Parametric Hidden Markov Models for Gesture Recognition

Andrew D. Wilson, *Student Member*, *IEEE Computer Society*, and
Aaron F. Bobick, *Member*, *IEEE Computer Society*

Abstract—A new method for the representation, recognition, and interpretation of parameterized gesture is presented. By parameterized gesture we mean gestures that exhibit a systematic spatial variation; one example is a point gesture where the relevant parameter is the two-dimensional direction. Our approach is to extend the standard hidden Markov model method of gesture recognition by including a global parametric variation in the output probabilities of the HMM states. Using a linear model of dependence, we formulate an expectation-maximization (EM) method for training the parametric HMM. During testing, a similar EM algorithm simultaneously maximizes the output likelihood of the PHMM for the given sequence and estimates the quantifying parameters. Using visually derived and directly measured three-dimensional hand position measurements as input, we present results that demonstrate the recognition superiority of the PHMM over standard HMM techniques, as well as greater robustness in parameter estimation with respect to noise in the input features. Last, we extend the PHMM to handle arbitrary smooth (nonlinear) dependencies. The nonlinear formulation requires the use of a generalized expectation-maximization (GEM) algorithm for both training and the simultaneous recognition of the gesture and estimation of the value of the parameter. We present results on a pointing gesture, where the nonlinear approach permits the natural spherical coordinate parameterization of pointing direction.

Index Terms—Gesture recognition, hidden Markov models, expectation-maximization algorithm, time-series modeling, computer vision.

1 Introduction

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

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

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

Techniques that use fixed prototypes for matching are not well-suited to modeling movements that exhibit such meaningful variation. In this paper, 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 three-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

- *A.D. Wilson is with the Vision and Modeling Group, MIT Media Laboratory, 20 Ames St., Cambridge, MA 02139.
E-mail: drew@media.mit.edu.*
- *A.F. Bobick is with the College of Computing, Georgia Institute of Technology, Atlanta, GA .*

*Manuscript received 9 June 1998; revised 25 May 1999.
Recommended for acceptance by M. Black.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107686.*

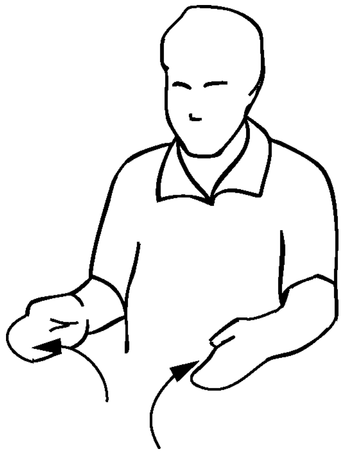


Fig. 1. The gesture that accompanies the speech "I caught a fish. It was *this* big." In its entirety, the gesture consists of a preparation phase in which the hands are brought into the gesture space, a stroke phase (depicted by the illustration) which co-occurs with the word "this" and, finally, a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

improved robustness in estimating the quantifying parameter with respect to noise in the input features.

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

2 Motivation and Prior Work

2.1 Using HMMs in Gesture Recognition

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

A typical problem with these techniques is determining when the gesture began without classifying each subsequence up to the current time. One solution is to use dynamic programming to match the sequence against a model from all possible starting times of the gesture to the current time. The best starting time is then chosen from all possible starting times to give the best match average over the length of the gesture. Dynamic time warping (DTW) and Hidden Markov models (HMMs) are two techniques based on dynamic programming. Darrell and Pentland [12] applied DTW to match image template correlation scores against models to recognize hand gestures from video. In previous work [5], 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.

HMMs forego the construction of a prototype in exchange for an expectation/maximization method of determining a stochastic sequence of states to represent gesture. Yamato et al. [32] first used HMMs in vision to recognize tennis strokes. Schlenzig et al. [23] used HMMs and a rotation-invariant image representation to recognize hand gestures from video. Starner and Pentland [24] applied HMMs to recognize ASL sentences, and Campbell et al. [9] 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, it is often effective to look for diagnostic features. For example, in [30], 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 [18], [11]. Campbell and Bobick [10] 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 [21].

