Hidden Markov Models for Modeling and Recognizing Gesture Under Variation

Andrew D. Wilson
Vision and Modeling Group
MIT Media Laboratory
20 Ames St., Cambridge, MA 02139
drew@media.mit.edu

Aaron F. Bobick
College of Computing
Georgia Institute of Technology
801 Atlantic Ave., Atlanta, GA 30332
afb@cc.gatech.edu

Abstract

Conventional application of hidden Markov models to the task of recognizing human gesture may suffer from multiple sources of systematic variation in the sensor outputs. We present two frameworks based on hidden Markov models which are designed to model and recognize gestures that vary in systematic ways. In the first, the systematic variation is assumed to be communicative in nature, and the input gesture is assumed to belong to gesture *family*. The variation across the family is modeled explicitly by the parametric hidden Markov model (PHMM). In the second framework, variation in the signal is overcome by relying on online learning rather than conventional offline, batch learning.

Keywords: hidden Markov models, gesture recognition, computer vision

1 Introduction

Hidden Markov models (HMMs) are a popular technique for recognizing human gesture in a variety of applications and sensor configurations. Chief among their benefits is the fact that gesture models may be trained automatically from a series of examples of the gesture class, and the fact that the trained models encode the variation present in the set of examples.

On the surface, applying HMMs to the task of recognizing gestures from video input is no different than applying HMMs to any other kind of signal: features are computed at each time step, example sequences of the features are stored, and models trained on the examples are later matched to a novel input feature sequence.

A naive application of HMMs to recognize gestures from video might treat the collection of image pixel values at each time step as the feature vector. Besides being computationally daunting, this approach suffers from the fact that it would take a great many examples to span the space of variation present in the raw appearance of a human gesture, particularly if multiple viewing conditions and multiple users are considered.

The standard approach is thus to compute some features from each image first and treat the concatenation of these features as the feature vector passed to the gesture model. For example, if the environment is sufficiently controlled, it may be possible to track the head and hands of the user to construct a compact feature vector useful in gesture recognition. Such an approach is ultimately also affected by variation in visual appearance: tracking may fail in unpredictable ways if the situation is novel enough. The hope is that features may be chosen such that the resulting system works in a wide enough set of circumstances to be useful.

Another class of variation that gesture recognition systems must confront is the variation in the execution of gestures themselves. Even within-subject variation is common in most gestures. Some of this variation is communicative in nature, in which case we might like to extract some well-defined meaning, and some may be considered noise.

In this article we present two frameworks that deal with these kinds of variation in different ways. Each differs from the usual approach of training standard HMMs on pre-defined features computed on the image sequence, making different tradeoffs based on features extracted from the image, assumptions about the nature of the gesture to be recognized, and assumptions about the environment.

In the first, we present the parametric HMM (PHMM), which models *families* of input signals that vary smoothly. PHMMs are appropriate for extracting meaningful, systematic variation from a gesture signal.

In the second framework, we exploit online learning in recognizing gesture from video. Online learning avoids the dual problems of feature selection and collecting a sufficient number of example gestures, and instead relies on features derived from application context. By adapting to the situation on-the-fly, the system is able to overcome variation in the gesture signal that might otherwise be modeled as noise by the usual training/testing paradigm.

2 Modeling Gesture Families

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 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.*

Here we 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

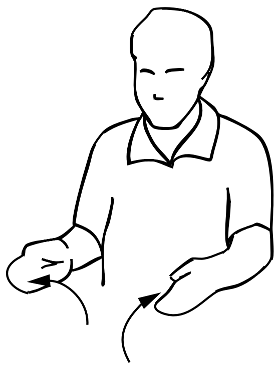


Figure 1: The gesture that accompanies the speech "I caught a fish. It was *this* big." In its entirety, the gesture consists of a preparation phase in which the hands are brought into the gesture space, a stroke phase (depicted by the illustration) which co-occurs with the word "this" and finally a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

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. 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.

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.

2.1 Motivation and Prior Work

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 [10] applied DTW to match image template correlation scores against models to recognize hand gestures from video. In previous work [4], 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 [27] first used HMMs in vision to recognize tennis strokes. Schlenzig, Hunter and Jain [20] used HMMs and a rotation-invariant image representation to recognize hand gestures from video. Starner and Pentland [21] applied HMMs to recognize ASL sentences, and Campbell et al. [7] 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 [25] 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 [14, 9]. Campbell and Bobick [8] 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 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 [16].

One approach is to explicitly model the space of variation exhibited by a class of signals. In [24], 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 [26] as well as Bobick and Davis [5] 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. They do not address parameter extraction.

Murase and Nayar [15] 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.

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 [11] 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.

2.3 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 [18] 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 $\boldsymbol{\Sigma}_j$.[1]

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 $\boldsymbol{\theta}$ 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 [26] 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:

$$\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}) = W_j \boldsymbol{\theta} + \bar{\boldsymbol{\mu}}_j \tag{1}$$

$$P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t, \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}), \Sigma_j) \tag{2}$$

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.

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

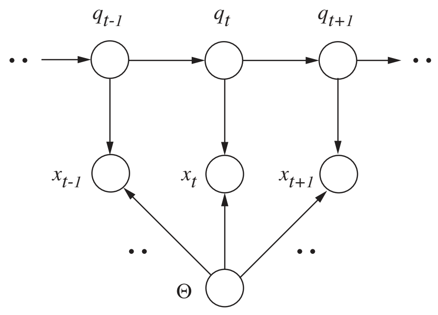


Figure 2: Bayes network showing the conditional dependencies of the PHMM.

