangulated graph will obey the *running intersection property*, which insures that if a node appears in two cliques it will appear on every node on the path between the cliques. Figure 5-6 shows the triangulated graph of Figure 5-5 and its associated clique tree.

In the triangulation process we would like to add as few edges to the graph as possible, since each edge added increases the size of some clique and thus increases computational complexity of any absorb operations involving the clique. Unfortunately, determining an optimal triangulation is NP-hard. However, there are a number of good heuristic-based polynomial-time triangulation algorithms available which I will not cover here.

From here on I will assume that the clique tree on a Bayes network was computed from the triangulated graph. The process of moralizing, triangulating and propagation is often referred to as the *junction tree algorithm*, in which the clique-tree is likely to be called the *junction tree*.

5.2.4 Learning

Beyond the inference process, we also require a method to estimate the parameters of the conditional probability distributions on the random variables of the network. Here I briefly describe the Expectation-Maximization (EM) algorithm applied to parameter estimation in Bayesian networks.

In the EM algorithm for Bayesian networks, we call each variable for which we have collected evidence *visible* or *observed*, and the rest are termed *hidden* variables. In general, the EM algorithm involves computing the distribution over the hidden variables given the observed variables (the "expectation" step). Note that in the case of Bayesian networks, this is exactly the inference process described above, followed by a marginalization for each node to be estimated. In the "maximization" step of EM, the expectations are combined to arrive at updated parameter values.

For an ensemble of training examples, the EM algorithm successively enters each set of data on the network, runs the probability propagation algorithm, computes

marginals on the nodes to be estimated and updates the sufficient statistics for each parameter to be estimated. After statistics have been collected for each example, the parameters for each node may be estimated appropriately.

5.3 Hybrid Bayesian Networks

In the presentation of Bayesian networks above, the form of the conditional probability distributions and the clique potentials has not been specified. In the most common case where all the random variables in the network are discrete-valued, the conditional probabilities and clique potentials are stored as conditional probability tables (CPTs) in which each configuration of the set of variables over which the distribution or clique is defined is assigned a probability. In the discrete case, the marginalization and rescaling operations in the absorb operation involve straightforward arithmetic on CPTs.

In the case where some of the random variables are continuous-valued, we may discretize them and treat them thereafter as discrete variables. The drawback of course is that the computational complexity of the absorb operation is sensitive to the level of discretization. Or if the continuous variables are always observed leaf nodes in the network, then we may safely remove them from the network if we change the distribution over the removed nodes (discrete) parents for each observation.

There are interesting cases for which we have continuous hidden nodes in a network that otherwise includes discrete nodes as well; these we will call *hybrid* networks. In particular, the PHMM may be cast as a hybrid Bayesian network, as may switching Kalman filters and many other combinations of HMMs, mixture models and Kalman filter-like structures that have no names. If we have hidden continuous nodes that we would not like to discretize as well as discrete nodes in the network, we may employ the *conditional gaussian* (CG) form of potentials as developed by Lauritzen[45, 44, 18].

In the following, we outline the form of the CG potential, and how hybrid Bayesian networks may be applied.

5.3.1 Conditional Gaussian Potentials

We will assume that each continuous variable in the network is modeled by a Gaussian distribution, and a discrete variable may not have a continuous variable as a parent. We must specify a form for the distribution of a continuous variable which may depend on the values of discrete as well as continuous parents. If we take the continuous variables to be modeled as Gaussian distributions, we arrive at forms that lead to mixtures of Gaussians. The following cases arise for a continuous variable y:

- y has no parents. We have $y \sim \mathcal{N}(\mu, \Sigma)$.
- y has a discrete parent q. We have $y \mid q = i \sim \mathcal{N}(\mu_i, \Sigma_i)$, a mixture of Gaussians.
- y has a continuous parent x. We have $y \mid x \sim \mathcal{N}(\mu + Wx, \Sigma)$. Essentially, the means are moved in response to changes in x.

- y has a discrete parent q and a continuous parent x. We have $y \mid x, q = i \sim \mathcal{N}(\mu_i + W_i x, \Sigma_i)$, the most general scenario.
- y has multiple discrete parents. The forms for the case of a single discrete parent are applied on the Cartesian product of the set of discrete variables.
- y has multiple continuous parents. The forms for the case of a single continuous parent are applied on the joint space of the set of continuous variables; i.e. the parents x_i are concatenated.

With these simple forms it is possible to implement many of the probabilistic structures of interest. The restriction that no discrete variable may have a continuous variable as a parent arises from the fact that any combination of the above forms will result in a mixture of Gaussians, which is amenable to exact analysis, whereas a discrete variable which depends on a Gaussian variable is something other than a mixture of Gaussians. In practice, this restriction turns out not to matter so much, as we can may usually reverse the direction of the dependency by invoking Bayes rule.

With these forms a potential in a hybrid Bayesian network is essentially a mixture of gaussians. A CG potential is defined over a set of discrete and continuous variables. As formulated in [18] the moment characteristics of a Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ with $p(i) = P(q = i) > 0$ are converted into their *canonical* characteristics $(g(i), h(i), K(i))$ with

$$\begin{aligned} K(i) &= \Sigma(i)^{-1} \\ h(i) &= K(i)\mu(i) \\ g(i) &= \log p(i) + \{\log \det K(i) - d \log(2\pi) - \mu(i)^T K(i)\mu(i)\}/2 \end{aligned}$$

where d is the dimension of the joint space of the continuous variables on which the potential is defined.

Concatenating the continuous variables over which a potential is defined as y, a CG potential over a set of discrete and continuous variables may be written as

$$\phi(i, y) = e^{g(i) + h(i)y - yK(i)y/2} \tag{5.2}$$

In the case where where a CG potential is constructed from a continuous variable with a continuous parent, h and K incorporate the regression matrix W

$$h = \begin{pmatrix} -W^T\Sigma^{-1} & \Sigma^{-1}\mu \end{pmatrix}^T \tag{5.3}$$

$$K = \begin{pmatrix} W^T\Sigma^{-1}W & -W^T\Sigma^{-1} \\ -\Sigma^{-1}W & \Sigma^{-1} \end{pmatrix} \tag{5.4}$$

where the indices i have been omitted for clarity. This follows from rearranging the terms of the expression for the conditional probability distribution for a Gaussian random variable with continuous parents.

5.3.2 Operations on CG potentials

Recall that the absorb operation requires the multiplication, division and marginalization of potentials. When multiplying or dividing potentials ϕ_U defined over variables U and ϕ_V defined over V we assume that the domains of both potentials are first extended to $U \cup V$ by padding h and K with zeros appropriately.

With the domains appropriately extended, multiplication is a matter of adding the canonical characteristics

$$\phi_U \phi_V = (g_1, h_1, K_1) \times (g_2, h_2, K_2) = (g_1 + g_2, h_1 + h_2, K_1 + K_2)$$

Division is similarly defined by subtracting characteristics:

$$\phi_U / \phi_V = (g_1, h_1, K_1)/(g_2, h_2, K_2) = (g_1 - g_2, h_1 - h_2, K_1 - K_2)$$

with appropriate guards against division by zero. These equations follow directly from expression 5.2.

For the marginalization operation, we must consider marginalizing out both continuous variables and discrete variables. We first address the easier case of marginalizing over a set of continuous variables. Again, denoting a value from the joint space of all continuous variables over which a potential is defined as y, we consider marginalizing over y_1 where

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad h = \begin{pmatrix} h_1 & h_2 \end{pmatrix}, \quad K = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}$$

we have

$$\begin{aligned} \tilde{g}(i) &= g(i) + \{p(i)\log(2\pi) - \log\det K_{11}(i) + h_1(i)^T K_{11}(i)^{-1} h_1(i)\}/2 \\ \tilde{h}(i) &= h_2(i) - K_{21}(i) K_{11}(i)^{-1} K_{12}(i) \\ \tilde{K}(i) &= K_{22}(i) - K_{21}(i) K_{11}(i)^{-1} K_{12}(i) \end{aligned}$$

to obtain the updated potential characteristics $\tilde{\phi} = (\tilde{g}(i), \tilde{h}(i), \tilde{K}(i))$.

Next we consider marginalizing over discrete variables. With the full set of discrete variables now defined on $\mathcal{I} \times \mathcal{J}$, we first consider the case when marginalizing over j where $h(i)$ and $K(i)$ do not depend on j. That is $h(i,j) = h(i)$ and $K(i,j) = K(i)$. This may be the case when all the components of a mixture are equal, or when the potential is defined only on discrete variables. Then the marginal involves the sum of the scalar components:

$$\tilde{g}(i) = \log \sum_j e^{g(i,j)}$$

and $\tilde{h}(i) = h(i)$ and $\tilde{K}(i) = K(i)$.

In the most general case when h and K do depend on j, we unfortunately are confronted with the fact that when adding two CG potentials the result is not necessarily a CG potential but is a mixture of CG potentials. One approach is to convert

the CG potential back to its moment characteristics and compute the marginal as

$$\tilde{p}(i) = \sum_j p(i,j) \tag{5.5}$$

$$\tilde{\mu}(i) = \sum_j \mu(i,j)p(i,j)/\tilde{p}(i) \tag{5.6}$$

$$\tilde{\Sigma}(i) = \sum_j \Sigma(i,j)p(i,j)/\tilde{p}(i) + \sum_j (\mu(i,j) - \tilde{\mu}(i))^T(\mu(i,j) - \tilde{\mu}(i))p(i,j)/\tilde{p}(i) \tag{5.7}$$

Note that this is an approximation to the true marginal, and that the resulting approximate marginal will have the same moments of the true marginal. This approximation is termed a *weak* marginal, and will have consequences for the construction of junction trees, as will be described later.

Lastly, we require the ability to enter evidence on a clique potential. If the observation is on a discrete variable, we simply set to zero all p_i but the value of i that corresponds to the observed configuration, in a procedure exactly analogous to the discrete network case.

If the observation is on a continuous variable, we must modify the domain of canonical characteristics. If observation y is on domain γ in a potential that contains γ defined as

$$h = \begin{pmatrix} h_1 \\ h_\gamma \end{pmatrix} K = \begin{pmatrix} K_{11} & K_{1\gamma} \\ K_{\gamma 1} & K_{\gamma\gamma} \end{pmatrix}$$

the revised characteristics are then

$$\begin{aligned} \tilde{K}(i) &= K_{11}(i) \\ \tilde{h}(i) &= h_1(i) - yK_{\gamma 1}(i) \\ \tilde{g}(i) &= g(i) + h_\gamma(i)y - K_{\gamma\gamma}(i)y^2/2 \end{aligned}$$

Because this process changes the dimensionality of h and K we must enter the evidence on all potentials that contain γ.

5.3.3 Strong Junction Tree Algorithm

There is one last complication we must attend to before considering parameter estimation in hybrid Bayesian networks, and that involves the consequences of the weak marginalization operation discussed in the previous section. As an approximation to the true marginal, a weak marginalization during the two-phase probability propagation algorithm will render further absorb operations approximate as well.

In fact, if a weak marginalization occurs during the first pass of the propagation, in which absorb operations happen from the leaves to the root of the tree, the guarantee of global consistency of local probabilities throughout the network after propagation no longer holds. Intuitively, this is because during the first phase the root absorbs all information available in the network. Global consistency is achieved when this complete information is then propagated outward to the rest of the network during the second phase. If an approximation occurs in the first phase, the root will not have

seen all the information available, and the subsequent second propagation will thus not be able to communicate all the information required to achieve global consistency.

One way to restore the guarantee of consistency is to arrange the junction tree such that no weak marginalizations are performed on the first pass of the propagation algorithm. Weak marginalizations in the second (outward) phase of propagation are allowed, since during the outward pass the information that is lost via weak marginalization during an outward pass is not needed subsequently in the propagation anyway. Such a junction tree is called a *strong junction tree*, with a *strong root*.

Computing a strong junction tree proceeds similarly as described earlier. To insure that there are no weak marginalizations during the first phase of propagation, it suffices to change the elimination order in the triangulation algorithm such that all discrete nodes are eliminated first. Also, the strong root of the tree may be determined during triangulation. This is enough to insure that when a separator between two neighboring cliques is not purely discrete, the clique further away from the root will have only continuous variables outside the separator. Thus weak marginalization will occur only in the second phase.

Now, since after propagation over a strong junction tree we have consistency, even though the second phase may involve approximate weak marginals, the marginals are the correct weak marginals of the full joint distribution. Thus we may compute the (possibly weak) marginal of any variable from any clique in which the variable appears and get the same correct distribution.

The drawback of using the strong junction tree is that because more constraints have been placed on the triangulation algorithm, it is possible that more edges will need to be added to the network during triangulation. With more edges added, the size of the cliques may increase, and so goes the computational complexity of the inference algorithm.

5.3.4 Learning: Fitting a CG Distribution

Lastly, we briefly address parameter estimation in hybrid Bayesian networks by the EM algorithm, as developed in [50, 51].

In the expectation step, the lth training observation e_l is entered on the network as evidence, the propagation algorithm is executed and for each variable y the posterior marginal on y's family is computed using the marginalization operation described above.

We are interested in computing the second conditional moments of the continuous variables in y's family. As usual i refers to a configuration over the Cartesian product of the discrete variables in y's family, which themselves are denoted as Q. Second conditional moments over any variable x is defined as

$$E[q_l^i x_l x_l^T \mid e_l] = E[q_l^i \mid e_l] E[x_l x_l^T \mid Q_l = i, e_l] \equiv w_l^i \langle x_l x_l^T \rangle_i$$

where l denotes the lth observation, q_l^i denotes $Q_l = i$, and posteriors on each discrete

configuration are denoted as

$$w_l^i = P(Q = i \mid e_l).$$

In the following, x denotes a value over the joint space of the set of y's continuous parents. Note that it may be the case that y is observed or hidden, and parts or all of x may be observed or hidden. If any variables are hidden the "value" of the variable is known by its mean μ_x and covariance Σ_x as computed in marginalization. If a variable is hidden the conditional second moment for e_l is updated by

$$\langle x_l x_l^T \rangle' = \langle x_l x_l^T \rangle + \mu_x \mu_x^T + \Sigma_x$$

while if it is observed to be x^* we use

$$\langle x_l x_l^T \rangle' = \langle x_l x_l^T \rangle + x^* (x^*)^T$$

with analogous equations for $\langle y_l y_l^T \rangle$ and $\langle x_l y_l^T \rangle$. If only parts of x are observed, we treat the observed parts as having covariance zero.

For the parameter update equations we consider the most general case, where the variable y has both continuous and discrete parents, and has a full covariance matrix Σ. For simpler situations the update equations simplify.

The new regression matrix W_i and μ_i are solved for simultaneously with

$$B_i = \begin{pmatrix} W_i & \mu_i \end{pmatrix}$$

by adding a 1 as the last component of x. For the updated value of B_i we have

$$B_i = \left(\sum_l w_l^i \langle y_l x_l^T \rangle_i \right) \left(\sum_l w_l^i \langle x_l x_l^T \rangle_i \right)^{-1} \tag{5.8}$$

and

$$\Sigma_i = \frac{\sum_l w_l^i \langle y_l y_l^T \rangle_i}{\sum_l w_l^i} - B_i \frac{\sum_l w_l^i \langle x_l y_l^T \rangle_i}{\sum_l w_l^i}. \tag{5.9}$$

where we use the updated B_i. These update equations are derived in the usual manner by setting the derivative of the likelihood function to zero and solving for the parameters of interest. The probability tables of discrete nodes may be updated in the usual way by averaging over the w_l^i.

5.4 A Bestiary of Bayesian Networks

By way of example we show that hybrid Bayes networks subsume some common interesting probabilistic structures. In the figures presented in this section, we will use the graphical convention of depicting discrete nodes with squares and continuous (Gaussian) nodes with circles.

Figure 5-7: A mixture of Gaussians as a hybrid Bayesian network and its (trivial) clique tree.

5.4.1 Mixture of Gaussians

The mixture of gaussians is the fundamental representation underlying hybrid Bayes nets. Figure 5-7 shows a mixture of Gaussians in its graphical form, and its associated (trivial) clique tree.

The associated joint probability is

$$P(x, q) = P(x \mid q)P(q)$$

where q indexes over the components of the mixture. The likelihood function is

$$P(x \mid \lambda) = \sum_j P(x \mid q = j)P(q = j)$$

The posterior probability $P(q = j \mid x)$ may be computed by marginalizing.

It is important to note that in the usual application of mixture of Gaussians, the continuous variable is observed. Thus the machinery of the hybrid Bayesian network is unnecessary. However, this structure will appear in other larger structures, possibly with hidden continuous variables.