One approach is to explicitly model the space of variation exhibited by a class of signals. In [27], 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 [31], as well as Bobick and Davis [6], model the variation within a class of human movement using linear principal 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 [19] parameterize meaningful variation in the appearance of images by computing a representation of the nonlinear manifold of the images in an eigenspace of the images. Their work is similar to ours in that training assumes that each input feature vector is labeled with the value of the parameterization. In testing, an unknown image is projected onto the manifold and the parameterization is recovered. Their framework has been used, for example, to recover the camera angle relative to a known object in the field of view.

Recently, there has been interest in methods that discover parameterizations in an unsupervised way (so-called *latent* parameterizations). In his "family discovery" paradigm, Omohundro [20], for example, outlines a variety of approaches to learning a nonlinear manifold in some feature space representing systematic variation. One of these techniques has been applied to the task of lip reading by Bregler and Omohundro [7]. Bishop et al. [4] have also introduced techniques to learn latent parameterizations. Their system begins with an assumption of the dimensionality of the parameterization and uses an expectation-maximization framework to compute a manifold representation. The present work is similarly concerned with modeling "families" of signals, but assumes that the parameterization is given for the training set.

Last, 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 [14] for example, examines a number of 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 Nonparametric Extensions

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

The first approach relies on our ability to come up with ad hoc methods to extract the value of the parameter of interest. For the example of the fish-size gesture presented in Fig. 1, one could design a procedure to recover the parameter: Wait until the hands are in the middle of the gesture space and have low velocity, then calculate the distance between the hands. Similar approaches are used in the ALIVE [13] and Perseus [17] systems. The typical approach of these systems is to first identify static configurations of the user's body that are diagnostic of the gesture and, then, use an unrelated method to extract the parameter of interest (for example, direction of pointing). Manually constructed ad hoc procedures are typically used to identify the diagnostic configuration, a task complicated by the requirement that this procedure work through the range of meaningful variation and also not be confused by other gestures. Perseus, for example, understands pointing gestures by detecting when the user's arm is extended. The system then finds the pointing direction by computing the line from the head to the user's hand.

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

The second approach employs multiple DTW or HMM models to cover the parameter space. Each DTW model or HMM is associated with a point in parameter space. In learning, the problem of allocating training examples labeled by a continuous variable to one of a discrete set of models is eliminated by uniting the models in a mixture of experts framework [15]. In testing, the parameter is extracted by finding the best match among the models and looking up its associated parameter value. The dependency of the movement's form on the parameter is thus removed.

The most serious objection to this approach is that, as the dimensionality of the parameter space increases, the large number of models necessary to cover the space will place unreasonable demands on the amount of training data.[1] For example, to recover a two-dimensional parameter with 4 bits of accuracy would theoretically require 256 distinct HMMs (assuming no interpolation). Furthermore, with such a set of distinct HMMs, all of the models are required to learn the same or similar dynamics (i.e., as modeled by the transition matrix in the case of HMMs) separately, increasing the amount of training data required. This can be embellished somewhat by computing the value of the parameter as the weighted average of all the models' associated parameter values, where the weights are derived from the matching process.

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

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

3 Parametric Hidden Markov Models

3.1 Defining Parameterized Gesture

Parametric HMMs explicitly model the dependence on the parameter of interest. We begin with the usual HMM formulation [22] 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 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 $\boldsymbol{\Sigma}_j$.[2]

In this paper, 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 modifies the spatial (or, more general, feature) variation and does not model 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 [31] derived a method for recognizing global temporal deformations of an activity; their method does not, however, represent the explicit spatial parameter variation.

Also, although θ is a global parameter—it affects all states—the actual effect varies state to state. Therefore, the effect of θ 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.