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 shows the PHMM architecture as a Bayes network. The diagram 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 to estimate the additional parameters. An appendix provides a brief description of the Baum-Welch algorithm; for a comprehensive discussion see [18].

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:

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')} \tag{3}$$

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:

$$Z_j \equiv \begin{bmatrix} W_j & \bar{\boldsymbol{\mu}}_j \end{bmatrix} \quad \Omega_k \equiv \begin{bmatrix} \boldsymbol{\theta}_k \\ 1 \end{bmatrix} \tag{4}$$

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 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 3:

$$\begin{aligned}
\frac{\partial Q}{\partial Z_j} &= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \frac{\partial}{\partial Z_j} \left(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k)\right)^T \Sigma_j^{-1} \left(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k)\right) & (5)\\
&= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \frac{\partial}{\partial Z_j} \left[\mathbf{x}_{kt}{}^T \Sigma_j^{-1} \mathbf{x}_{kt} - 2\hat{\boldsymbol{\mu}}_j^T \Sigma_j^{-1} \mathbf{x}_{kt} + \hat{\boldsymbol{\mu}}_j^T \Sigma_j^{-1} \hat{\boldsymbol{\mu}}_j\right] \\
&= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \left[-2 \frac{\partial}{\partial Z_j} \left(Z_j \Omega_k\right)^T \Sigma_j^{-1} \mathbf{x}_{kt} + \frac{\partial}{\partial Z_j} \left(Z_j \Omega_k\right)^T \Sigma_j^{-1} Z_j \Omega_k\right] \\
&= -\frac{1}{2} \sum_k \sum_t \gamma_{ktj} \left[-2 \frac{\partial}{\partial Z_j} \Omega_k{}^T Z_j{}^T \Sigma_j^{-1} \mathbf{x}_{kt} + \frac{\partial}{\partial Z_j} \Omega_k{}^T \left(Z_j{}^T \Sigma_j^{-1} Z_j\right) \Omega_k\right] \\
&= \Sigma_j^{-1} \sum_k \sum_t \gamma_{ktj} \left[\mathbf{x}_{kt} \Omega_k{}^T - Z_j \Omega_k \Omega_k{}^T\right] & (6)
\end{aligned}$$

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:

$$Z_j = \left[\sum_{k,t} \gamma_{ktj} \mathbf{x}_{kt} \Omega_k^T\right] \left[\sum_{k,t} \gamma_{ktj} \Omega_k \Omega_k^T\right]^{-1} \tag{7}$$

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

$$\Sigma_j = \sum_{k,t} \frac{\gamma_{ktj}}{\sum_t \gamma_{ktj}} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))^T \tag{8}$$

as is the matrix of transition probabilities [18] (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 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:

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} = \sum_t \sum_j \gamma_{tj} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{9}$$

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

$$\boldsymbol{\theta} = \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} W_j \right]^{-1} \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} (\mathbf{x}_t - \bar{\boldsymbol{\mu}}_j) \right] \tag{10}$$

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 2.5.1 in some cases it may be possible to classify the sequence by the value of $\boldsymbol{\theta}$ as determined by a single PHMM.

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

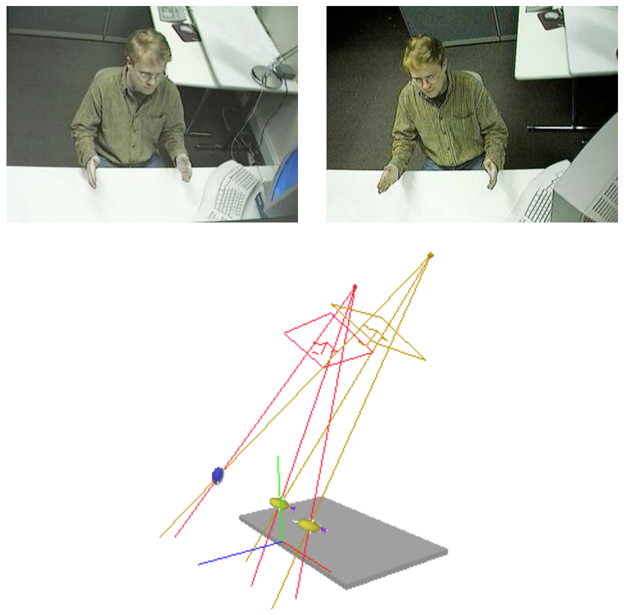


Figure 3: The Stereo Interactive Virtual Environment (STIVE) computer vision system used to collect data in section 2.5. Using flesh tracking techniques, STIVE computes the three-dimensional position of the head and hands at a frame rate of about 20Hz. We used only the position of the hands for the first two experiments.

---

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 $\boldsymbol{\theta}$.

2.5 Experiment 1: Size gesture

To test the ability of the parametric HMM to learn the parameterization, thirty examples of the type depicted in Figure 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 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 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

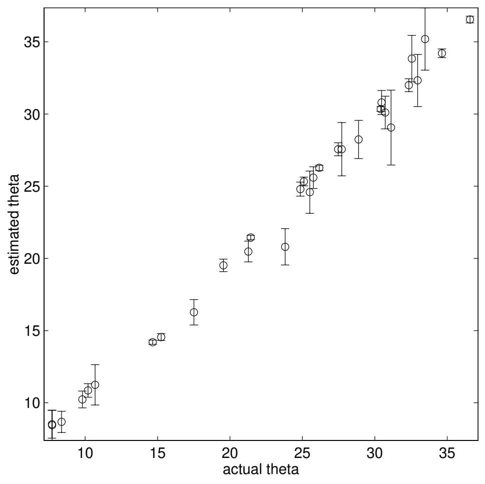


Figure 4: Parameter estimation results for the size gesture. Fifty random choices of the test and training sets were used to compute mean and standard deviation (error bars) on all examples. The HMM was retrained for each choice of test and training set.