5.4.2 HMM

An HMM may be thought of as a mixture model with dynamics. An HMM as a Bayes network and its clique tree is shown in Figure 5-8.

The usual output probability distribution function $b_j(x_t) = P(x_t \mid q_t = j)$ has its exact analogue in the network, as does the transition matrix $A_{i,j} = P(q_t \mid q_{t-1})$ and initial state distribution $\pi_j = P(q_0 = j)$. Note that in the standard formulation of HMMs, the distributions themselves do not depend on time. To reflect this in a Bayes net, we may simply constrain the conditional probability distributions for each times step to be the same. This practice is called *parameter tying.*

Two-phase propagation algorithm for Bayes networks is exactly equivalent to the forward-backward algorithm used for parsing[65]. The forward-backward algorithm gives the posterior probabilities $\gamma_{tj} = P(q_t = j \mid \lambda, x_0, ...x_T)$. These same posteriors may be obtained in the Bayesian network formulation by entering evidence at each observation node, running the inference algorithm, and marginalizing to obtain the distribution for each q_t.

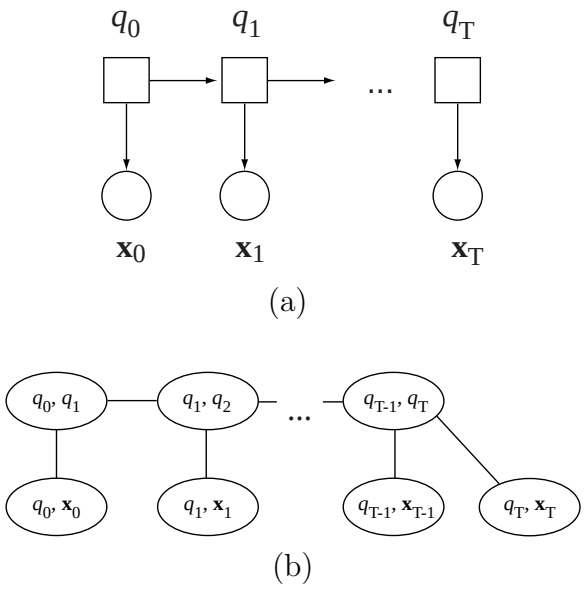

Figure 5-8: A hidden Markov model as a hybrid Bayesian network and its clique tree.

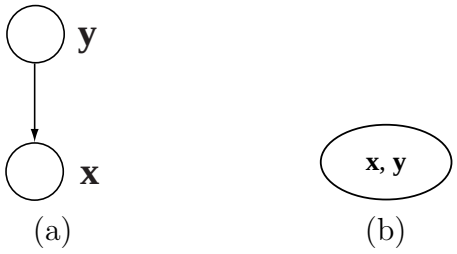

Figure 5-9: A factor analyzer as a hybrid Bayesian network and its (trivial) clique tree.

As in the case of mixture models, most often all of the continuous nodes in the case of HMMs are observed.

5.4.3 Factor Analysis

In a factor analysis setting, we have the network shown in Figure 5-9. The matrix W is called the *loading matrix*. The loading matrix may be set to simply pick out components of the parent variable, but more commonly in factor analysis problems the loading matrix is used to perform dimensionality reduction, in which case the values of W are learned from data. For example, W may be trained to accomplish a dimensionality reduction similar to that of principal components analysis (PCA).

5.4.4 Kalman Filter

In the Bayes network representation of the Kalman filter, shown in Figure 5-10, the matrix W takes the role of the transition matrix A. The covariance of the state variables are exactly the matrix Q, the linear transform that maps x into y is exactly

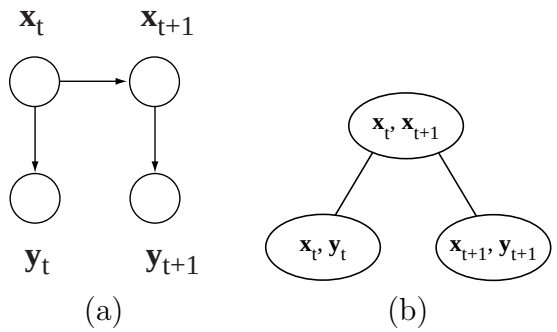

Figure 5-10:

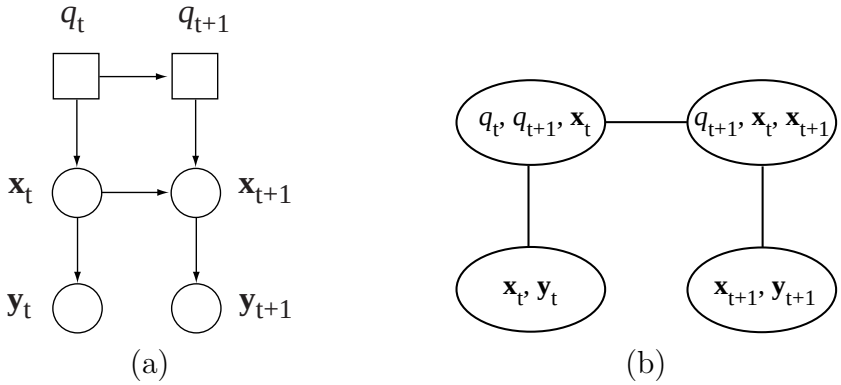

Figure 5-11: Switching Kalman filter as a hybrid Bayesian network, and its associated clique tree.

the observation matrix H, and the covariance of y is the observation noise covariance R.

Kalman smoothing may be achieved by entering evidence y_t and y_{t+1}, propagating, and computing the posterior of x_{t+1} by marginalizing. In a later section we will show how the notion of dynamic Bayesian networks addresses the application of this model to a sequence of observations over time and the necessary propagation of the state distribution over time.

The switching Kalman filter (Figure 5-11) may be thought of as a hybrid of the Kalman filter and the HMM.

5.4.5 Others

In addition to the networks presented here, it has also been shown that hybrid Bayes networks may be be used to implement mixture of experts, factorial HMMs, coupled HMMs, autoregressive HMMs and more.

However, the attraction of the hybrid Bayesian network framework is not its power to subsume existing probabilistic models but instead that it allows the construction of various hybrid structures without deriving inference and learning algorithms for each case. For example, the switching Kalman filter is an interesting cross between an HMM and a Kalman filter that is relatively easy to put together as a hybrid

Bayes net. In fact, one view of the compilation of the Bayes network into a junction tree is that of the automatic construction of inference and learning algorithms for a given network topology. If matched by an appropriate software implementation this flexibility is a boon to the ability to experiment with novel probabilistic models.

5.5 PHMM as Hybrid Bayesian Network

Figure 2-2 of Chapter 2, which is repeated here in Figure 5-12, hinted that the PHMM may be cast as a Bayesian network. With the above exposition of hybrid Bayesian networks, we are now prepared to investigate this possibility in full.

The isomorphism between the parameters of the PHMM and that afforded by the hybrid Bayesian network is complete. The implementation of the HMM as a Bayes net has already been highlighted. The added twist of the PHMM, the dependence of the means of the output probability distributions on a linear transformation W_j matches that of the hybrid Bayes net model for continuous parents. Note that the nonlinear PHMM is not addressed by the hybrid Bayesian network framework, as nonlinear regression functions can not be incorporated by equations 5.3 and 5.4.

The nonlinear PHMM aside, there are a number of advantages to casting the PHMM as a Bayesian network. First of all, as a Bayesian network, we may obtain a posterior distribution over the PHMM's variable $\boldsymbol{\theta}$ rather than a point estimate of $\boldsymbol{\theta}$ derived previously. Secondly, the testing phase of the PHMM no longer would require an iterative (EM) process to recover the value of $\boldsymbol{\theta}$. The junction tree inference algorithm will allow the calculation the posterior distribution of $\boldsymbol{\theta}$ without an iterative algorithm. Lastly, as a Bayesian network, the PHMM may be incorporated into other Bayesian networks. Similarly, it is an easy matter to consider variants and hybrids of the PHMM. For example, Figure 5-13 shows a variant of the PHMM which models the evolution of $\boldsymbol{\theta}$ as a process of its own.

The computational complexity of the inference on the junction tree is proportional to the number of variables in the largest clique in the junction tree. In the case of PHMMs, HMMs and other models that span T time steps, we would like to have the size of the largest clique be constant with respect to T. The usual junction tree for a PHMM is shown in Figure 5-14. As with the usual HMM formulation, there are no cliques that scale with T in size.

Figure 5-14 shows the tree produced by the weak junction tree algorithm, which as described in Section 5.3.3 has the undesirable property that global consistency over the variable distributions is no longer guaranteed after propagation. The strong junction tree for the same PHMM is shown in Figure 5-15. Unfortunately, there is a clique in the strong junction tree with size proportional to T. Note that even if the state output distributions are a modeled as a single Gaussian, the distribution over $\boldsymbol{\theta}$ is represented in the junction tree as a (large) mixture of Gaussians. At the root clique, the size of this mixture is exponential in the length of the sequence T. Thus the PHMM is intractable if we use the strong junction tree algorithm.

So what are the PHMM training and testing algorithms presented in Chapter 2 actually doing, in terms of Bayesian networks? In fact, from the Bayes network

point of view the PHMM has exactly the same network topology as the usual HMM. The PHMM can be thought of as an HMM which employs the usual HMM training algorithms (EM) during runtime to recover the parameter $\boldsymbol{\theta}$; the computation of $\boldsymbol{\theta}$ is no different than, say, the training of the Σ_j during the training phase, except $\boldsymbol{\theta}$ is tied across all time steps. The PHMM thus trades the intractability of the exact junction tree of Figure 5-14 for the iterative runtime testing procedure. Given that in the experiments presented in Chapter 2 only a few iterations are required to recover $\boldsymbol{\theta}$, this may be a advantageous tradeoff.

The iterative testing procedure of the PHMM suggests a way to exploit the Bayes network propagation algorithm to obtain a distribution over $\boldsymbol{\theta}$ rather than a point estimate. The first phase of PHMM testing involves running the forward backward algorithm on an HMM with a fixed value of $\boldsymbol{\theta}$. This may be recast as propagation on the network in Figure 5-16, where the same value of $\boldsymbol{\theta}$ is entered everywhere. Note that this network, like the usual HMM, is tractable under the strong junction tree. The second phase of PHMM testing involves updating the value of $\boldsymbol{\theta}$ given the current parse of the input. With the network in Figure 5-16, the *distribution* over $\boldsymbol{\theta}$ may be updated by running the M-step of the EM algorithm, where the E step is computed as before by propagation on the network, and the parameters of the distribution on $\boldsymbol{\theta}$ are tied across all states. The Bayesian network analogue of the PHMM is complete.

This suggests a general mechanism for coping with intractability in Bayesian networks: make (possibly multiple) tractable versions of the original network by removing arcs, replicating nodes and tying parameters, and combine the results in an iterative EM process.

In summary, there are a number of options in treating a PHMM as a Bayes net:

- Run the usual inference algorithm using the weak junction tree, and hope that the missing guarantee of consistency won't affect performance in the real world.
- Run the inference algorithm on the strong junction tree, and hope that the high computational complexity of the resulting inference process won't be a problem.
- Use the PHMM formulation of Chapter 2, running multiple iterations of the usual inference algorithm during testing.
- Exploit a variational approximation to the full Bayesian network, as in [38].

The first and third options are likely to be the most practical. How do these two approximations compare? We note that with the PHMM-based formulation, it is possible to obtain an estimate of $\boldsymbol{\theta}$ that is arbitrarily close to the true value of $\boldsymbol{\theta}$ by running sufficiently many EM iterations, assuming that EM does not run afoul of local minima. The Bayesian network PHMM formulation based on the weak junction tree does not guarantee a good global estimate of $\boldsymbol{\theta}$ since as the propagation procedure proceeds down the clique tree, the value of $\boldsymbol{\theta}$ is updated via (strong) marginals which are approximations to the true marginal. For a sufficiently long chain, this approximation will affect the ability of the propagation algorithm to arrive at a globally best estimate of $\boldsymbol{\theta}$.

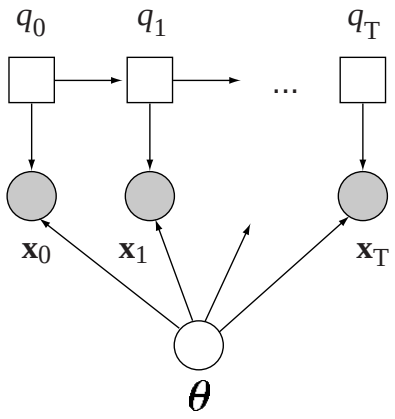

Figure 5-12: PHMM as hybrid Bayesian network.

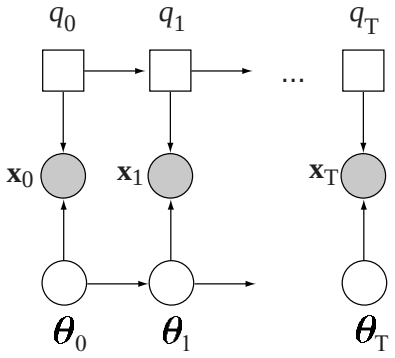

Figure 5-13: PHMM as hybrid Bayesian network, with recursive estimation of $\boldsymbol{\theta}$.

However, in many applications the weak junction tree-based formulation will perform adequately. In situations where only a portion of the signal needs be examined to arrive at a good global estimate of $\boldsymbol{\theta}$, the weak junction tree formulation may find it in a more computationally efficient manner than the PHMM-based formulation.

5.6 Watch and Learn as Hybrid Bayesian Network

Recall that in the Watch and Learn system presented in Chapter 4, the information relating to external application context was incorporated into the HMM by manipulating the output probability distributions directly before running the inference

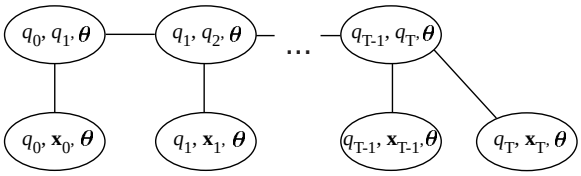

Figure 5-14: PHMM junction tree.

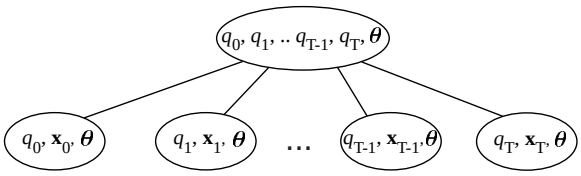

Figure 5-15: PHMM strong junction tree.

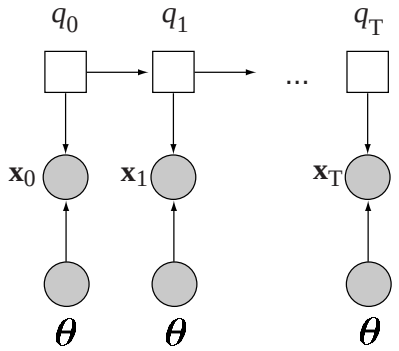

Figure 5-16: Tractable network used in PHMM testing, with parameters on $\boldsymbol{\theta}$ tied across all time steps.

algorithm.

With the hybrid Bayesian network framework, we are prepared to provide a more probabilistic approach to driving the parsing algorithm by application context. Figure 5-17 shows two possible Bayesian networks that permit combining application context and Markov models of behavior. In Figure 5-17a, the distribution q_t is made to depend directly on the context signal, as well as q_{t-1}, while in Figure 5-17b the context impacts the observation output distributions directly. In either case, if we always have context information at each time step, the network may be able to do without the arcs from each c_t to c_{t+1}. The network in Figure 5-17b is also known as a *factorial HMM*[26].

In the next Chapter we will further explore the idea of combining context with Markov models of behavior via Bayesian networks.

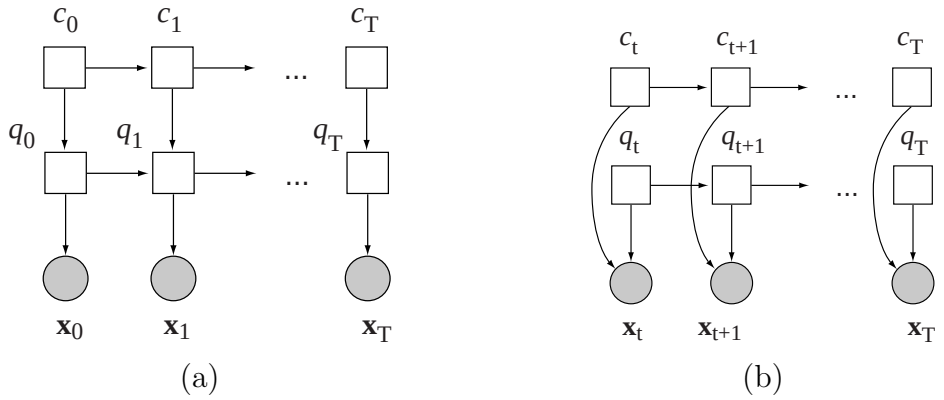

Figure 5-17: Watch and Learn as two different hybrid Bayes networks.