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 hyperplane in feature space, where d is the dimension of $\boldsymbol{\theta}$. For the example of 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 6×1, and would represent the one-dimensional hyperplane (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 3×2. 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 complete Bayesian estimate of $\boldsymbol{\theta}$, given an observed sequence we would need to specify a prior distribution on $\boldsymbol{\theta}$. In the work presented here, we assume the distribution of $\boldsymbol{\theta}$ is finite-uniform, implying that the value of the prior $P(\boldsymbol{\theta})$ for any particular $\boldsymbol{\theta}$ is either a constant or zero. We therefore can ignore it in the following derivations and simply use bounds checking during testing to make sure that the recovered $\boldsymbol{\theta}$ is plausible, as indicated by the training data.

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

For readers familiar with graphical model representations of HMMs (for example, see [3]), Fig. 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, need 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.

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 [22].

The *expectation* step of the Baum-Welch algorithm (also known as the "forward/backward" algorithm) computes

[2] Technically, there are also the initial state parameters π_j to be estimated; in this work, we use causal topologies with a unique starting state.

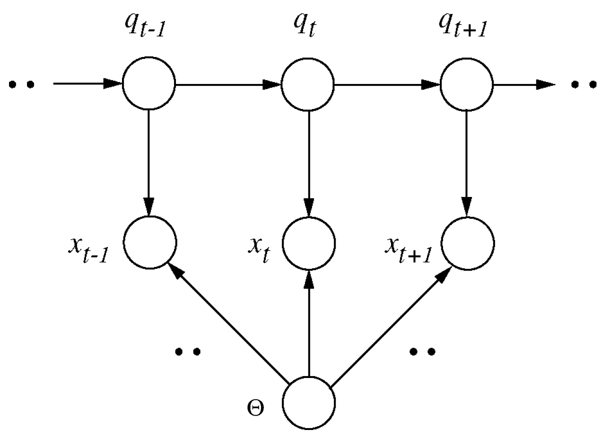


Fig. 2. Bayes network showing the conditional dependencies of the PHMM.

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:

$$Z_j \equiv \begin{bmatrix} W_j & \bar{\boldsymbol{\mu}}_j \end{bmatrix} \quad \Omega_k \equiv \begin{bmatrix} \boldsymbol{\theta}_k \\ 1 \end{bmatrix} \tag{5}$$

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 (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 (3):

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

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:

$$Z_j = \left[\sum_{k,t} \gamma_{ktj} \mathbf{x}_{kt} \Omega_k^T\right] \left[\sum_{k,t} \gamma_{ktj} \Omega_k \Omega_k^T\right]^{-1}. \tag{6}$$

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

$$\Sigma_j = \sum_{k,t} \frac{\gamma_{ktj}}{\sum_t \gamma_{ktj}} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))^T, \tag{7}$$

as is the matrix of transition probabilities [22] (see also the Appendix).

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 (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:

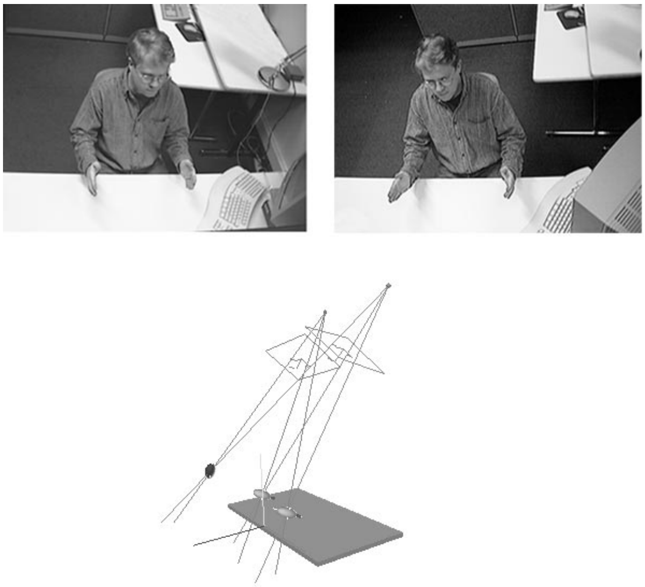


Fig. 3. The Stereo Interactive Virtual Environment (STIVE) computer vision system used to collect data in Section 4.1. Using flesh-tracking techniques, STIVE computes the three-dimensional position of the head and hands at a frame rate of about 20Hz. We used only the position of the hands for the first two experiments.