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" [18]). 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 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 2.5 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 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 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.

2.5.1 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.

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

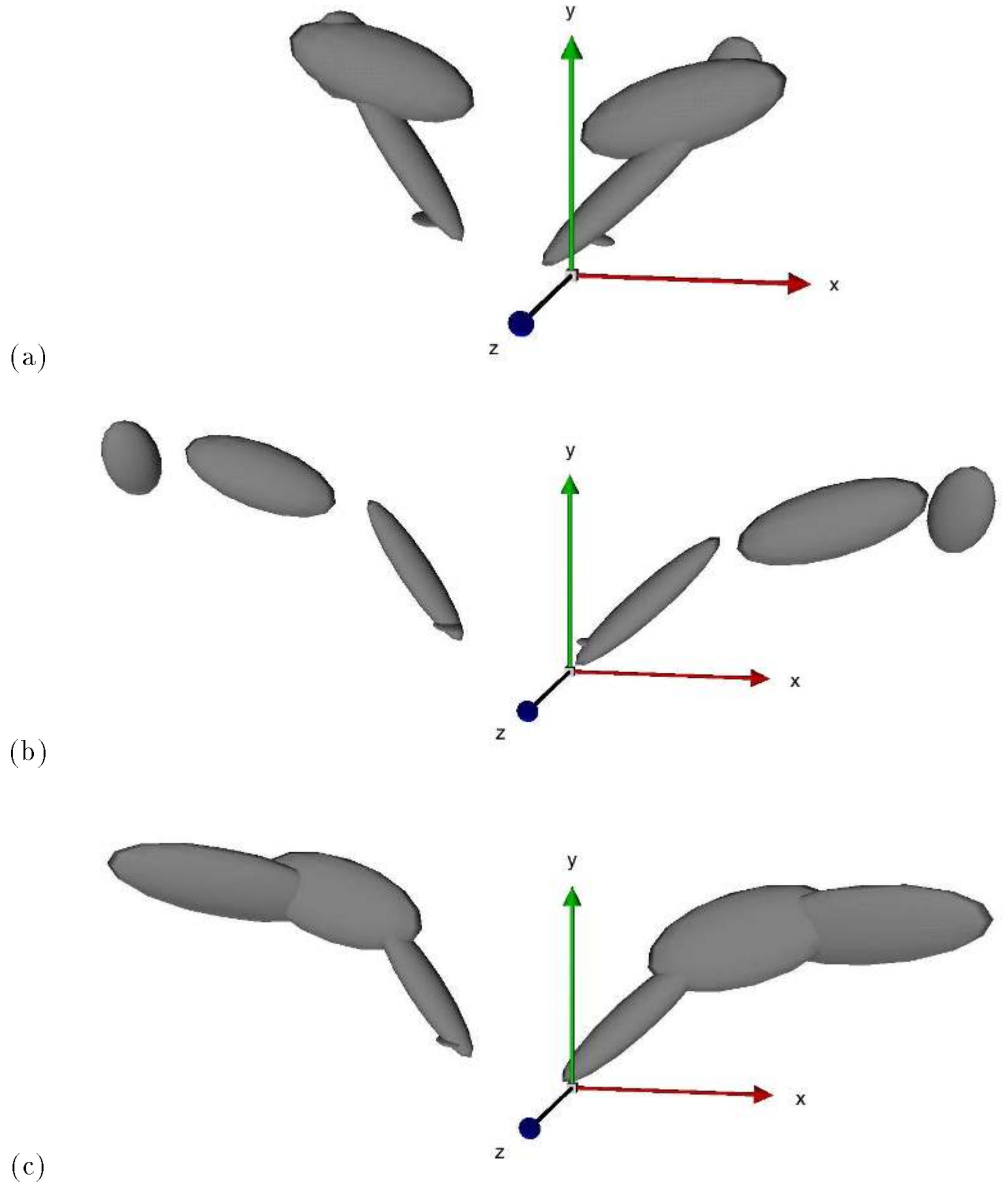


Figure 5: The state output density of the two-handed fish-size gesture. Each corresponds to either left or right hand position at a state (for clarity, only the first four states are shown); (a) PHMM, $\boldsymbol{\theta} = 19.0$, (b) PHMM, $\boldsymbol{\theta} = 45.0$, (c) HMM. The ellipsoid shapes 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`.

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 *point* 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 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 6: Confusion matrices for the *point* and *direct* gesture models. Row headings are the ground truth classifcations.

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 7a and 7b display the 1.0σ ellipsoids of the Gaussian densities of the states of the PHMM; 7a is for $\theta = (15^\circ, 15^\circ)$, 7b is for $\boldsymbol{\theta} = (15^\circ, -15^\circ)$. Notice how the position of the means has shifted. Figure 7c and d display the 1.0σ ellipsoids for the states of the conventional HMM.

Note that in Figures 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.

2.5.2 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.