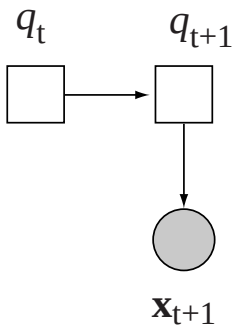

Figure 5-18: A single time slice of an HMM.

5.7 Dynamic Hybrid Bayesian Networks

Thus far in the presentation various Bayesian networks that are applied to sequences over time, the reality of the application of these temporal models on very long sequences has been ignored. In fact, the models assume that the start and end of the sequence is known, an unreasonable assumption for applications without a preprocessing phase that includes segmentation.

Dynamic Bayesian networks, as they relate to the work to be presented in the next chapter, address this issue by considering one time slice of the Bayes network and how to propagate probabilities from one time slice to the next. For example, Figure 5-18 shows a single time slice of an HMM. This model specifies the dependencies between variables at time $t-1$ and the variables at time t (*inter*-time slice dependencies), as well as the dependencies among the variables at time t (*intra*-time slice dependencies). This network is minimal in terms of its ability to describe the evolution of a process over time, and serves as a compact representation of the network in Figure 5-8. Note that a network of one time slice may be replicated or "unrolled" so that it spans multiple time steps.

5.7.1 Propagation Forward in Time

The difficulty in using a single time slice model is in how to propagate the posterior probability distributions in the current time slice to be prior distributions for the next time slice. In general, referring to the diagram in Figure 5-19, the solution is to compute the posterior distribution on q_t by marginalization and then set the distribution on q_{t-1} for the next time slice to match. The next time slice is then ready to accept evidence at the observation node and have the inference algorithm run. This process repeats at each time step.

If there are multiple variables that propagate across time slices, the correct approach is to calculate the posterior of the joint distribution of all propagated variables, and then set each variable in the next time slice accordingly. Unfortunately, the marginal over the joint space may be intractable for more than only a few variables. One active line of research thus involves finding approximations to this propagation. In Chapter 6, however, we will ignore this problem for the time being, by computing

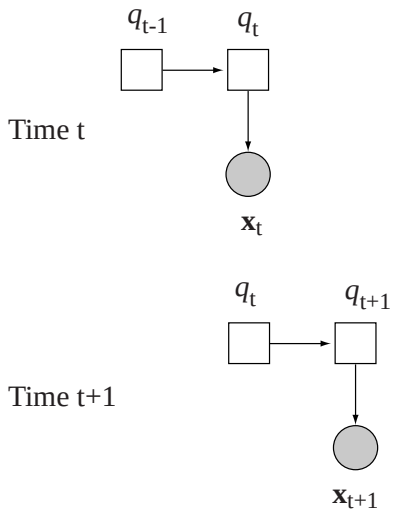

Figure 5-19: Propagation across time slices.

the marginal over the joint space only when it is convenient to do so; that is, when the multiple variables are in the same clique, and otherwise propagating distributions singly.

Note that the single time slice may be unrolled a number of times and the same propagation scheme may be applied. We may wish to unroll a single time slice model if the network requires a longer time scale view in order to arrive at correct posterior distributions on the Markov state. A large time window will allow the HMM to be less sensitive to spurious input, as well as find a more globally optimal parse.

In preparation for the framework developed in Chapter 6, we would like to extend this notion of propagation over time to address the situation in which there are multiple samples of each observation variable for a single time step. This could be useful in the situation where the network is compactly defined with a single instance of a random variable but during runtime we would like to incorporate the combined information of multiple samples as evidence. In this case, the marginal that is computed and entered into the next times slice should be an average marginal or *aggregate* marginal that is computed from the ensemble of examples.

The same machinery developed for parameter estimation (section 5.3.4) may be used to compute the aggregate marginal. The idea is to "train" the variable in the next time slice by the computing statistics on the distributions of the variable or group of variables in the current time slice. After all evidence has been examined, the statistics collected on the variable in the current time slice are used to determine the parameters of the variable or group of variables in the next time slice. Equations 5.8 and 5.9 apply.

5.8 Real-Time Implementation

There are two common approaches to the implementation of hybrid Bayesian networks. One approach is to decide on a network topology and work out the inference and learning equations on paper. These tailor-made equations may be implemented in computer software, and since they are tailored to situation they may be optimized for the task at hand.

Though this approach exploits the theoretical machinery of Bayesian networks, it has the practical difficulty that if one wishes to change the topology, the equations must be changed as well. If the topology is known from the start however, this is the easier of the two approaches.

The second option is to code the junction tree algorithm directly into software in the most general way possible, so that designer need only specify the network topology and the type and dimension of each node; the software compiles this graph into a junction tree ready for inference and learning.

The difficulty here is twofold. First, building a software library to support constructing hybrid Bayes networks and their accompanying inference and learning algorithms to handle general networks is quite involved. Secondly, it may difficult to achieve the levels of performance necessary for experiments that run in real-time using such a generalized framework. In particular, it may be difficult to exploit the optimizations that were readily available in the first approach.

My view has been that Bayesian networks free the designer of probabilistic models from the burdensome and error-prone process of manipulating likelihood equations, and thus encourage experimentation with novel topologies. And after studying a few probabilistic models, its not long before one realizes the regularity of these systems anyway. Why not capitalize on this regularity?

With this in mind I implemented the framework described in this chapter in Java, with an eye towards realtime implementation. With today's adaptive compilers for Java this has proved to be an attainable goal. The performance of implementation, as measured by how many inference operations may be performed in time, has benefitted from a number of key optimizations, two of which I describe here.

First, the implementation caches computations aggressively, allocating a reusable object for each marginalization, multiplication and division operation performed during the inference and learning processes. The assumption is made that upon successive calls of the inference algorithm the set of observed variables is not likely to change. This caching tends to regain some of the performance increase that would be otherwise be obtained by the hand-coded approach, and has also been applied in the learning process.

Secondly, the following further approximation to the weak marginalization process was implemented. Consider the fact that the canonical characteristic K usually does not depend on any evidence entered on the network. In fact, the only time in which K gains a dependence on the evidence is in the weak marginalization process, through μ in equation 5.6. Any successive propagations that in turn depend on this marginalization will similarly depend on the same piece of evidence. If this dependence on the evidence were removed, it would be possible to precompute or cache K^{-1}. In the

present implementation, this optimization is made possible by instead of using equations 5.5, 5.6, 5.7, the most probable mode i is optionally chosen. It follows then that at any point in the inference process, the value of K at any potential is drawn from a finite set of such K, the particular K chosen depends on which modes are selected in any previous weak marginalization operations that current marginalization depends on. Thus subsequent calculations involving K^{-1} will be able to exploit its cached value, and it may be the case that during the inference process no matrix inversions are necessary.

5.9 Conclusion

This chapter introduces the basics of Bayesian networks and presents the particular Bayesian network framework of hybrid Bayes networks. The mathematical frameworks of work presented in previous chapters were considered in the context of hybrid Bayes networks. Furthermore this chapter lays the foundation for the work presented in the next chapter, which continues development of context-adaptive models for gesture recognition and exploits the realtime implementation of Bayes nets described here.

Chapter 6

Bayesian Networks for Online Adaptive Gesture Recognition

6.1 Introduction

In the preceding chapters, frameworks have been presented that demonstrate the adaptation of some part of a recognition model in response to the context of the situation. The PHMM adapts a parameter of global deformation to match a gesture family model to an instance of the family. In this case, the adaptive parameter indicates how the gesture is conducted, or where in the gesture family the gesture belongs. The PHMM exploits invariants in how the deformation parameter impacts the state output distributions over the entire span of the gesture.

The natural gesture video parsing system of Chapter 3 exploits a temporal model informed by theories of natural gesture. The adaptive parameters that are recovered by the system includes the appearance models of the various rest states that the speaker uses throughout the video. These are then exploited to subsequently parse the entire video into separate gestures.

The Watch and Learn system of Chapter 4 builds on the natural gesture parsing system by incorporating a time-varying application context signal that guides the HMM parse while appearance models are learned. These recovered appearance models allow open-loop interpretation of the user's appearance, and are analogous to the rest-state appearance models recovered by the natural gesture parsing system.

An interesting property of these systems is that in each case, the adaptive part of the model is unknown upon start and is recovered after the examination of data collected from the particular situation at hand. Furthermore, if the adaptive part is at some point known, the resulting recognition problem is fairly easy, or at least amenable to traditional recognition techniques. Given this observation, it is fair to think of the adaptive parameters as "hidden" variables, as they are known in the application of the expectation-maximization (EM) algorithm.

Lastly, in each of these systems, constraints must be exploited to guide the adaptation process. These constraints may be taken from the domain, as with the Watch and Learn system and the natural gesture parsing system, or learned from previously

seen data, as with the PHMM.

In this chapter we continue the development of the adaptive model approach applied to the problem of online learning of gesture, with the following variations:

- We exploit the hybrid Bayesian network framework presented in the previous chapter. Hence we are concerned more with network topologies than with particular parsing and recognition algorithms, or how these particular algorithms must be extended and modified.
- More than one component of the system is adapted. This makes for interesting learning scenarios, including a form of learning transfer.
- The whole-image appearance-based approach of Watch and Learn is exchanged for a object-based approach that allows the system to ignore irrelevant changes in the image, permits tracking of multiple objects, and allows gesture models that are based on movement rather than similarity to learned images. Based on movement, these gesture models are thus more likely to be reusable in situations where the appearance has changed.
- A simple reinforcement learning approach is used to learn the mapping from user actions to system actions.

6.2 Related Work

Increasingly, sophisticated probabilistic techniques are being applied to the problem of human motion analysis (see [25] for a survey). Probabilistic techniques are popular in the domain of human motion because they tend to be more robust and so are more suited to difficult tracking scenarios. Also, they provide a firm mathematical foundation upon which to integrate multiple disparate sources of information, including movement models and appearance models, while simultaneously being amenable to a analysis.

[1] embeds the three-dimensional, model based tracking of human head and hands within a recursive framework based on Kalman filters, and later on dynamics[82]. The latter work can incorporate top-down information about categorical movement in the form of innovations in a control process. Their system begins with a *a priori* mixture of Gaussians model of flesh color and requires initialization based on an assumed pose of the user, but otherwise is similar in the present work in the broad sense that it exploits movement models in a top down fashion, as well as employs Kalman filter-based tracking frame to frame.

The "condensation" algorithm [6, 29] exploits particle filtering. It has been demonstrated on a variety of tracking tasks. The condensation algorithm is attractive for its simplicity: it requires only a likelihood function which evaluates a point in model parameter space, and a propagation method which updates each point to the next iteration. The present work also exploits sampling, but differs from condensation in that at each frame, the parameters of a probabilistic model are changed to reflect the

data, where in the condensation algorithm, no model is reconciled with the samples: the collection of samples implicitly represent the updated distribution. The advantage of the condensation approach is thus that any form of the likelihood function can be used, since ultimately its parameters will not have to be tuned.

The drawback of the condensation algorithm is that many samples must be collected to achieve good tracking results. Furthermore, when higher-level models are added to the likelihood function, even more samples are required. For example, [30] augments the algorithm with switching movement models, and in [5], the condensation algorithm was applied with an HMM incorporated into the likelihood function. To propagate a particle to the next time step, the transition matrix of the HMM is sampled, along with object contour parameters. In the present work, we update parametric models of distributions from multiple samples. Furthermore, the present work splits the model in high-level and low level pieces so that sampling may be handled differently in each.

[11] uses the composition of various representations to recognize human movement using learned representations. The hierarchy consists of blob tracking, Kalman-filter smoothing, and hidden Markov models driven by EM, Kalman filter equations and dynamic programming. In one view of the present work, a similar approach is taken with a variety of representations, but here they are united in the mathematical framework of Bayesian networks. We also explore the notion of online learning.

The application of dynamic hybrid Bayesian networks directly to the analysis of motion is explored in [56], which adopts a switching (mixture) Kalman filter approach. The graphical model is the same as that of Figure 5-11. [58] explores the application of linear dynamical systems and HMMs for the recognition of driving behavior. Here we also employ a combination of HMMs and linear dynamical models.

A dynamic Bayesian network is applied to various image processing tasks in [37, 36]. Their framework is similar to the present framework in that a latent transformation variable is searched to find the best fit to the input.

There has also been work on using multiple high-level sources of information to drive object discovery processes. For example [48] exploits known action models, object classes and context in a hand gesture recognition task, while in [32] multiple moving objects output from a noisy tracking process are rejected based on their movement history and several categorical movement scenarios.

The present work is concerned with the real time implementation of Bayesian networks. Of the work mentioned above, only [6] and [1, 82] demonstrate real time implementations.

Lastly, the present work will utilize a simple reinforcement learning technique to train the mapping from gestures to actions. Reinforcement learning techniques are presented in [67], while a related probabilistic approach to reinforcement learning based on EM is explored in [31].

6.3 Model

6.3.1 Images versus Objects

Watch and Learn infers appearance models given a strong model of the temporal structure of the behavior combined with time-varying application context information. The system is interesting because it works exactly backwards according to most gesture recognition paradigms. Most gesture recognition systems proceed by acquiring an object of known or assumed appearance by a tracking algorithm, and its position over time is examined to determine the class of categorical motion that object underwent, if any. Watch and Learn proceeds in the opposite direction: it begins with movement model and infers an appearance model that is consistent with the movement model and current input. The advantage of this backwards approach is that no appearance model need be assumed at the outset, and so the system will work in a great variety of situations. Furthermore, since the input is the whole image, tracking is not an issue. Watch and Learn relies on an invariant that is much more powerful than our appearance: our behavior in context.

The main advantage of the image-based approach of the Watch and Learn system is that it is able to correlate any changes in the image stream with application context. Thus it is possible to have the system pick up on almost any movement that fits the behavior model.

The biggest drawback of the approach is that because the learned images are so tuned to exact circumstances (context) these learned representations will not be useful in any other session. For example, if the pose of the user is somewhat different in the next learning session, Watch and Learn will fail. Similarly, if the user, say, trains a hand gesture and then changes the stance of his feet, there is a chance that these will make the incoming images sufficiently different from the learned images that its online adaptation will incorrectly adapt to the new stance. Watch and Learn lacks a focus of attention.

A representation based on tracked objects rather than whole images gives the system a focus of attention by virtue of the segmentation implied by a tracked object. Tracking objects also has the consequence that, since a tracked object has a necessarily local view of the image, the tracked object may be lost by the system. A multiple hypothesis scheme is one way to have object representations be able to commit to local regions while obtaining robustness in tracking.

Tracking objects will allow us to build representations of the movement of the object. In Watch and Learn, a representation of the user's movement was stored implicitly and by the collections of learned images, so was lost when the session ended. The present work models the velocity of the object over time. Since a representation based on simple movement statistics will be approximately independent of the particular appearance of that object, these movement models have a higher chance of being reusable.

6.3.2 Categorical Motion

In the work presented in this chapter we use a blob-based model of objects. We begin with the assumption that there are distinct objects of interest that we would like to track, but, in the spirit of Watch and Learn, without necessarily assuming a particular appearance model of the objects.

If an appearance model of the object is unavailable, we rely on the assumption that an object of interest is likely to exhibit an interesting, appropriate motion. Thus if we see the object undergo such a motion, we are more likely to believe that it is an object of interest. By "appropriate" we mean that it moves in a *familiar* way or that it moves in a way that is consistent with *domain-specific context*.

Furthermore, we would like to have the system learn an appearance model of the object, so that the object may subsequently be tracked without relying on the movement pattern of the object.

The ability to "spot" an object of interest based on its pattern of movement is loosely inspired by the informal observation that mammals are able to pick out interesting objects from a complex visual scene in a process seemingly based in a large part by movement alone. For example, we are usually able to spot someone waving to us from across the street, even when there is plenty of extraneous movement such as the passing of cars and pedestrians.

6.3.3 Multiple Hypothesis Tracking

Modeling the scene as a series of blobs, it impossible to refer to a blob without also referring to a location, and implicitly, its appearance in the image. The appearance of objects throughout the image does not change smoothly as you consider different locations: the space of appearance and locations is lumpy. This means that we will not be able to use the simple EM-based approach of Watch and Learn to smoothly adapt to object appearance models, since, fundamentally, there are multiple independent objects in the scene.