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} = \sum_t \sum_j \gamma_{tj} (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{8}$$

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

$$\boldsymbol{\theta} = \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} W_j \right]^{-1} \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} (\mathbf{x}_t - \bar{\boldsymbol{\mu}}_j) \right]. \tag{9}$$

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 10 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 4.2, in some cases it may be possible to classify the sequence by the value of $\boldsymbol{\theta}$ as determined by a single PHMM.

4 Results of Linear Model

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

4.1 Experiment 1: Size Gesture

To test the ability of the parametric HMM to learn the parameterization, 30 examples of the type depicted in Fig. 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 Fig. 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 interpreting 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 performing segmentation on an unsegmented stream of data containing multiple gestures.

A PHMM was trained with 15 sequences randomly selected from the pool of 30; 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" [22]). In this example, typically 10 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 15 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.

Fig. 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 6×1 vector whose direction indicates the linear path in six-space along which the mean $\hat{\mu}_j$ moves as $\boldsymbol{\theta}$ varies; the magnitude of W_j reflects the sensitivity of the mean to variation in $\boldsymbol{\theta}$. Table 1 gives the magnitude of the six W_j vectors for this experiment. The absolute scale of W_j is determined by the units of the feature measurements and the units of the gesture quantity $\boldsymbol{\theta}$. But, the relative scale of the W_j demonstrates that the mean of the middle states (for example, 3 and 4) is more sensitive to $\boldsymbol{\theta}$ than either the initial or final states. Fig. 5 shows 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

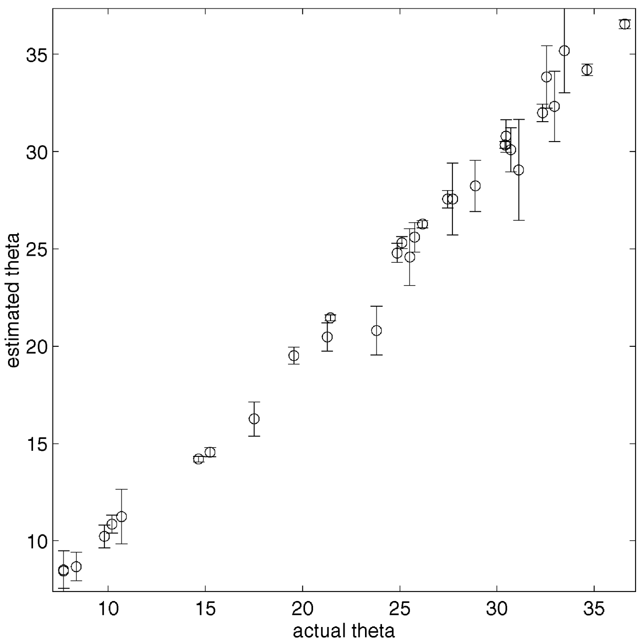


Fig. 4. Parameter estimation results for the size gesture. Fifty random choices of the test and training sets were used to compute mean and standard deviation (error bars) on all examples. The HMM was retrained for each choice of test and training set.

to accommodate the variation in $\boldsymbol{\theta}$. *The system automatically learns which segment of the gesture is most diagnostic of* $\boldsymbol{\theta}$.

4.2 Experiment 2: Recognition

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

Consider two variations of a pointing gesture: one in which the hand moves straight away from the body at some angle and another in which the hand moves from the body with some angle and then changes direction midway through the gesture. The latter gesture might co-occur with the speech "*you*, go over *there*." The first gesture we will call *point* and the second *direct*. *Point* gestures are parameterized by the angle of pointing direction (one parameter), while *direct* gestures are parameterized by the initial pointing angle to select an object and an angle to indicate the object's direction of movement (two parameters). In this experiment, we show that two HMMs 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 $\theta \in [-25, 25]$. 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.