2.5.3 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

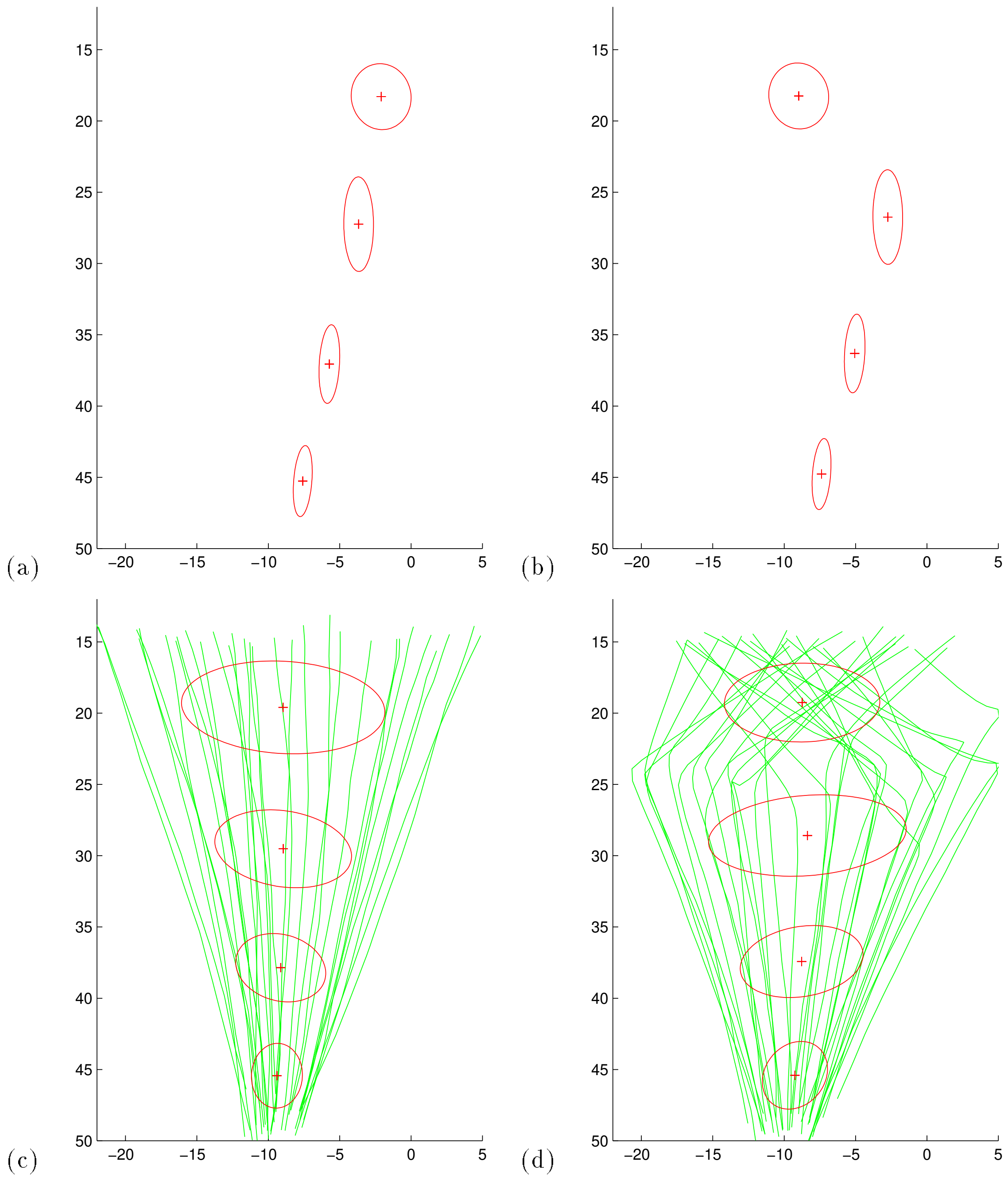


Figure 7: The state output densities of the *point* and *direct* gesture models. (a) PHMM $\theta = (15°, 15°)$, (b) PHMM $\theta = (15°, -15°)$, (c) *point* HMM with training set sequences shown, (c) *direct* HMM with training set sequences.

Figure 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 **??** 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.

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 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 to filter potential noise, it is analogous to techniques used by previous researchers (for example, [13]) for real-world applications.

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 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 errors discussed in the previous section. Note that although it would be possible to set a log

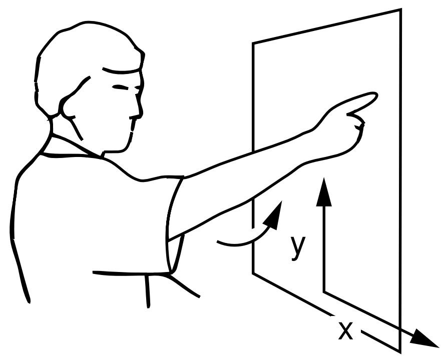


Figure 8: The point gesture used in section 2.5.2. The movement is parameterized by the coordinates of the target $\boldsymbol{\theta} = (x, y)$ within a plane in front of the user. The gesture consists of a preparation phase, a stroke phase (shown here) and a retraction.