If we are unsure which object (blob) in the scene is the true object of interest, one solution is to track each of them until one or more of them exhibits an interesting categorical motion. Unfortunately, any given scene is likely to have many, many objects. In fact, any scene will have an arbitrarily large number of blob objects in it if the complexity of the appearance model is uncontrolled.

Here we assume a very simple appearance model based on mixture model of color, with very few number of mixture components. Furthermore, we will avoid tracking every object in the scene and instead prefer initially to follow only objects that exhibit image-based motion based on image differencing.

6.3.4 A Taxonomy of Movement

In the object discovery process described above, three different notions of movement are exploited, each exerting constraints of varying strength. At the lowest level, coarse image motion found by image differencing indicates likely areas of the image that

include the target object. Image motion is a fairly weak, frame-to-frame constraint, as there are likely to be many regions of the image that exhibit such motion.

The next kind of movement is the ballistic, frame-to-frame tracking of an object, and is considered a mid-level representation. This category of motion is stronger than image-based motion, since it adds the constraint that the new position of the tracked object must match an appearance model, as well as the constraint the object's velocity will tend to be constant.

The highest and strongest representation of movement is that of long time scale categorical movements, including whole gestures. Very few movements under consideration will be good examples of categorical motion; hence they are the strongest cue for the target object when no appearance model is available.

These three models of movement will be encoded directly into our representation and algorithm for interpreting object movement. The low level motion representation will be used to seed new object hypotheses, the mid-level representation will be used to follow existing object hypotheses, and the highest level will be used to prefer one object hypothesis over another.

6.3.5 Domain Context

In the Watch and Learn system, domain knowledge is exploited in two ways: first, the Markov model for gesture is known *a priori*. In the case of the musical conducting example, the three state REST, UP and DOWN beat model is hand-coded. This is an acceptable part to hand code, since part of the motivation of Watch and Learn is the assumption that this coarse Markov model is invariant to a variety of other circumstances. Second, the Watch and Learn system correlated this Markov model with a time-varying context signal which ensured the semantic consistency of the learned states: namely, that the UP state corresponds to an "upbeat" pose, and the DOWN state corresponds to a "downbeat" pose. Note that if there weren't the need to drive the output actions based on state membership, the context signal would be unnecessary.

The present system expands on the notion of domain context in two ways. First, as discussed above, domain specific categorical movements are exploited in the object discovery process, and ultimately in the learning of appearance models.

Secondly, instead of only considering how the time-varying context impacts which *phase* or HMM state of the gesture the system is in, this information will also impact which gesture the system is in.

6.3.6 Gesture to Action Mapping

In Watch and Learn, the context events used to drive the learning were then also generated by the system once the learning had proceeded sufficiently. In the present system, we consider a more general representation that encodes context and output actions separately. The distinct representation of the gesture to action mapping permits more flexibility in how domain context impacts the interpretation, and also permits the separate learning of the gesture to action mapping.

In the present system the gesture to action mapping may be trained in an online fashion by a simple reinforcement learning mechanism that is used to learn the mapping from a known gesture to an action primitive. This process incorporates the history of the actions and gestures performed, and positive and negative feedback given regarding each action taken by the system. Note that the feedback on actions may also impact our belief about which gesture just occurred.

6.3.7 Multiple Symmetric Learned Representations

Suppose a tracked object undergoes a familiar movement, thus increasing our belief that the object is of interest. Abstractly, the learning of the appearance model is driven by the movement model. Since we can now be confident that we have the right object, it is reasonable to follow it regardless of the motion it undergoes. We can continue in this traditional feed-forward style of computation indefinitely, or we may take the opportunity to learn *new* movement models of the now familiar object, so that the next time an object must be acquired, a larger variety of movements may be picked up, improving the odds that the object may be acquired. Thus the appearance model, used for tracking, drives the learning of movement models.

Note that in the first object acquisition phase, learning appearance models is driven by movement models, while later, the appearance models are used to infer movement models. This back and forth style of learning goes beyond that of the other systems presented in this thesis, each of which learn in primarily one direction during runtime.

The appearance model and movement model are therefore symmetric in terms of how the learning proceeds. The algorithm may start with appearance and infer movement, or it may start with movement and infer appearance. The very first movement models must be hand-coded, and will serve to bootstrap the learning of new models once an object is acquired. As more gestures are learned the less necessary the hand-coded gesture will become. Eventually, it may be possible to discard the hand-coded models in favor of more accurate learned models.

This notion of symmetric learned representations also includes domain context and action. For example, at a given moment, domain context may be sufficiently constrained, driving the system to expect a particular gesture. This expectation ripples throughout the system, driving the tracking and ultimately the acquisition of an appearance model.

At any given point in the life of the system, *some* representations must be known by the system. Object tracking will fail, for example, if both the appearance model and the movement model is unknown. Similarly, if the domain context is uncertain, learning the semantic labelling of the gesture will have to wait on the user's feedback. Likewise, the present system does not support the simultaneous learning of the size of the object and object appearance model.

The ability to start the adaptation with either one known representation or another and adapt the remaining unknown representations is the chief mechanism by which the system is able to adapt to novel situations.

6.3.8 Online Learning of Gesture

Novel gesture models may be acquired in two ways: first, by simply training new gesture models for movements that are unfamiliar, and second, by training new gestures in response to changes in context.

The system is able acquire new gesture models online in an unsupervised way by simply initiating a new gesture model when an unfamiliar movement is seen. This gesture may then be given appropriate semantics by the reinforcement learning procedure. This, of course, requires feedback on the part of the user, and may require the user to provide several examples paired with appropriate feedback.

The second style of acquiring new gestures involves adapting a new gesture model in the same way as in the unsupervised case, except that the context guides *which* gesture model is learned. This may be conducted in such a way that the gesture to action mapping is determined appropriately in advance, such that the reinforcement learning-based procedure is unnecessary. For example, the system may have a gesture model that is labelled THROW, but is otherwise untrained. The system may also know *a priori* that certain application events are likely to be accompanied by a THROW gesture. Thus if a unknown movement is seen during these events, the THROW gesture model will be trained, and not some other untrained gesture model. Furthermore, the action associated with the THROW gesture may be known *a priori*, in which case feedback to train the action mapping is unnecessary; the context information has done all the work necessary.

6.4 Bayesian Network Implementation

6.4.1 Network Topology

A hybrid Bayesian network designed to support the various inference and learning processes described above is shown in Figure 6-1. Here, we consider the tracking of a single moving object in the image. The various components of the network model include

- a mixture of Gaussians which models the object's appearance as a distribution of pixel color values
- a mixture of hidden Markov models which models the velocity of the object over time
- a Kalman filter-like integration of instantaneous object velocity to track the absolute position of the object from frame to frame.
- random variables for output actions and domain context, each of which depend on the current gesture
- a Markov model that models the evolution of this object as a hypothesis, from seeding to removal

The following details each of the components of this Bayesian network in turn.

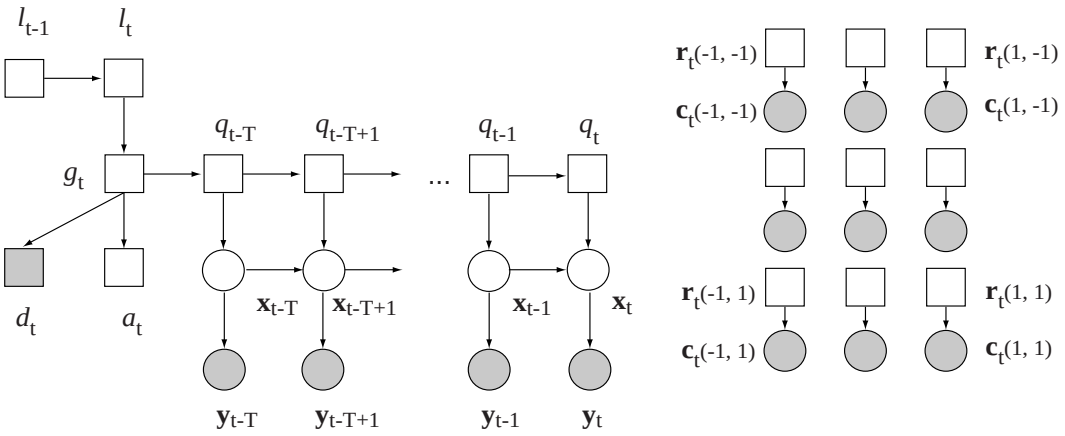

Figure 6-1: The full Bayesian network which incorporates movement, appearance, context, action and control models. The variables are described in the text and in table 6.1. Shaded nodes indicate variables that are typically observed.

l_t	control state variable
g_t	current gesture
q_t	HMM state
x_t	absolute position of tracked object or pixel
y_t	absolute position with additive noise
c_t	YUV color of pixel
r_t	color mixture component variable
d_t	domain context
a_t	action
T	total number of timesteps modeled by HMM

Table 6.1: The meaning of each variable in the network of Figure 6-1.

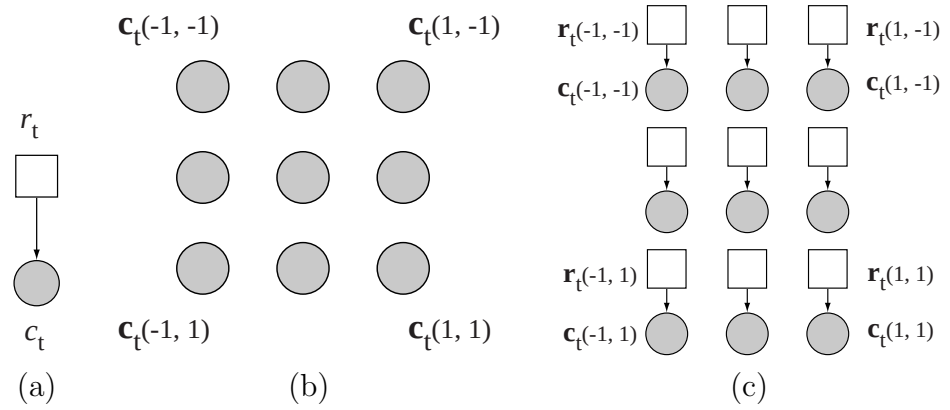

Figure 6-2: Appearance models. (a) The simplest appearance model, a mixture model on pixel color. In (b) and (c) pixel colors $c_t(dx, dy) = I(x + dx, y + dy)$ are drawn about an origin (x, y). (b) an image-based approach, one node for each pixel in a 3×3 image. (c) color at each pixel is modeled by a mixture of Gaussians, each with mixing coefficients $r_(dx, dy)$. If color mixture parameters are tied across all nodes, the net result is a color histogram matching over all pixels.

6.4.2 Appearance Model

Pixels are drawn from an image I_t, each with position $x_t = (x_t, y_t)$ and color $c_t = I_t(x_t, y_t) = (Y, U, V)$. The appearance model models the distribution of colors c_t drawn from the pixels belonging to the tracked object. Presently, a mixture of Gaussians on the value c_t is employed, with the multinomial r_t indicating from which of $|r_t|$ mixture components c_t is drawn.

Note that while this distribution only models the color of *single* pixel, in general more than one pixel from the image will need to be examined to track an object. Later we will consider how to handle multiple samples from the image, which collectively will incorporate color information from multiple samples from the image of the tracked object.

This model assumes that the distribution of color is constant over the whole of the object, thus tending to favor a uniformly colored object. This simplistic model of appearance can easily be extended to incorporate a more image-based model. For example, Figure 6-2b shows a network that incorporates pixel values from the neighborhood around a given pixel location. If separate parameters are learned for each pixel node in this network, an image is learned. If a single mixture of Gaussians is tied across all the nodes (Figure 6-2c), the effect is similar to color histogram matching: all pixels must be drawn from the same distribution.

6.4.3 Hidden Markov Model

Variables q_{t-T} through q_t and v_{t-T} through v_{t-1} encode an HMM on the velocity of the tracked object over the last T time steps. Each HMM state $q_t = j$ encodes a

distribution over the two-dimensional velocity of the object at time t:

$$v_t = x_t - x_{t-1}$$

where x_t is the absolute position of the object at time t. The output distribution on v_t for state $q_t = j$ is Gaussian:

$$P(v_t \mid q_t = j) = \mathcal{N}(\mu_j, \Sigma_j).$$

Note that in the Bayesian network formulation of an HMM, the output probability distributions and transition matrix $P(q_t \mid q_{t-1})$ are tied across all time steps.

While in our previous work HMMs are used to model position, here we elect to use velocity features instead, as an HMM on velocity often results in a compact model for many motions that consist of straight-line ballistic movements. For example, many kinds of waving gestures may be specified with only two states. Furthermore, velocity as a feature has the advantage that HMMs built using velocity are invariant to translation.

The multinomial random variable g is a used to obtain a mixture of HMMs in the following manner. HMM states are grouped into $|g|$ disjoint sets, one set for each gesture g. Considering the Markov model view of the HMM, no transitions are allowed from one set of states corresponding to a gesture $g = i$ to a another set corresponding to $g = j$. Thus a single HMM is used to effectively implement $|g|$ separate HMMs. This arrangement is graphically illustrated in Figure 6-3. In this example, three HMMs are encoded by a mixture of HMMs, which has the single block-diagonal transition matrix

$$\begin{pmatrix}
1-a_{12} & a_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1-a_{23} & a_{23} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1-a_{45} & a_{45} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1-a_{56} & a_{56} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1-a_{78} & a_{78} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-a_{89} & a_{89} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

where $a = P(q_t \mid q_{t-1})$. The dependence of q_{t-T} on g is then configured such that g picks out one of the disjoint set of states that corresponds to a single gesture:

$$\begin{aligned}
P(q_{t-T} \mid g = 1) &= \begin{pmatrix} \pi(1) & \pi(2) & \pi(3) & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
P(q_{t-T} \mid g = 2) &= \begin{pmatrix} 0 & 0 & 0 & \pi(4) & \pi(5) & \pi(6) & 0 & 0 & 0 \end{pmatrix} \\
P(q_{t-T} \mid g = 3) &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \pi(7) & \pi(8) & \pi(9) \end{pmatrix}
\end{aligned}$$

where $\pi(j)$ is configured as an initial state distribution to model a start state for each gesture. If no start state is specified for the gesture, $\pi(j) = \frac{1}{3}$.

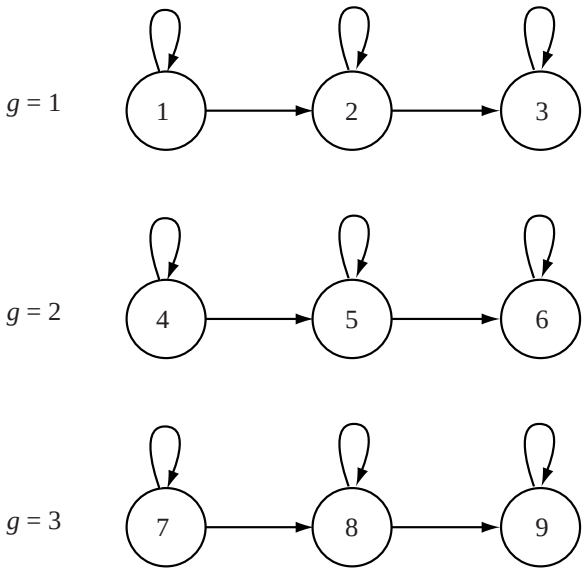

Figure 6-3: Three gestures may be encoded by a mixture of HMMs, which uses a single transition matrix to encode all three separate Markov models.

Note that even though g only impacts q_{t-T}, this is enough to obtain a mixture of HMMs, since by block diagonal construction of the transition matrix if q_{t-T} is in a one of the states belonging to gesture $g = i$ then so will the rest of the nodes q. The alternative of parenting, say q_t, with g introduces redundant parameters associated with modeling q_t as dependent on both q_{t-1} and g, a dependence already captured by the transition matrix.

After entering evidence at each v_t node, we may determine which of $|g|$ movements occurred by calculating the posterior probability of g.

6.4.4 Position Model

The absolute position x_t of the object is modeled as a distribution that depends on the HMM state at time t and the past position x_{t-1}:

$$P(x_t \mid x_{t-1}, q_t = j) = \mathcal{N}(x_{t-1} + \mu_j, \Sigma_j)$$

where μ_j and Σ_j are the same mean velocity and covariance of HMM state j as in the previous section. This part of the network is illustrated separately in Figure 6-4.

While the traditional HMM output probability distributions are distributions on *velocity* v_t, the arc from x_{t-1} to x_t encodes a Kalman filter-like integration of these velocity distributions to obtain a distribution over *absolute position*. In terms of the probabilistic forms introduced in Chapter 5, the linear regression matrix $W = I$ encodes the addition $x_{t-1} + v_j$ at x_t.

Intuitively, this part of the network incorporates the impact of the HMM with the known past position of the object to arrive at a distribution on the current position of the object.

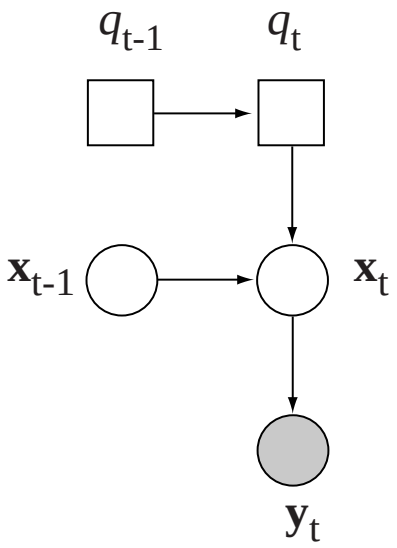

Figure 6-4: The tracking part of the network combines the past position of the object with current position observations and an HMM on object velocity.

The covariance at x_t reflects the model's uncertainty about the position of the object, where position is specified as a point in image coordinates. This uncertainty is the composition of two sources of uncertainty: uncertainty in the position of the object at time $t-1$, represented by covariance at x_{t-1}, and uncertainty in velocity distributions of HMM states at time t. These two sources of uncertainty may be made distinct setting the covariance at y_t to take into account the size and shape of the object as an ellipse.

The network models the position and appearance of a single pixel drawn from each of the past T images. In general we are not interested in modeling the color and position of pixels taken singly from each frame of an image sequence, since each object will consist of many pixels.

The network may be applied on multiple pixel observations by computing the average marginal for each posterior we are interested in, as described in Chapter 5. This proceeds by running the inference algorithm on the network for each observation and keeping summary statistics on the marginals of interest. Unfortunately, the combinatorics of the observation space are daunting: the number of possible value of y_t for an $N \times M$ image is $(NM)^T$. To follow an object through the entire T frames requires drawing a pixel that belongs to that object on each frame *simultaneously*.

The size of the observation space may be reduced drastically by considering only the current frame, and modeling the past velocity of the object as a known distribution. The resulting network is shown in Figure 6-5. In the reduced network, the object velocity at time t is denoted as v_t.

The approach of running the inference algorithm for every pixel in even only the most recent image is computationally prohibitive. However, a high percentage of the pixels in an image will belong to no tracked object. Instead of examining every pixel, we may sample the network to generate values of y_t that likely belong to the object.

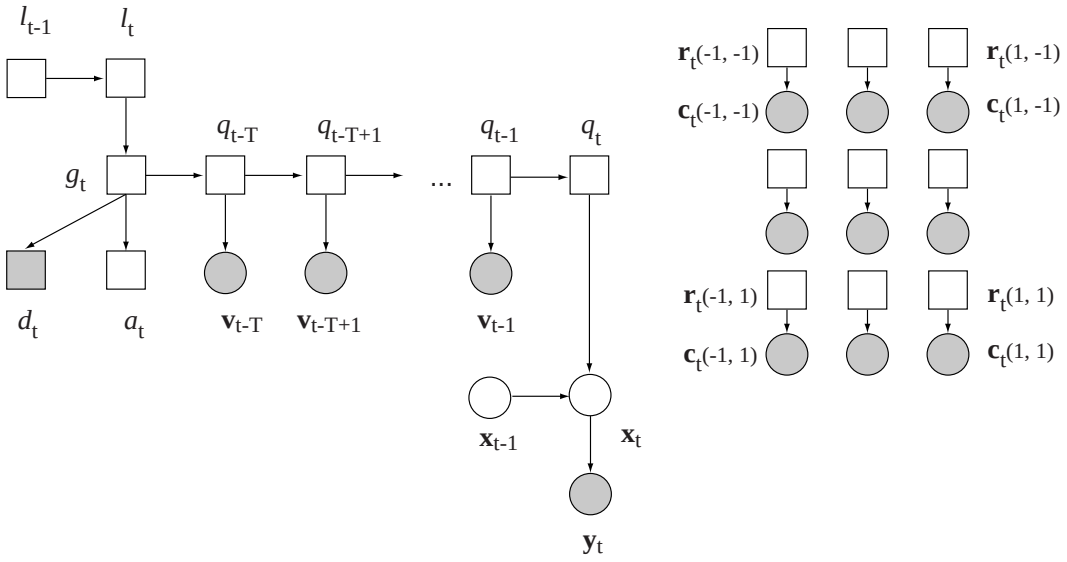

Figure 6-5: The approximation of the Bayesian network in Figure 6-1 with reduced observation space. v_t denotes the (known) velocity of the object at time t.

A random variable in a Bayesian network may be sampled by drawing samples from the variable's conditional distribution, given that its parents have been sampled as well. Once a number of y_t samples have been collected, the color of each pixel at y_t may be collected from the image.

An entire image may be examined by the network by entering every pixel of the image in turn, being careful to reject pixels with low likelihood, which are not likely to belong to the object. Alternatively we may deduce that a pixel does not belong to the object by comparing the likelihood for other tracked objects. The pixel then belongs to the object for which the likelihood is greatest. A related approach is to model a "junk" hypothesis, an object with a large size and appearance model with high covariance. Any pixel not ascribed to the target object will be attributed to the "junk" hypothesis, and may be subsequently ignored.

Note that the drawn samples of x_t are predictions based on the current knowledge of the current gesture taking place, the current HMM state and the past position of the object.

6.4.5 Action and Context

The dependence on gesture and action is implemented by the arc from g to a, with an associated probability table $P(a = a_i|g = g_j)$, which maps a given gesture to one of $|a|$ actions.

One can imagine a variety of ways to have the system generate actions given this map. One approach is to generate an action by sampling this distribution every time the current gesture changes. The current gesture may be computed as the posterior of g. The manner in which actions may be generated appropriately will of course be dependent on the application.

A dependence on time-varying domain context is similarly implemented by the arc from g to d. When the application enters one of $|d|$ context states, this may be entered as evidence at node d. The manner in which this impacts the current interpretation depends on the probability table $P(d = d_i|g = g_j)$.

The action and context maps connect the network with the application environment. For example, we may identify the fact that certain application events are likely to be accompanied by a THROW gesture. The occurrence of one of these events thus drives expectations for a THROW gesture, with $P(d = \text{THROWCONTEXT} \mid g = \text{THROW}) > 0$. This expectation guides the parse of the movement and all other aspects of the network's interpretation. This distribution may be learned, or set *a priori*.

Similarly, if the system sees a THROW gesture (in any context d), the mapping from $g = THROW$ to a generated action $a = \text{THROWEVENT}$ is specified by $P(a = \text{THROWEVENT} \mid g = \text{THROW})$. Again, this distribution may be learned, or set *a priori*.

6.4.6 Control

Lastly, nodes l_t and l_{t+1} model the evolution of our state of knowledge regarding the tracked object. The Markov model associated with the probability table $P(l_t \mid l_{t+1})$, illustrated in Figure 6-6, tracks the progress of the state of the tracked object from NEW, when it is just started, through SEEINGCATEGORICAL, which indicates that a known gesture is occurring, to VALID, to indicate that this tracked object has undergone a categorical movement at some point in the past. If when the object is NEW, no categorical motion is seen, the object may become DEAD. Similarly, if the object is VALID and no categorical movement has been seen in a long time, the object may also become DEAD. The DEAD state corresponds to the belief that the tracked object is not interesting as indicated by its patterns of movement over time.

The dependency $P(g \mid l_t)$ is used to select which set of HMMs is allowable in each of the $|l|$ states. $P(g \mid l_t = \text{SEEINGCATEGORICAL})$ is nonzero for any gesture g that is well known, while $P(g \mid l_t = \text{NEW}, \text{VALID}, \text{DEAD})$ may be nonzero for any gesture.

As an example, consider tracking the user's hand. We hope that the currently tracked object is the hand, but the system represents its uncertainty by $l_t = \text{NEW}$. If the currently tracked object moves back and forth, as in a waving motion that has been seen before when tracking hands, $l_t = \text{SEEINGCATEGORICAL}$. When hand stops waving, $l_t = \text{VALID}$ signifying that this object has undergone a known motion in the past, and therefore the increased confidence that the object is indeed the hand. Eventually, if no known motion is shown by hand, $l_t = \text{DEAD}$, signifying the lack of confidence as the object as the hand.

6.4.7 Multiple Objects

The network includes distributions for the position and appearance of a tracked object in the image. The nodes x_t and c_t model the (x, y) position and YUV color of a single

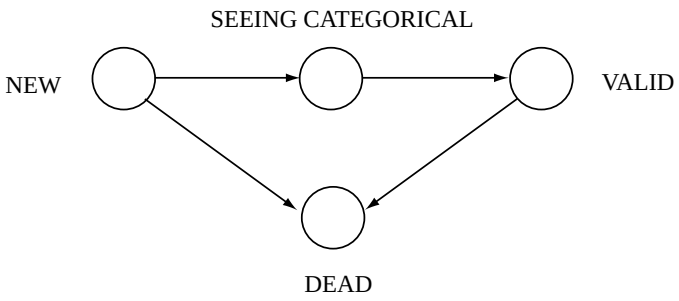

Figure 6-6: Control Markov model models the state of our knowledge of the tracked object.

pixel drawn from the image. When position and color information is entered on x_t and c_t, the likelihood as computed by the inference algorithm described in Chapter 5 will be higher if the pixel belongs to the object than if it does not. Both position and color must match.

Multiple objects may be modeled simultaneously by adding a multinomial variable h so that the network may model $N = |h|$ objects simultaneously. In this arrangement, h must be the parent of every node in the graph. Then, instead of using the overall likelihood of a pixel entered into the network, we may additionally consider the posterior probability of h to indicate to which object the pixel belongs. Similarly, in the case where only one object is under consideration, we may still use the multinomial h to model the true object and a "junk" hypothesis ($|h| = 2$), so that the posterior of h may be used in the same manner.

Alternatively, we may maintain N distinct copies of the same network and test the pixel against each of N networks. Both approaches have the same computational complexity.

6.4.8 Propagation

The dynamic Bayesian network includes variables l and x that model dependencies on the previous time step. It also includes an unrolled dynamic Bayesian network for the HMM on velocity of each blob. Since the HMM may be unrolled sufficiently many time steps T to capture entire gestures at once, we will require that only l and x be propagated from time step to time step in the manner of a dynamic Bayesian network. This is done at each time step in the manner described in the previous chapter, where the marginal of the variable at time t is used to set the parameters of the corresponding $t-1$ variable in the next time slice. In the case of multiple pixels, these marginals are the "aggregate" marginals as described previously.

6.4.9 Splitting the Network

We may use samples in an even more efficient manner if we also split the network into two pieces: a low-level network that benefits from examining many samples, and a high-level network that requires many fewer samples. The lowest level parts of the network, including position and appearance, benefit from having many pixels

examined so that the appearance of the object is sampled many times, while the high-level parts of the network, including the control network and the HMM, require many fewer samples.

Samples of y_t may be generated from network. The color values at each of these positions is then collected from the current image. These color values are then entered into the appearance network in turn. The likelihood of each of these color values is determined by inference on the appearance network. All the but the top few of the samples are then discarded, as ranked by appearance model likelihood.

Each of the remaining samples are then entered into the high-level network, which is responsible for the frame to frame tracking of each hypothesis given the previous state of the HMM. The "aggregate" marginal is then used to propagate the value of x_t to the next times step.

The position of the object determined, the *velocity* of the object may be computed from a stored buffer that records the position of each blob over time. The velocities v_{t-T} through v_{t-1} may then be entered into the HMM. Posteriors on g may then be computed, and l may be propagated to the next time step. Thus the state of the HMM is propagated, as well as x, to impact the sampling process for the next time step.

The splitting of the network is based on the observation that the high-level network does not need highly accurate spatial position information. Similarly, the high-level network does not require a high degree of temporal resolution since, for example, the posterior of q_t is not likely to change at frame rate. Therefore the high-level network may be updated asynchronously with the mid-level network, further saving computation.

As an example, again consider tracking the user's hand. The image around the predicted location of the hand is sampled. This predicted is based on the past position of the object and the current state of the HMM. All but a few of these pixels are rejected based on the fact that they do not fit the color-based appearance model. The remaining samples are then used to update the tracking model, to give an updated position of the hand. This position is used to calculate the current velocity of the hand, the history of which is entered into the high-level network. The posterior of g indicates that the hand is undergoing a waving motion, while the control network is propagated such that $l_t = SEEINGCATEGORICAL$ to represent the fact that the object is undergoing a known motion. Belief about the HMM step for the next time step is then computed from the high-level network and then entered on the mid-level tracking network.

6.4.10 Algorithm

In summary, the sampling, inference and propagation algorithm proceeds as follows:

1. sample the high-level network to obtain many samples of x_t
2. look up each color c_t at each location x_t in the image
3. enter each value of c_t into the appearance model, and record its likelihood

4. keep the top few (x_t, c_t), as sorted by appearance model likelihood

5. enter each top x_t value into the high-level network in turn, and compute the aggregate posterior x_t

6. compute $v_t = x_t - x_{t-1}$

7. enter v_t through v_{t-T} and d_t into the high level network and run the inference algorithm

8. propagate x_t and l_t to the next time step

Note that this algorithm encodes several approximations outlined earlier, including propagating over a small number of samples instead of the whole image (or the whole image sequence) and propagating only samples that have high likelihood by the appearance model. In the case of tracking multiple objects, this process is repeated for each replication of the network, or is executed only once if a single network with multinomial random variable h is used.

6.4.11 Scaling Issues

One way to characterize the computational complexity of a Bayesian network is to calculate how the clique sizes scale with various other quantities. For the network presently under consideration, there is no clique that grows with the length of the sequence T.

One concern is how the number of parameters to the Bayesian network grows as the number of gestures increase. As can be seen from the example transition matrix presented in section 5-8, the number of parameters to the transition matrix grows exponentially with the number of gestures. Many of these parameters are zero, and so a real-time implementation of Bayesian networks may use this fact to reduce the number of mixture components considered in further calculations, effectively removing this issue as a concern.

Presently, each HMM state has a unique output distribution assigned to it. If the number of gestures is large, this approach would likely lead to a redundant set of output distributions. In this case an approach whereby output distributions are shared among states would be desirable. This can be accomplished by incorporating a new discrete random variable r and conditional probability $P(r \mid q)$ with output distributions in turn selected by the value of r. This conditional probability $P(r \mid q)$ maps the states q onto output distributions indexed by r, and would have to be trained for new gestures.

6.5 Experiments

For the following experiments the networks described above were implemented in the hybrid Bayesian network software package described in Chapter 5. The experiments

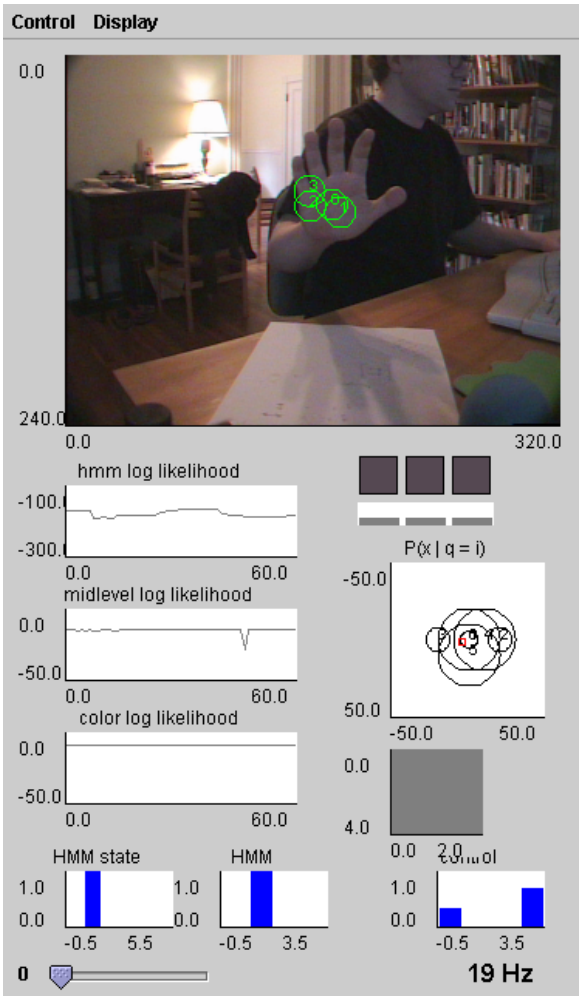

Figure 6-7: Graphical interface to the runtime experimental system.

demonstrate a few of the many interesting inference and learning scenarios made possible by the Bayesian network architecture.