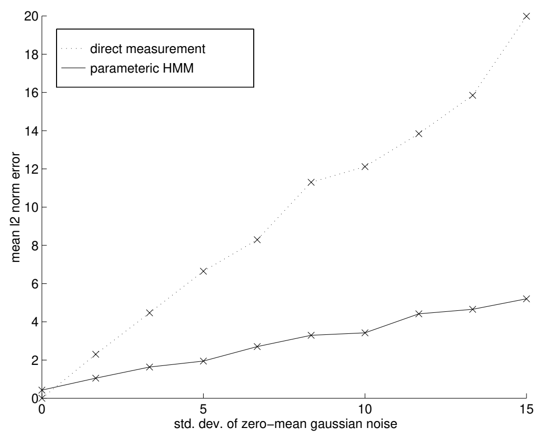
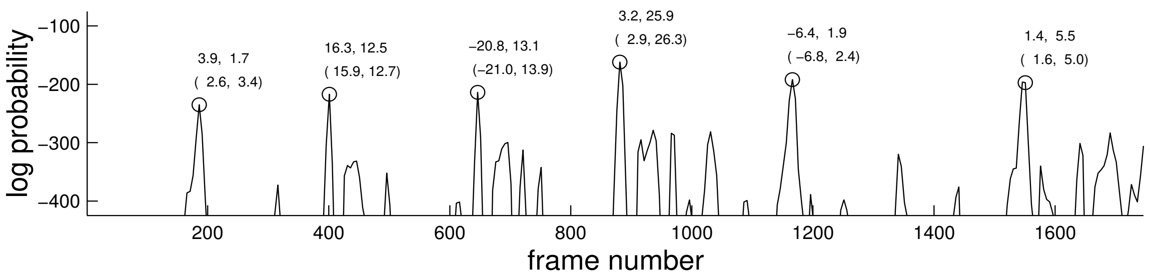
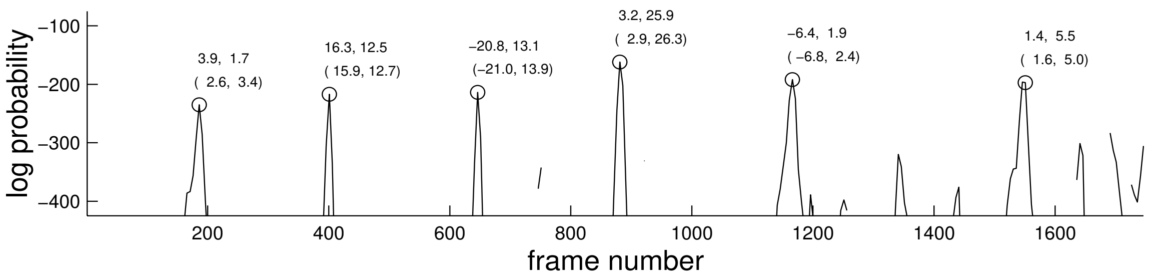


Figure 9: Average error over the entire pointing test set as a function of noise. The value of $\boldsymbol{\theta}$ was estimated by an direct measurement and by a parametric HMM retrained for each noise condition. The average error was computed by comparing the estimate of $\boldsymbol{\theta}$ to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



(a)



(b)

Figure 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.

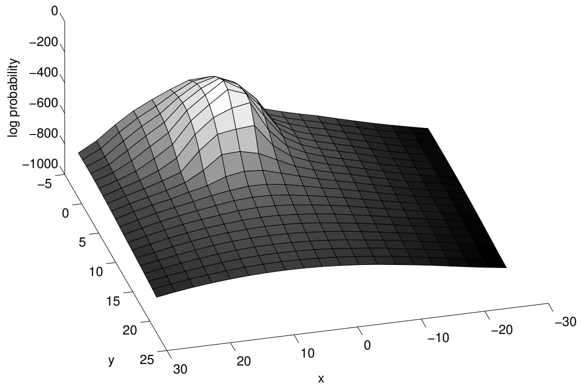


Figure 11: Log probability as a function of $\boldsymbol{\theta} = (x, y)$ for a pointing test sequence. The smoothness of the surface makes it possible to use iterative optimization techniques such as EM to find the maximum.

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 [6]. 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 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.

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 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.[2]

2.6 Nonlinear PHMMs

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.

[2]Given the graphical model equivalent in Figure 2 it is possible to exactly solve the for the best value of θ using the junction tree algorithm [12] and conditional gaussian potentials [17], which model distributions of the form of equations 1 and 2. In this work we have opted however to stay within the more specialized HMM framework.

More generally, the parameterization may not be linear, and there may be no linearization of the parameterization available. In this case more general modeling technique is in order. 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 [23] we extend the PHMM framework to use neural networks and GEM algorithms to model non-linear dependencies.

3 Watch and Learn: Learning Gestures Online

The PHMM models gesture families by explicitly modeling the variation of the gesture. Recognition then involves determining where in the modeled space of variation the input lies. In this section we consider a different but related approach, whereby rather than explicitly modeling the variation from a set of examples, the system instead combines information from the application context with the input to fix free parameters.

The result is a system that is able to model classes of gesture that undergo significant variation from session to session. Online learning is used to adapt a model with free parameters to a particular runtime situation.

3.1 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 may be preferable to adapt existing models to the situation at hand. Preliminary work in developing a system that learns gestures in an online manner is presented. 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. Gesture is used in an interactive context for controlling interaction events. 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.

3.2 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 (HMMs) are a popular choice to model gestures because they are easily trained and are efficient in the testing phase [19].

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 typical of the situations in which the training sequences were collected. A systematic bias present in the testing conditions with respect to the training data conditions may confuse the classification. For example, if the input features are not translation-invariant and the user has moved a bit, the trained models may no longer be appropriate.

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.

3.3 Temporal Structure, Context and Control

The idea of the online adaptive gesture learning algorithm 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 12. 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”.

If our only concern is recognition then such a transposition is unimportant; the likelihood

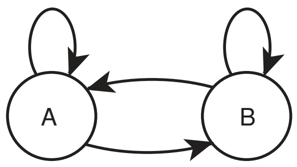


Figure 12: The simplest Markov model appropriate for a periodic signal. Given equal transition probabilities from state A to B, the Markov model is symmetric in A and B. Without contextual information, alignment of a signal to this Markov model would yield one of two possible equivalent assignments of semantics to A and B.