Figure 6-7 shows a screen shot of the running experimental system. Example video of the runtime system for each of the experiments is located at `http://www.media.mit.edu/~drew/movies`. The following experiments report data taken from a *single session* of the experimental system.

6.5.1 Learning Appearance Model Driven By Categorical Movement

A multiple hypothesis approach to object discovery is necessary when tracking distinct objects by appearance. In the first experiment we outline how the control network models our belief that each hypothesis is a valid object. Appearance models are induced implicitly when the system happens upon a object that moves in a familiar way and the control network preserves the hypothesis.

To demonstrate this process, we would like the network to pick up on the appear-

ance of an unknown object when it undergoes a simple waving motion, as if the user is trying initiate a dialog with the system by waving at it. One of the $|g|$ gesture models was hand coded for a back and forth movement, approximately a waving motion. Only two states were used in the HMM. The HMM of the network was unrolled to $T = 30$ time steps, about a second and a half of clock time. The network was configured to follow four object hypotheses, and the appearance model was configured to include three Gaussian components, each on YUV color values.

Two additional "null gestures" are defined: g = STATIONARY to model the absence of motion, and g = RANDOM to model movement that are modeled by no other gesture. Both are modeled by a single state with zero velocity, STATIONARY wth a small covariance, RANDOM a large covariance.

Each tracking hypothesis is "seeded" in a region of the image in which there is some image motion. These regions are modeled by a mixture of Gaussians over image coordinates, which is trained in an online fashion by sampling the image and entering as observations those image locations which have changed more than some threshold value. A tracking hypothesis is seeded again when l_t = DEAD, implying that under that hypothesis appearance model, no categorical motion was seen. The transition probabilities of the control Markov model are configured such that once a hypothesis undergoes a categorical motion, many timesteps must pass without seeing another categorical motion before l_t = DEAD.

The output distributions for the Markov model on l_t is constructed to support the above process as follows: For l_t = VALID, g may be STATIONARY or RANDOM. For l_t = SEEINGCATEGORICAL, g may be STATIONARY or L/R WAVE. For l_t = DEAD, g may be again STATIONARY or RANDOM.

During runtime, the user waves a color object in front of the camera. The amount of time it takes for some hypothesis to correctly acquire the object depends on how much motion there is in the scene. If there is no other motion in the scene, the network will begin tracking the correct object after about three or four periods of a wave. A video sequence illustrating an example of this process is located at `http://www.media.mit.edu/~drew/movies`.

Figure 6-8 shows the posteriors of l_t, g_t and q_t over time during the object acquisition phase.

On occasion, the algorithm will use one or more blob hypotheses to track the users arm or hand, or the shadow of the arm or hand instead of the object itself. Presently, there is no way to avoid such ambiguities.

6.5.2 Learning Novel Gestures

Once the appearance model of the target object as been acquired, the network may be used to induce models of gestures that the system has never seen before.

In this demonstration, a gesture model is initialized with high covariance for each state output distribution. Configured this way, a sequence that is not modeled by the known waving gesture will be assigned to the new gesture model. The new gesture is trained in an online fashion with an EM algorithm: after each time step, the statistics outlined in Chapter 5 are recorded for each state output probability distribution. This

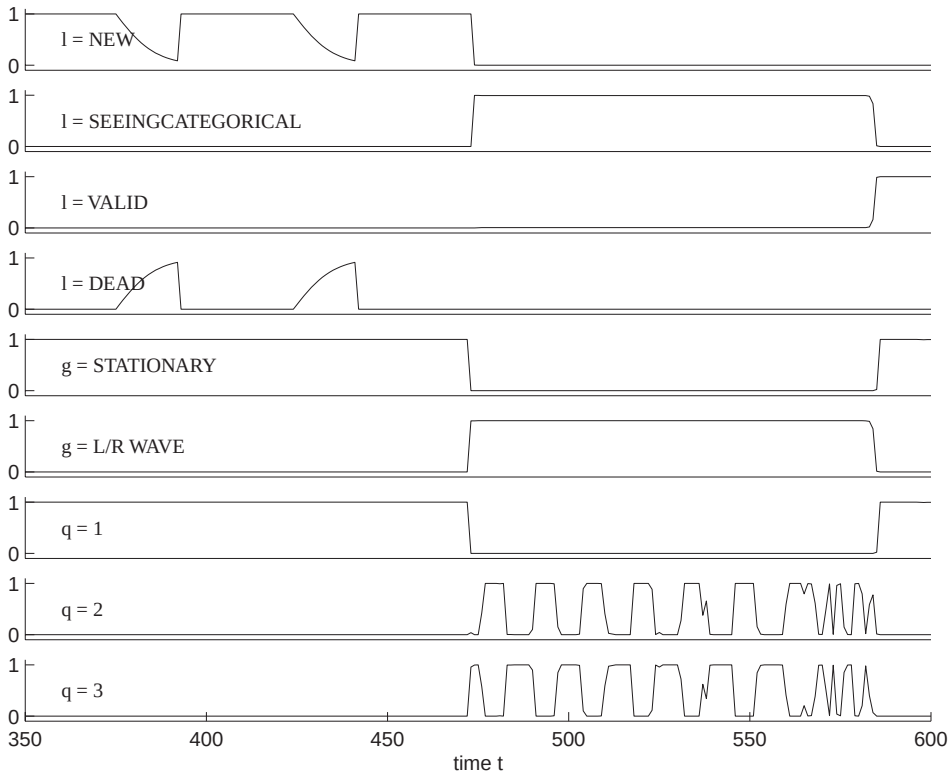

Figure 6-8: Plot of posterior probablities of l_t, g_t and q_t during the acquisition of an object. The object is repeatedly restarted until frame 470 when the tracked object exhibits a motion described by HMM g = L/R WAVING, a left to right waving motion. At that time, the control network registers l_t = SEEINGCATEGORICAL. The waving gesture is approximated by two states, $q = 1$ and $q = 2$, which alternate. When the gesture ends the control network registers $l = VALID$ denoting that the object has undergone a known categorical motion, and consequently should be saved.

is the expectation step. Then, at every tenth tick, the maximization step is executed to arrive at new values of the parameters of the output distributions. The novel gesture is gradually trained after only a few examples.

The Markov model associated with l_t allows the training of the new gesture only after $l_t =$ VALID, by initially allowing $g = TRAINED$ only for $l_t = VALID$. Figure 6-9 shows posterior values over time of l_t, g_t and q_t during the time the novel gesture is learned. Figure 6-10 shows the output distributions for the two states of the gesture model for the novel movement.

Note that the acquisition of the new gesture proceeds in an unsupervised fashion. If a domain context signal is available, it may be used to train a different "unused" gesture, such that the resulting gesture to action mapping is correct for that gesture. In the more difficult unsupervised case, feedback may be used to train the gesture to action mapping so that the gesture corresponds to an appropriate output action.

Because the target object was presumably selected because it exhibited a known categorical motion, it is reasonable to incorporate the known gesture into the network such that it also may be considered a known gesture, and so may also be used as a cue for object discovery. This can be accomplished by creating an observation on the variables l and g, where $l_t =$ SEEINGCATEGORICAL and $g = i$ the new gesture. The observation may be incorporated into running statistics of g, the parameters of which may be updated periodically.

6.5.3 Learning Gesture Model to Action Mapping

In the last experiment, we show how a simple reinforcement learning-based mechanism may be used to train the gesture to action mapping. In this demonstration, we assume that actions are generated from the system when the current gesture g changes. We define the current gesture as the gesture for which the posterior of $g = i$ is maximum.

An action is generated by sampling the action selection matrix $P(a = i \mid g = j)$. The last action and gesture pair is saved to memory, such that when the user offers positive or negative feedback, this pair is used to generate an observation which is then incorporated into the running statistics of the action selection matrix.

Specifically, if we denote $(a = i, g = j)$ as the last action and gesture pair, and the system receives positive feedback at a subsequent time step, then the observation $(a = i, g = j)$ is added to variable a's statistics. If we have the additional constraint that gestures and actions are in a one-to-one mapping, we may also add a "soft" observations on all gestures other than k:

$$P(a = i \mid g = k) = \begin{cases} 1/(|g| - 1), & \text{if } k \neq j \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, if the feedback is negative, we may add a soft observation on a:

$$P(a = k \mid g = j) = \begin{cases} 1/(|a| - 1), & \text{if } k \neq i \\ 0, & \text{otherwise.} \end{cases}$$

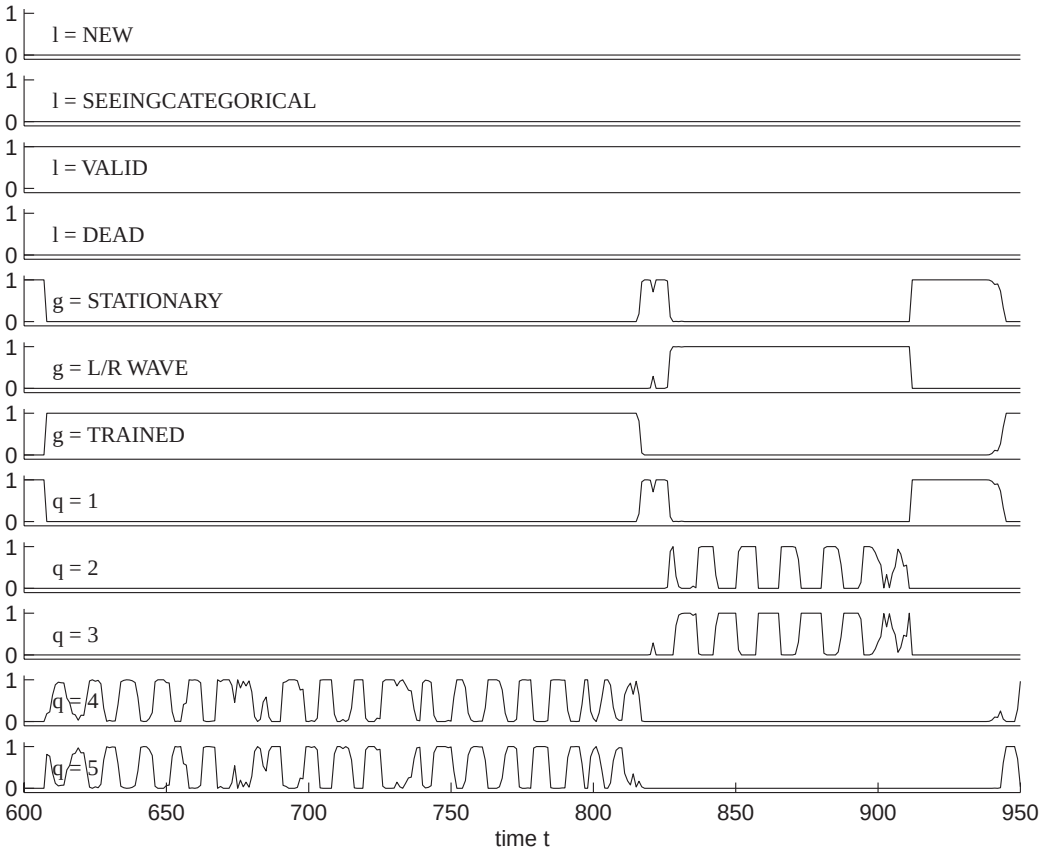

Figure 6-9: At time $t = 600$ a novel movement is approximated by neither $g =$ STATIONARY nor $g =$ L/R WAVE. The gesture model $g =$ TRAINED is trained to model the up and down movement of the tracked object. Later, at frame 820, the system sees a known left to right waving gesture.

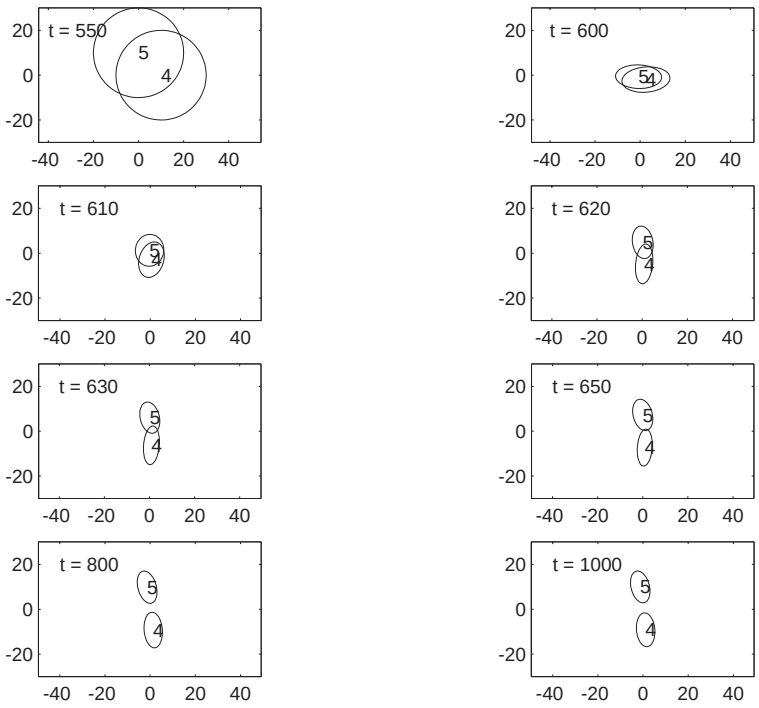

Figure 6-10: Continuous online training of the novel gesture is shown at several time steps in left to right, top to bottom order.

and analogous observations on remaining gesture models such that the one-to-one mapping is preserved.

At the start of training the gesture to action mapping, when there are few example pairs available, the change in the mapping after a single example will be great. This early rate of adaptation my be slowed by adding a Dirichlet prior to g [27], which amounts to adding synthetic (uniform) observations to a variable before runtime.

Figure 6-11 shows the posterior value of g_t over time, the points in time when action was taken and the corresponding feedback given by the user. Figure 6-12 shows the action selection matrix $P(a_t \mid g_t)$ over the same time course. In this example, only positive and negative feedback observations are used, without observations to enforce a one-to-one mapping.

6.6 Conclusion

A framework for the online learning of gesture has been introduced. A dynamic hybrid Bayesian network which incorporates a variety of information is demonstrated to support a number of interesting inference and learning scenarios. The model is able to leverage previously learned gesture models, exploit domain context information, learn actions associated with gestures, and control the development of multiple tracking hypotheses. A taxonomy of successively more constraining movement models is combined with a sampling-based approach, allowing an efficient real time implemen-

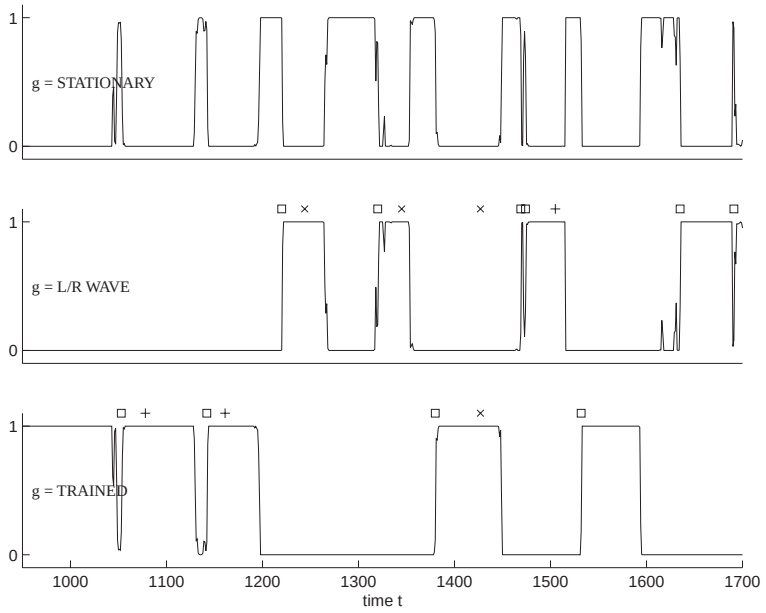

Figure 6-11: The posterior value of g is shown over time. When the value of g changes, an action is emitted (square). At some point later the user provides either positive (plus) or negative (cross) feedback on the action taken by the system.

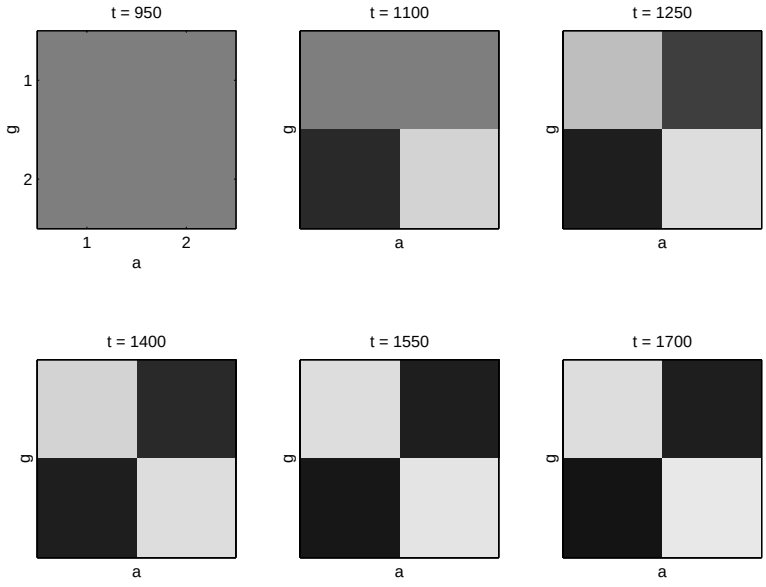

Figure 6-12: The action selection matrix $P(a_t \mid g_t)$ at several points during the time depicted by Figure 6-11.

tation.

The combination of the above sources of information into a single probabilistic model gives the designer many options in deciding what to learn and when. The experiments presented here demonstrate only three such scenarios: the acquisition of a simple appearance model driven by known categorical movement models, the subsequent acquisition of new gesture models, and a reinforcement learning-based learning of the mapping from gesture to action.

The flexibility of the Bayesian network approach will allow other novel scenarios. For example, it reasonable to consider training the mapping from gesture to domain context events. As with the demonstrated scenarios, it will be important to ensure sufficient constraints so that the learning will be easy. In developing ever-more complex models for online learning, this concern will be paramount, and will require a careful crafting of the model and design of the interaction and feedback.

Chapter 7

Conclusion

7.1 Summary

This thesis presents four projects which demonstrate the adaptive approach to gesture recognition, where in each case what is adapted and what drives the adaptation varies. Taken together, they demonstrate that the adaptive approach may be applied in a wide variety of gesture recognition scenarios. Each differs significantly along the lines of the problem addressed, the nature of the constraints brought to bear, and mathematical techniques.

The Parametric Hidden Markov Model (PHMM) demonstrated how the HMM formulation may be extended to represent families of gestures, and how the PHMM may be used to extract "how" a gesture is executed, which in many dialog situations is a important piece of information not found in the spoken dialog. Determining how a gesture was executed proceeds by adapting a parameter in an online fashion. Constraints on this adaptation include a learned manifold representation of the gesture family.

The natural gesture video parser exploits a rather different set of constraints. A coarse model of the behavior of a person telling a story is leverage to adapt appearance models of the speaker at rest. These rest state appearance models are then exploited to perform a more complete parse of the speaker's gestures.

The Watch and Learn system builds upon the natural gesture video parser by adapting all the appearance models of the gesture model in a real time implementation. This adaptation is driven by a time-varying context signal that is derived from application context. Once the appearance models are learned, the system is able to parse the gesture without the aid of the context signal.

The final project demonstrates the implementation of a Bayesian network-based system for online gesture learning. This ties together mathematical ideas of the PHMM and the idea of driving adaptive appearance models from motion as demonstrated by the natural gesture parser and Watch and Learn, while extending the ideas in a variety of directions. For example, the notion of driving the learning from multiple sources of constraint is explored.

7.2 Features and Behavior

The various adaptive models presented in this thesis exploit a number of notions of behavior, each coupled with a different set of features computed from sensory input. For example, the PHMM has been demonstrated on the three-dimensional position of the head and hands of the user, the natural gesture parser and Watch and Learn use low-level image-based appearance models, and the final Bayesian network-based system exploits a combination of tracked objects and appearance models.

The choice of features impacts the nature of the behavior model. For example, in Watch and Learn, each of the HMM states corresponds to an appearance model which is learned during runtime. At the start of the system when the appearance models are unknown, the HMM is driven completely by application context. Thus the behavior model must be matched to the application context information available to the system. As the appearance models are learned, the same HMM must be able to approximate changes in the video input over time. While the particular examples presented are quite simple, in general, the HMM topology and states must be chosen carefully to work with the context information and permit a working, learnable HMM on appearance models.

In the frameworks that rely on tracking information, the model of behavior is linked tightly with image geometry. In the Bayesian network-based framework, for example, this means that two gestures that we identify as having the same behavior may not be learnable under the same model if they exhibit different movement in space. Note that one of the primary motivations of the PHMM is to abstract over these differences in geometry in the case when the different versions of the same behavior lie on a smooth manifold in space. Turning to natural gesture, the PHMM approach will not work in the case of learning beat gestures, for example, because they typically do not exhibit geometric regularity, but rather regularity in a motion profile over time.

Low-level appearance-based representations are attractive from the standpoint that they can be fit easily to any image-based data easily. But there are limits to this approach as well. For example, the simple approach used in this thesis is unable to model changes in the appearance of the object (though see [73] for a possible approach that is compatible with the PHMM). The behavior models built on simple image-based approaches will thus not be able to model the rotation of the object. Similarly, significant motion in depth will not be learnable as such.

If we have a kind behavior model that we would like to use, is it possible to determine automatically what features best support the desired behavior model? In general feature selection is a difficult problem. But if there is a set of features that are at the model's disposal, it would be possible to try subsets or combinations of features until features are found that support the behavior model. The goal of such an optimization would be to employ compact, reusable high-level behavior models across multiple systems. In fact, this process has much the same flavor as the adaptive approach presented in this thesis; instead of learning images or trajectories, combinations of features are learned in support of a known behavior model.

Given the variety of frameworks and applications presented in the thesis, are there

any generalizations that may be made to indicate what features a general behavior model might have? Considering only the space of systems demonstrated by the thesis, I believe that the most important features of a behavior model applied adaptively are that it must first represent the ordering of events in time. That is, state B typically follows state A, without respect to how the states 'image' in feature space. The important aspect of such a model is that at some point, semantics must be bound to each of the states.

Secondly, behavior models should exploit context. It may appear as if context is just another set of features which are known *a priori*. But the important aspect of context is that it bootstraps learning processes and indexes into previous experiences, but otherwise is not complete enough to suffice as a whole behavior model unto itself. Correlating context with the current set of features allows an adaptive process to learn the order of events in a behavior learn their sensory representations, bind these events to useful semantic labels, and ultimately learn new contexts.

7.3 The Field of Machine Perception

The adaptive gesture models above were developed in response to the dominant pattern recognition paradigm, in which static models are employed to perform recognition. To some extent the progression of models and techniques reflects the changing in my thinking about the problems of gesture recognition and pattern recognition, my reaction to the overemphasis of bottom-up tracking in the field of computer vision and the promise of adaptive systems. In particular, the work presented supports the following:

- Domain context is a strong and often overlooked source of constraint that may be used to solve pattern recognition problems. In some cases, the strength of domain context will expose the fact that precise tracking is unnecessary in many circumstances which were previously assumed to require tracking.

- The online adaptive approach will be more generally applicable than the alternative approach of a strictly memory-based system. The generalization ability of the online adaptive approach stems from the ability of such systems to adapt to circumstances that were not encountered in offline training scenarios.

- Bayesian networks are a powerful, expressive mathematical framework for constructing adaptive systems that incorporate a variety of modeling constraints. Bayesian networks subsume many of the important probabilistic frameworks, while allowing interesting hybrid structures adept at combining multiple sources of information and similarly permit many types of learning scenarios.

- The online adaptive approach is one technique to address some of the standard problems in applying computer vision techniques to the task of watching and interacting with people.

I hope the ideas presented in this thesis will encourage future researchers to think about their gesture recognition problem in a sophisticated way. In the remainder of this chapter I speculate on the utility of adaptive gesture models beyond the projects presented in this thesis.

7.4 The Future of Adaptive Interfaces

Adaptive gesture recognition models will play a role in designing interfaces for tomorrow's ubiquitous computing environments. As argued in Chapter 1, the convergence of a large variety of computing devices and increased sophistication of user interaction will encourage the design of interfaces that learn to work with the user. These adaptive interfaces will be one mechanism to deal with the emerging complexity of interfaces. We are already seeing the beginnings of this movement: the two most popular examples may be the newer consumer speech recognition systems that require "training" passages to be spoken, and the Microsoft Office Assistant, which not so surprisingly is built on a discrete Bayesian network [28].

Designers of tomorrow's interfaces will use adaptive models because in many cases it will be the only way to get things to work. Consider, for example, a computer vision system that is to be deployed in indoors to monitor and interact with people in an home environment. Now consider the harsh reality of selling a production version of the system. The user buys the system at a retail store, unpacks it at home to find a couple of small cameras that must be calibrated to the geometry of the rooms in which they are deployed. Meanwhile the lighting must also be calibrated or perhaps even changed to match a factory-specified skin color model model. After going through all this, the user is disappointed to find out that many of the gestures that the system requires don't feel natural, and must be rehearsed many times before they become second nature and are reliably recognized by the system. I doubt most people would have the patience to configure such a system, and although it may be reasonable to have a specialist initially configure the system, having one come out for every significant change in the household will significantly slow the adoption of the system.

An adaptive interface will be more likely succeed in such a demanding environment, but only if the user understands that the system is adaptive. One challenge in the design of an adaptive system is how the overall system should behave when it is in the midst of adaptation, and possibly has an incomplete model of domain context and the user. In the worst case, the system will be offering up responses that are almost always inappropriate, and the user will be continually badgered to deliver feedback until the system finally learns the desires of the user. A more reasonable approach is to have the system first observe how things normally work, and incorporate these observations into its model. Later, as the system is able to reliably predict the outcome of some of the events, it will begin to offer appropriate responses with some degree of confidence.

One stumbling block for the adaptive interfaces approach is the fact that in today's world, people expect machines to behave deterministically, and so any machine that doesn't perform identically given the same input time after time is "broken" or "un-

reliable". By definition, adaptive interfaces do not perform identically an successive occasions. This concern will be addressed in part by designing adaptive interfaces to represent their state of knowledge in a way that is accessible to the user. This effectively returns the machine to the class of deterministic machines by exposing the hidden state. How these systems should represent their internal state to the user is a delicate design problem that depends greatly on the domain and the attitude of the user towards the system.

I believe that this problem of the non-deterministic nature of an adaptive interface will fade as a real concern, however, as the interfaces become so sophisticated that it is too difficult to depict the state of the system, and, more significantly, that the average user will be used to the idea of machines that learn. Tomorrow's user will *expect* the machine to pick up on how things work.

Lastly, I would like to suggest that in the near term, adaptive interfaces will not have a more motivated application than in assisting the disabled and elderly populations lead their lives. One of the greatest challenges facing these segments of the population is that it seems as if each person has a unique set of problems. Often these problems are addressed half-heartedly by publicly available products or services, or no solution addresses the need satisfactorily and a custom solution must be devised at significant cost. Too often the problem is inadequately addressed, partly due to cost, or a matter of not having the right people tend to the person's need. A sufficiently sophisticated adaptive interface would be able to compensate for the person's disability, working with the person's own remaining functional abilities, as well as changing in response to changes in the nature of the disability.

Obviously, the work presented in this thesis only scratches at the surface of what will required of tomorrow's adaptive interfaces, and only hints at what will be possible in the future.

7.5 The Role of Movement in the Natural World

Not much is known about the high-level workings of animal visual perception, so it is difficult to say whether animals actually perform the top-down interpretation of movement. Certainly there are many examples of so-called "pre-attentive" processes that drive things that move to be considered further. For example, reptiles are almost blind to objects that don't move, instead relying on their keen sense of smell. And there are plenty of demonstrations of perception guided by context, mostly in the form of infant visual perception studies. This suggests that a similar context-guided perception of movement is possible.

A robust common example of how top-down expectations of movement guide perception happens when tossing a ball for a dog. If the person throwing the ball goes to throw it but then doesn't release the ball, the dog will often look where the ball would be had the ball been released, and perhaps will even go running in that direction. While it is possible that the dog has some experience with the physics of the world, it is also likely that the dog is drawing on a ball-playing context built up over years of playing ball with his owner.

Like the dog owner magicians similarly play on our expectations of where objects are. A domestic cat will likely find television fairly uninteresting until a nature show involving some sort of small animal or bird comes on. Rodents are known to freeze when they see a large object that passes over them. While the focus of this thesis has not been on biological plausibility it is interesting to consider examples of motion selectivity in nature.

7.6 Adaptive Approach as a Computational Model of Perception

Around the time when I began to think deeply about human movement and gesture recognition, I had an epiphany when I watched again the computer graphic animated short *Luxo Jr.*[42]. Written and produced by John Lasseter of Pixar, *Luxo Jr.* tells a short story about two lamps, which as a viewer, we easily anthropomorphize into a mother (or father) and child (see Figure 7-1). We quickly make the deduction that the head of each lamp is the head of each character, the hinge at the middle of the lamp is the waist, and so on. I found several other examples of animators similarly bringing inanimate objects to life, but I found none that illustrate our ability to anthropomorphize as well. What is going on when we view this film for the first time? Is it possible to develop a computational theory of how lamps are viewed as human? Why would we care if we could?

To no one's surprise, while animators have amazing abilities to synthesize such wonderful examples to tickle the brain [69, 43], they offer very few clues how the interpretation of their animations proceeds in the heads of their viewers. I argue that demonstrations such as *Luxo Jr.* indicate that perception is an active, iterative process that is not strictly based on memory alone. Seeing *Luxo Jr.* for the first time probably involves a back and forth process, in which a bit of the lamps motion is interpreted, the outcome of which is used to initiate an interpretation of the lamp's geometry, which is used to refine the interpretation of the motion, which refines the geometry, and so on. Small gestures such as the small lamp looking up at the larger lamp serve to lock in the interpretation even further. All our domain-specific context knowledge about children playing with balls with mother looking on is brought to bear as well, as well as the pervasiveness of the laws of physics. Quickly, all this disparate information is combined to arrive at a strong, unambiguous perception of a human character.

Does this mean we are running an EM algorithm in our heads when we view something new? Probably not. But the iterative flavor of my account of how we see the film has the feel of EM, and suggest that todays probabilistic systems that use EM for adaptation may give us a start on a computational theory. The adaptive approach to gesture recognition similarly uses an iterative approach to lock in on an interpretation, while the demonstrations of the approach presented in Chapter 6 presents the combination of multiple kinds of information such as movement, context and appearance to arrive at a preferred interpretation. Adaptive approaches such

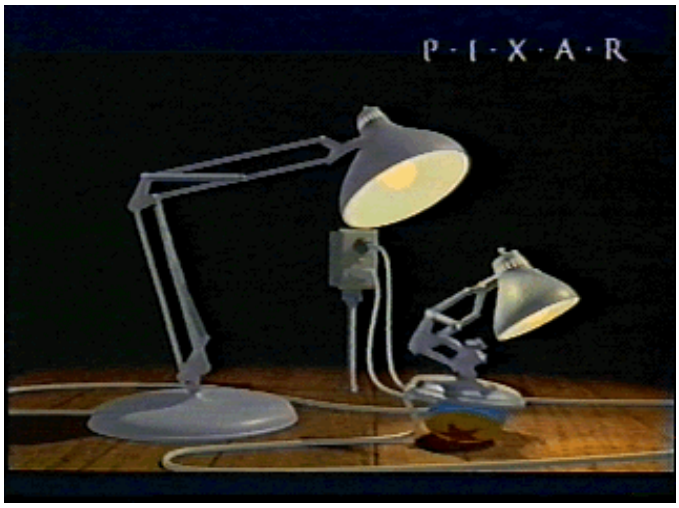

Figure 7-1: A frame from the Pixar short *Luxo Jr.*.

as those presented in this thesis may eventually provide the basis for a powerful computational theory of active perception. A good place to start is with structures like that presented in Chapter 6.

If a computational model of active perception is developed to the point where a system may come to understand *Luxo Jr.* without previous knowledge of how a lamp may be anthropomorphized, I believe we will have arrived at a model that is general enough to address many of the more practical concerns of developing systems that monitor and interact with humans. The reasoning is simple: if the system can build up a model of the lamp as a human, it ought be able to interpret any human as human, as well as any human can. Thinking about the problem of interpreting the lamp forces us to throw out everything that is irrelevant in interpreting humans as human. The concepts we will have to throw out are the very same ones that trip up todays gesture recognition systems.

Again, the present work only scratches at the surface of anthropomorphizing *Luxo Jr.*, but it offers ideas and techniques that are relevant and hopefully will stimulate further research into this fascinating and important problem.