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 3.7 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 no longer required to perform an alignment which avoids the original ambiguity.

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

3.4 Previous Work

In [25] we use an approach similar to that presented in this paper 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 [14]. 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 appearance of the rest-states was determined only after considering the input data; the EM-like process of determining the rest state appearance models relied on the assumption that the subject's behavior was reasonably well-modeled by the Markov model.

Once rest-states are identified, it is possible to use the Markov model to parse the sequence into rest periods, biphasic, and triphasic gestures. Figure 13 illustrates typical triphasic gestures found by the system. This classification might be used for an intelligent coding of the sequence. For example, triphasic gestures, which are thought to be more communicative in their particular form than beat gestures, might be coded in detail.

Figure 13: Keyframes from four different triphasic gestures correctly labeled automatically by the system.

The present work similarly fits a hand-coded Markov model with a block of video input in order to determine the value of a set of free parameters, but differs in that this process happens in realtime in an online fashion.

3.5 Learning Algorithm

3.5.1 Expectation-Maximization Algorithm for Hidden Markov Models

A hidden Markov model uses the topology of a 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 12. 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:

$$\mu_j = \frac{\sum\limits_t \gamma_{tj}\mathbf{x}_t}{\sum\limits_t \gamma_{tj}} \tag{11}$$

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

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. These equations are derived in the appendix.

3.5.2 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 3.3, 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 update the state output distribution parameters, the biased γ's may be thought of as an attention focusing mechanism. We may bias γ_{tj} as a way to incorporate exterior knowledge to influence this focus of attention and thus guide the learning. 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 probabilities derived from application context.[3]

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

$$\omega_{tj} = P(q_t = j \mid \Omega) \tag{13}$$

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:

$$\Gamma_{tj} = \rho_j \gamma_{tj} + (1 - \rho_j)\omega_{tj} \tag{14}$$

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

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

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 b_{jt} takes the form of Gaussian distributions, starting with large covariances to represent uncertainty brings b_{jt} 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. 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:

$$\mu_j' = (1-\alpha)\mu_j + \alpha \sum_t \Gamma_{tj}\mathbf{x}_t \tag{15}$$

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.

3.6 Images as Input

3.6.1 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 body-centric image at 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-based tracking algorithm models the spatial distribution of the silhouette pixels over the image as a Gaussian with fixed covariance.

$$h_{x,y} \quad = \quad \frac{1}{\sqrt{2\pi \mid \Sigma \mid}} e^{-\frac{1}{2}\left(\left[\mathrm{x}\ \mathrm{y}\right]^T - \mathbf{c}\right)^T \Sigma^{-1} \left(\left[\mathrm{x}\ \mathrm{y}\right]^T - \mathbf{c}\right)} \tag{16}$$

$$\mathbf{c}' = \sum_{I_{x,y}>b} h_{x,y}[\mathrm{x}\ \mathrm{y}]^T \tag{17}$$

where $I_{x,y}$ is the value of the pixel at (x, y), $\mathbf{c}$ is the vector of tracked coordinates from the previous time step, b is threshold for binarizing the image and Σ is the constant 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 of the Gaussian is assigned a random value at each successive frame until the likelihood is above the threshold. Otherwise, the mean of the Gaussian is updated to reflect the translation of the silhouette.

3.6.2 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 buffered in memory. These form the observation sequence 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:

$$b_{jt} = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2\sigma_j^2 XY}(\mathbf{x}_t - \mu_j)^T(\mathbf{x}_t - \mu_j)} \tag{18}$$

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

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 11, here involving the weighted sum of the images $\mathbf{x}_t$ in the image buffer. The update of σ_j is similarly a weighted sum:

$$\sigma_j = \frac{\sum_t \Gamma_{tj}(\mathbf{x}_t - \mu_j)^T(\mathbf{x}_t - \mu_j)}{\sum_t \Gamma_{tj}} \tag{19}$$

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

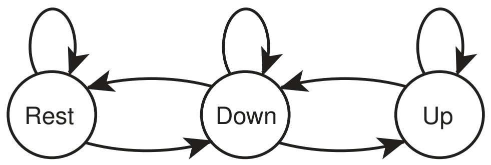


Figure 14: The Markov model used to represent the temporal pattern of a beat in the *Watch and Learn* system applied to the simple conducting scenario.

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.

3.7 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 simple beat gesture may be learned by adaptive online learning.

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 14). 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. Figures 15 and 16 show a typical set of appearance models for the beat Markov model during the adaptation process. Figure 15 shows the appearance models in the middle of the adaptation process, Figure 16 after the adaptation is complete. The system continually adjusts these states to reflect the subtle changes in the way the user executes the gesture from instance to instance.

Figures 17 and 18 show γ_{tj} and ω_{tj} over a window of 64 frames (about 2 seconds) of video. Figure 17 is taken during adaptation, Figure 18 after adaptation. Note that during adaptation,

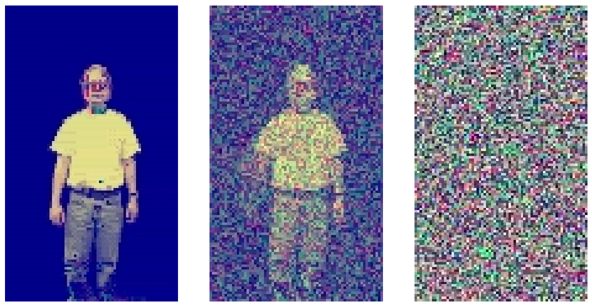


Figure 15: The appearance models (images) associated with each state of the beat HMM during online adaptation. When the algorithm is started, the pixels values of the images are randomized. At this point, the appearance model of the rest state has been trained, and the appearance model of the "up" state is being adapted.