Appendix A

Expectation-Maximization Algorithm for Hidden Markov Models

We derive equation 2.3 from the expectation-maximization (EM) algorithm [22, 3] for HMMs. In the following, the observation sequence $\mathbf{x}_t$ is the observable data, and the state q_t is the hidden data. We denote the entire observation sequence as $\mathbf{x}$ and the entire state sequence as $\mathbf{q}$.

EM algorithms are appropriate when there is reason to believe that in addition to the observable data there are unobservable (hidden) data, such that if the hidden data were known, the task of fitting the model would be easier. EM algorithms are iterative: the values of the hidden data are computed given the value of some parameters to a model of the hidden and observable data (the "expectation" step), then given this guess at the hidden data, an updated value of the parameters is computed ("maximization"). These two steps are alternated until the change in the overall probability of the observed and hidden data is small (or, equivalently, the change in the parameters is small). For the case of HMMs the E step uses the current values of parameters of the Markov machine — the transition probabilities a_{ij}, initial state distribution π_j, and the output probability distribution $b_j(\mathbf{x}_t)$ — to estimate the probability γ_{tj} that the machine was in state j at time t. Then, using these probabilities as weights, new estimates for a_{ij} and $b_j(\mathbf{x}_t)$ are computed.

Particular EM algorithms are derived by considering the auxiliary function $Q(\phi' \mid \phi)$, where ϕ denotes the current value of the parameters of the model, and ϕ' denotes the updated value of the parameters. We would like to estimate the values of ϕ'. Q is the expected value of the log probability of the observable and hidden data together given the observables and ϕ:

$$
\begin{aligned}
Q(\phi' \mid \phi) &= \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log P(\mathbf{x}, \mathbf{q}, \phi')\right] && \text{(A.1)} \\
&= \sum_{\mathbf{q}} P(\mathbf{q} \mid \mathbf{x}, \phi) \log P(\mathbf{x}, \mathbf{q}, \phi') && \text{(A.2)}
\end{aligned}
$$

where $\mathbf{x}$ is the observable data and the state sequence $\mathbf{q}$ is hidden. This is the

"expectation step". The proof of the convergence of the EM algorithm shows that if during each EM iteration ϕ' is chosen to increase the value of Q (i.e. $Q(\phi' \mid \phi) - Q(\phi \mid \phi) > 0$), then the likelihood of the observed data $P(\mathbf{x} \mid \phi)$ increases as well. The proof holds under fairly weak assumptions on the form of the distributions involved. Choosing ϕ' to increase Q is called the "maximization" step.

Note that if the prior $P(\phi)$ is unknown then we replace $P(\mathbf{x}, \mathbf{q}, \phi')$ with $P(\mathbf{x}, \mathbf{q} \mid \phi')$. In particular, the usual HMM formulation neglects priors on ϕ. In the work presented here, however, the prior on $\boldsymbol{\theta}$ may be estimated from the training set, and furthermore may improve recognition rates, as shown in the results presented in Figure 2-10.

The parameters ϕ of an HMM include the transition probabilities a_{ij} and the parameters of the output probability distribution associated with each state:

$$Q(\phi' \mid \phi) = \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log \prod_t a_{q_{t-1}q_t} P(\mathbf{x}_t \mid q_t, \phi')\right] \tag{A.3}$$

The expectation is carried out using the Markov property:

$$\begin{aligned} Q(\phi' \mid \phi) &= \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\sum_t \log a_{q_{t-1}q_t} + \sum_t \log P(\mathbf{x}_t \mid q_t, \phi')\right] \\ &= \sum_t \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log a_{q_{t-1}q_t} + \log P(\mathbf{x}_t \mid q_t, \phi')\right] \\ &= \sum_{t,j} P(q_t = j \mid \mathbf{x}, \phi)\left[\sum_i P(q_{t-1} = i \mid \mathbf{x}, \phi) \log a_{ij} + \log P(\mathbf{x}_t \mid q_t = j, \phi')\right] \end{aligned} \tag{A.4}$$

In the case of HMMs the "forward/backward" algorithm is an efficient algorithm for computing $P(q_t = j \mid \mathbf{x}, \phi)$. The computational complexity is $O(TN^k)$, T the length of the sequence, N the number of states, $k = 2$ for completely connected topologies, $k = 1$ for causal topologies. The "forward/backward" algorithm is given by the following recurrence relations

$$\alpha_1(j) = \pi_j b_j(\mathbf{x_t}) \tag{A.5}$$

$$\alpha_t(j) = \left[\sum_i \alpha_t(i) a_{ij}\right] b_j(\mathbf{x}_t) \tag{A.6}$$

$$\beta_T(j) = 1 \tag{A.7}$$

$$\beta_t(j) = \sum_j a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j) \tag{A.8}$$

from which γ_{tj} may be computed:

$$\gamma_{tj} = \frac{\alpha_t(j)\beta_t(j)}{P(\mathbf{x} \mid \phi)} \tag{A.9}$$

In the "maximization" step, we compute ϕ' to increase Q. Taking the derivative

of equation A.4 and writing $P(q_t = j \mid \mathbf{x}, \phi)$ as γ_{tj} we arrive at:

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')} \tag{A.10}$$

which we set to zero and solve for ϕ'.

For example, when $b_j(\mathbf{x}_t)$ is modeled as a single multivariate Gaussian $\phi = \{\mu_j, \Sigma_j\}$ we obtain the familiar Baum-Welch reestimation equations:

$$\mu_j = \frac{\sum_t \gamma_{tj} \mathbf{x}_t}{\sum_t \gamma_{tj}} \tag{A.11}$$

$$\Sigma_j = \frac{\sum_t \gamma_{tj} (\mathbf{x}_t - \mu_j)(\mathbf{x}_t - \mu_j)^T}{\sum_t \gamma_{tj}} \tag{A.12}$$

The reestimation equation for the transition probabilities a_{ij} are derived from the derivative of Q and are included here for completeness:

$$\begin{aligned} \xi_t(i,j) &= P(q_t = i, q_{t+1} = j \mid \mathbf{x}, \phi) \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{x}_t) \beta_{t+1}(j)}{P(\mathbf{x} \mid \phi)} \end{aligned} \tag{A.13}$$

$$a_{ij} = \frac{\sum_t^{T-1} \xi_t(i,j)}{\sum_t^{T-1} \gamma_{tj}} \tag{A.14}$$

Bibliography

[1] A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. In *Proceedings of 13th ICPR*, Vienna, Austria, August 1996. IEEE Computer Society Press.

[2] Y. Bengio and P. Frasconi. An input output HMM architecture. In G. Tesauro, M. D. S. Touretzky, and T. K. Leen, editors, *Advances in neural information processing systems 7*, pages 427–434. MIT Press, 1995.

[3] C. M. Bishop. *Neural networks for pattern recognition*. Clarendon Press, Oxford, 1995.

[4] C. M. Bishop, M. Svensen, and C. K. I. Williams. EM optimization of latent-variable density models. In M. C. Moser D. S. Touretzky and M. E. Hasselmo, editors, *Advances in neural information processing systems 8*, pages 402–408. MIT Press, 1996.

[5] M. Black and A. Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Proc. European Conf. Comp. Vis.*, 1998.

[6] A. Blake and M. Isard. CONDENSATION - conditional density propagation for visual tracking. *Int. J. of Comp. Vis.*, 1998.

[7] A. Bobick and J. Davis. An appearance-based representation of action. In *Int. Conf. on Pattern Rec.*, volume 1, pages 307–312, August 1996.

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

[9] A. F. Bobick and A. D. Wilson. A state-based approach to the representation and recognition of gesture. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 19(12):1325–1337, 1997.

[10] M. Brand and A. Hertzmann. style machines. In *Computer Graphics: SIGGRAPH Proceedings*, 2000.

[11] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Proc. Comp. Vis. and Pattern Rec.*, 1997.

[12] C. Bregler and S. M. Omohundro. Surface learning with applications to lipreading. *Advances in neural information processing systems 6*, pages 43–50, 1994.

[13] L. Brieman. *Statistics*. Houghton Mifflin, Boston, 1973.

[14] L. W. Campbell, D. A. Becker, A. J. Azarbayejani, A. F. Bobick, and A. Pentland. Invariant features for 3-d gesture recognition. In *Second International Conference on Face and Gesture Recognition*, pages 157–162, Killington VT, 1996.

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

[16] J. Cassell. A framework for gesture generation and interpretation. In R. Cipolla and A. Pentland, editors, *Computer vision in human-machine interaction*. Cambridge University Press, in press.

[17] J. Cassell and D. McNeill. Gesture and the poetics of prose. *Poetics Today*, 12(3):375–404, 1991.

[18] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelahter. *Probabilistic networks and expert systems*. Springer Verlag, 1999.

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

[20] T. Darrell, P. Maes, B. Blumberg, and A. Pentland. A novel environment for situated vision and behavior. In *Proc. of CVPR–94 Workshop for Visual Behaviors*, pages 68–72, Seattle, Washington, June 1994.

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

[22] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. In *Proceedings of the Royal Statistical Society*, volume B-39, pages 1–38, 1977.

[23] W. T. Freeman, D. B. Anderson, P. A. Beardsley, C. N. Dodge, M. Roth, C. D. Weissman, W. S. Yerazunis, H. Kage, K. Kyuma, Y. Miyake, and K. Tanaka. computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications*, 18(3), 1998.

[24] M.J.F. Gales. maximum likelihood linear transformations for HMM-based speech recognition. CUED/F-INFENG Technial Report 291, Cambridge University Engineering Department, 1997.

[25] D. Gavrila. The visual analysis of human movement. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[26] Z. Ghahramani and M. Jordan. Factorial hidden markov models. *Machine Learning*, 29:245–275, 1997.

[27] D. Heckerman. A tutorial on learning with Bayesian networks. MSR-TR-95-06, Microsoft Research, 1996.

[28] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumiere project: bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of teh Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

[29] M. Isard and A. Blake. ICONDENSATION: unifying low-level and high-level tracking in a stochastic framework. In *Proc. European Conf. Comp. Vis.*, pages 893–908, 1998.

[30] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Proc. Int. Conf. Comp. Vis.*, 1998.

[31] Y. Ivanov, B. Blumberg, and A. Pentland. Em for perceptual coding and reinforcement learning tasks. In *Symposium on Intelligent Robotic Systems 2000*, pages 93–100, Reading, UK, 2000.

[32] Y. Ivanov and A. Bobick. Recognition of multi-agent interaction in video surveillance. In *Proc. Int. Conf. Comp. Vis.*, 1999.

[33] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[34] F. V. Jensen. *An introduction to Bayesian networks*. Springer, New York, 1996.

[35] M. P. Johnson, A. Wilson, B. Blumberg, C. Kline, and A. Bobick. Sympathetic interfaces: using a plush toy to direct synthetic classes. In *SIGCHI'99*, pages 152–158, Pittsburgh, 1999. ACM press.

[36] N. Jojic and B. Frey. Topographic transformation as a discrete latent variable. In T. Lean S. Solla and K. Muller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[37] N. Jojic, N. Pretrovic, B. Frey, and T. Huang. Transformed hidden Markov models: estimating mixture models of images and inferring spatial transformations in video sequences. In *Proc. Comp. Vis. and Pattern Rec.*, 2000.

[38] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. In M. Jordan, editor, *Learning in graphical models*. Kluwer Academic Press, 1998.

[39] M. I. Jordan. AAAI tutorial: graphical models and variational approximation. In *AAAI*, 1998.

[40] R.E. Kahn and M.J. Swain. Understanding people pointing: The Perseus system. In *Proc. IEEE Int'l. Symp. on Comp. Vis.*, pages 569–574, Coral Gables, Florida, November 1995.

[41] A. Kendon. How gestures can become like words. In F. Poyatos, editor, *Cross-cultural perspectives in nonverbal communication*, New York, 1988. C.J. Hogrefe.

[42] J. Lasseter. Luxo jr. 1986.

[43] J. Lasseter. principles of traditional animation applied to 3D computer animation. In *Computer Graphics: SIGGRAPH Proceedings*, 1997.

[44] S. Lauritzen. *Graphical Models*. Oxford, 1996.

[45] S. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 15, 1989.

[46] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley, New York, 1997.

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

[48] D. Moore, I. Essa, and M. Hayes III. Exploiting human actions and object context for recognition tasks. In *Proc. Comp. Vis. and Pattern Rec.*, 1999.

[49] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int. J. of Comp. Vis.*, 14:5–24, 1995.

[50] K. Murphy. Fitting a constrained conditional gaussian distribution. U.c. berkeley technical report, University of California at Berkeley, 1998.

[51] K. Murphy. Inference and learning in hybrid Bayesian networks. U.c. berkeley technical report csd-98-990, University of California at Berkeley, 1998.

[52] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. In M. Jordan, editor, *Learning in graphical models*. Kluwer Academic Press, 1998.

[53] D. A. Norman. *The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution*. MIT Press, Cambridge, Massachusetts, 1998.

[54] N. Oliver, S. Pentland, and F. Berard. A real-time lips and face tracker with facial expression recognition. *Proc. Comp. Vis. and Pattern Rec.*, 1997.

[55] S. M. Omohundro. Family discovery. In D. S. Touretzky, M. C. Moser, and M. E. Hasselmo, editors, *Advances in neural information processing systems 8*, pages 402–408. MIT Press, 1996.

[56] V. Pavlovic, J. Rehg, T. Cham, and K. Murphy. A dynamic Bayesian network approach to figure tracking using learned dynamic models. In *Proc. Int. Conf. Comp. Vis.*, 1999.

[57] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[58] A. Pentland and A. Liu. Towards augmented control systems. In *IEEE Intelligent Vehicles 95*, pages 350–355, 1995.

[59] H. Poizner, E. S. Klima, U. Bellugi, and R. B. Livingston. Motion analysis of grammatical processes in a visual-gestural language. In *ACM SIGGRAPH/SIGART Interdisciplinary Workshop, Motion: Representation and Perception*, pages 148–171, Toronto, April 1983.

[60] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical recipes in C*. Cambridge University Press, Cambridge, 1991.

[61] 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.

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

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

[64] 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.

[65] P. Smyth, D. Heckerman, and M. Jordan. Probabalistic independence networks for hidden Markov probability models. *Neural Computation*, 9(2), 1997.

[66] 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.

[67] R. S. Sutton and A. G. Barto. *reinforcement learning: an introduction*. MIT Press, 1998.

[68] J. Tenenbaum and W. Freeman. Separating style and content. In *Advances in neural information processing systems 9*, 1997.

[69] F. Thomas and O. Johnston. *Disney animation: the illusion of life*. Abbeville Press, New York, 1981.

[70] K. Tuite. The production of gesture. *Semiotica*, 93-1/2:83–105, 1993.

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

[72] A. Wilson and A. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 21(9):884–900, 1999.

[73] A. D. Wilson. Luxomatic: Computer vision for puppeteering. MIT Media Lab Perceptual Computing Group Technical Report 512, Massachusetts Institute of Technology, 1999. Available at http://www-white.media.mit.edu/vismod.

[74] A. D. Wilson and A. F. Bobick. Learning visual behavior for gesture analysis. In *Proc. IEEE Int'l. Symp. on Comp. Vis.*, Coral Gables, Florida, November 1995.

[75] A. D. Wilson and A. F. Bobick. Nonlinear PHMMs for the interpretation of parameterized gesture. *Proc. Comp. Vis. and Pattern Rec.*, 1998.

[76] A. D. Wilson and A. F. Bobick. Recognition and interpretation of parametric gesture. *Proc. Int. Conf. Comp. Vis.*, pages 329–336, 1998.

[77] A. D. Wilson and A. F. Bobick. Realtime online adaptive gesture recognition. In *International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 1999.

[78] A. D. Wilson and A. F. Bobick. Realtime online adaptive gesture recognition. In *Int. Conf. on Pattern Rec.*, 2000.

[79] A. D. Wilson, A. F. Bobick, and J. Cassell. Recovering the temporal structure of natural gesture. In *Second International Conference on Face and Gesture Recognition*, pages 66–71, Killington VT, 1996.

[80] A. D. Wilson, A. F. Bobick, and J. Cassell. Temporal classification of natural gesture and application to video coding. *Proc. Comp. Vis. and Pattern Rec.*, pages 948–954, 1997.

[81] A. D. Wilson and A.F. Bobick. Using hidden Markov models to model and recognize gesture under variation. *International Journal on Pattern Recognition and Artificial Intelligence Special Issue on Hidden Markov Models in Computer Vision*, in press.

[82] C. Wren and A. P. Pentland. Dynaman: recursive modeling of human motion. *Image and Vision Computing*, to appear.

[83] Y. Yacoob and M. J. Black. Parameterized modeling and recognition of activities. *Computer Vision and Image Understanding*, 73(2):232–247, 1999.

[84] 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.