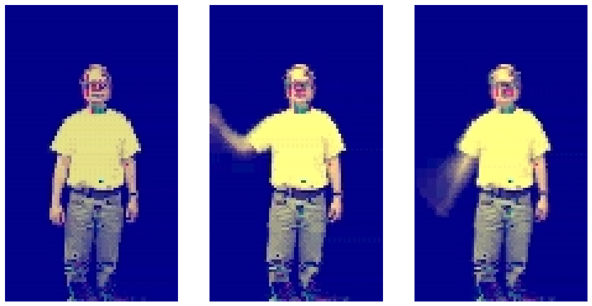


Figure 16: The appearance models (images) associated with each state of the beat HMM after online adaptation.

the probabilities from the application ω_{tj} (square wave) guide the state membership γ_{tj}.

QuickTime video of the camera input and the learning algorithm are located at `http://www.media.mit.edu/~drew/watchandlearn`.

Figure 19 shows appearance models learned in a second session with the system. Note that even while many of the viewing circumstances such as pose, distance to camera, and clothing have changed, the adapted appearance models have the correct semantics.

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 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

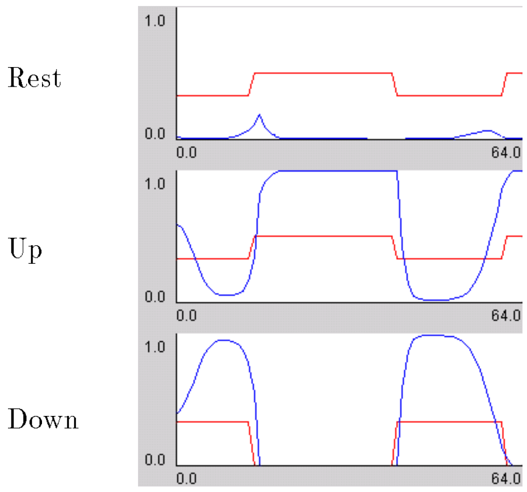


Figure 17: Plots of γ_{tj} and ω_{tj} (square wave) for the beat gesture, during online adaptation.

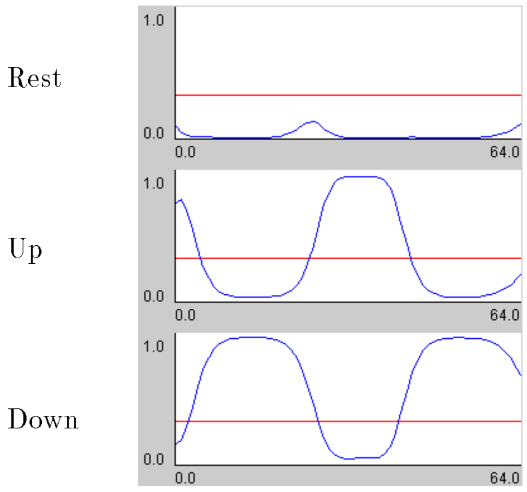


Figure 18: Plots of γ_{tj} for the beat gesture, after online adaptation.

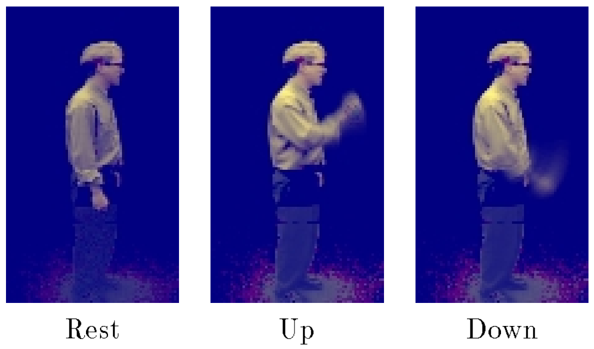


Figure 19: The appearance models (images) associated with each state of the beat HMM after online adaptation during a second session.

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.

3.8 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 20).

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

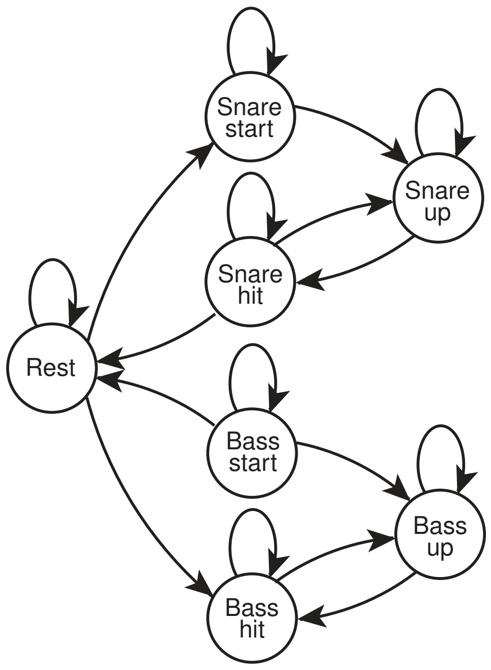


Figure 20: The Markov model used for a drum-set configuration of *Watch and Learn*. MIDI events are generated when the Markov model enters the "Snare hit" and "Bass hit" states. "Snare start" and "Bass start" states capture the preparation of the gesture and do not generate MIDI events. The appearance 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.

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 18). 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 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.

4 Conclusion

We present two frameworks for tackling the problem of recognizing gestures that undergo significant systematic variation.

First, 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.

As a gesture or activity recognition technique, the PHMM 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. The parameterized technique presented is domain-independent and is applicable to any sequence parsing problem where some context or style ([22]) spans an entire sequence.

Second, 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. The *Watch and Learn* computer vision system demonstrates the online adaptive learning approach to learning gesture models.

Both techniques use representations encoding free parameters that are fixed during runtime. With the PHMM the parameterization is fixed over each gesture that is a member of some gesture family. In the case of Watch and Learn, this parameterization is fixed once per session.

The PHMM has currently been tested with tracked points and blobs as input, while the Watch and Learn system operates on whole (masked) images. Operating on whole images has the advantage that no assumptions about the nature of tracked objects are required, and the features may be highly diagnostic within a particular session. Conversely, starting from tracked point or blobs as features has the advantage that such features are more likely to generalize to a future session than the image-based features. We are currently exploring ways to combine the two approaches.

A Appendix: Expectation-maximization algorithm for hidden Markov models

In this section we derive equation 3 from the expectation-maximization (EM) algorithm [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:

$$\begin{aligned} Q(\phi' \mid \phi) &= \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log P(\mathbf{x}, \mathbf{q}, \phi')\right] && (20) \\ &= \sum_{\mathbf{q}} P(\mathbf{q} \mid \mathbf{x}, \phi) \log P(\mathbf{x}, \mathbf{q}, \phi') && (21) \end{aligned}$$

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 in this paper, 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 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:

$$Q(\phi' \mid \phi) = \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log \prod_t a_{q_{t-1}q_t} P(\mathbf{x}_t \mid q_t, \phi')\right] \qquad (22)$$

The expectation is carried out using the Markov property:

$$\begin{aligned} Q(\phi' \mid \phi) &= \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\sum_t \log a_{q_{t-1}q_t} + \sum_t \log P(\mathbf{x}_t \mid q_t, \phi')\right] \\ &= \sum_t \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log a_{q_{t-1}q_t} + \log P(\mathbf{x}_t \mid q_t, \phi')\right] \\ &= \sum_{t,j} P(q_t = j \mid \mathbf{x}, \phi)\left[\sum_i P(q_{t-1} = i \mid \mathbf{x}, \phi) \log a_{ij} + \log P(\mathbf{x}_t \mid q_t = j, \phi')\right] && (23) \end{aligned}$$

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

$$\begin{aligned}
\alpha_1(j) &= \pi_j b_j(\mathbf{x_t}) && (24)\\
\alpha_t(j) &= \left[\sum_i \alpha_t(i) a_{ij}\right] b_j(\mathbf{x}_t) && (25)\\
\beta_T(j) &= 1 && (26)\\
\beta_t(j) &= \sum_j a_{ij} b_j(\mathbf{x}_{t+1})\beta_{t+1}(j) && (27)
\end{aligned}$$

from which γ_{tj} may be computed:

$$\gamma_{tj} = \frac{\alpha_t(j)\beta_t(j)}{P(\mathbf{x} \mid \phi)} \tag{28}$$

In the "maximization" step, we compute ϕ' to increase Q. Taking the derivative of equation 23 and writing $P(q_t = j \mid \mathbf{x}, \phi)$ as γ_{tj} we arrive at:

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')} \tag{29}$$

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:

$$\begin{aligned}
\mu_j &= \frac{\sum_t \gamma_{tj}\mathbf{x}_t}{\sum_t \gamma_{tj}} && (30)\\
\Sigma_j &= \frac{\sum_t \gamma_{tj}(\mathbf{x}_t - \mu_j)(\mathbf{x}_t - \mu_j)^T}{\sum_t \gamma_{tj}} && (31)
\end{aligned}$$

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

$$\begin{aligned}
\xi_t(i,j) &= P(q_t = i, q_{t+1} = j \mid \mathbf{x}, \phi)\\
&= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{x}_t)\beta_{t+1}(j)}{P(\mathbf{x} \mid \phi)} && (32)\\
a_{ij} &= \frac{\sum_t^{T-1} \xi_t(i,j)}{\sum_t^{T-1} \gamma_{tj}} && (33)
\end{aligned}$$

References

[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] 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.

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

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

[7] 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.

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

[9] J. Cassell and D. McNeill. Gesture and the poetics of prose. *Poetics Today*, 12(3):375–404, 1991.

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

[11] M.J.F. Gales. maximum likelihood linear transformations for HMM-based speech recognition. CUED/F-INFENG Technial Report 291, Cambridge University Engineering Department, 1997.

[12] F. V. Jensen. *An introductions to Bayesian networks.* Springer, New York, 1996.

[13] 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.

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

[15] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int. J. of Comp. Vis.*, 14:5–24, 1995.

[16] 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.

[17] S. L. Lauritzen R. G. Cowell, A. P. Dawid and D. J. Spiegelahter. *Probabilistic networks and expert systems.* Springer Verlag, 1999.

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

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

[20] 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.

[21] 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.

[22] J. Tenenbaum and W. Freeman. Separating style and content. In *Advances in neural information processing systems 9*, 1997.

[23] A. Wilson and A. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 21(9):884–900, 1999.

[24] 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.

[25] 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.

[26] Y. Yacoob and M. J. Black. Parameterized modeling and recognition of activities. *Computer Vision and Image Understanding*, 73(2):232–247, 1999.

[27] 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.