TABLE 1
The Magnitude of W_j

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

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 the states is most sensitive to θ, in this case, the size of the fish.

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 percent was achieved on the set of 20 test sequences. With 20 states, this performance improved to only 70 percent.

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

To classify each of the 20 testing examples, it suffices to compare the value of θ_1 and θ_2 recovered by the PHMM testing algorithm. We used the single PHMM trained as above to recover parameter values. A training example was classified as a *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 two HMMs could only achieve a 70 percent 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 Fig. 6.

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

Note that, in Figs. 7c and 7d, 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

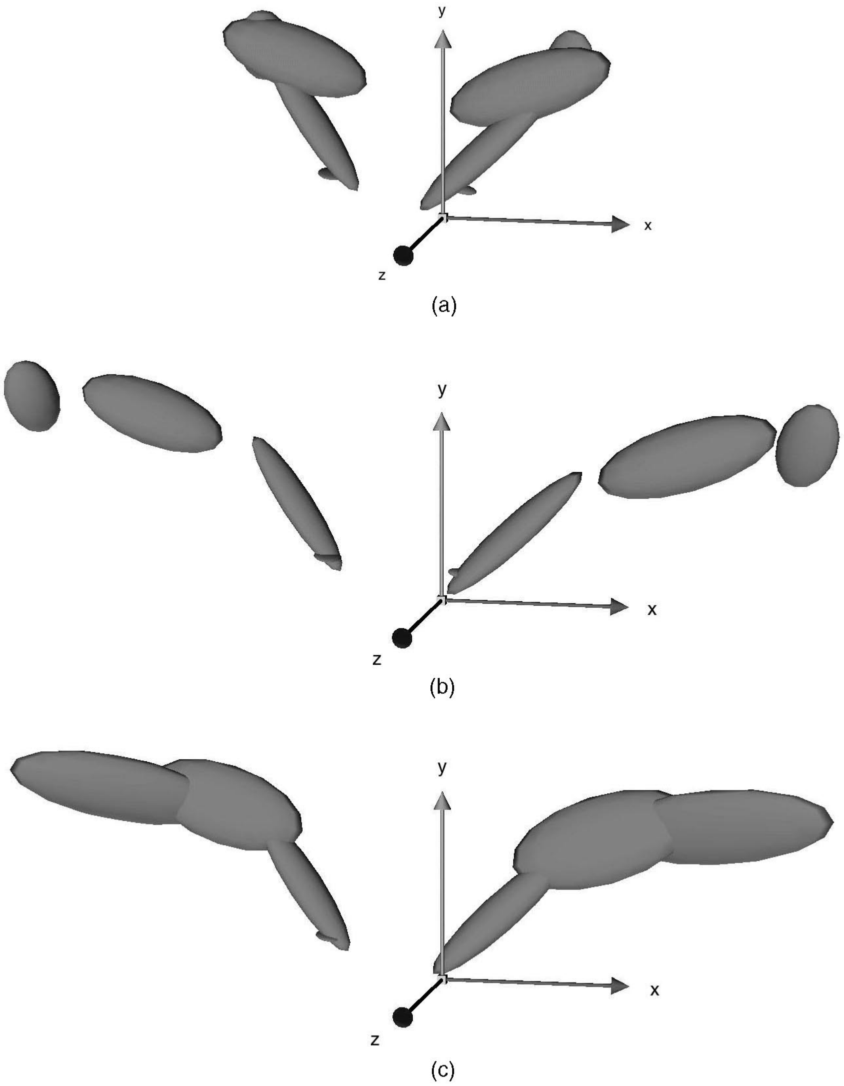


Fig. 5. The state output density of the two-handed fish-size gesture. Each corresponds to either left or right hand position at a state (for clarity, only the first four states are shown); (a) PHMM, $\theta = 19.0$, (b) PHMM, $\theta = 45.0$, (c) HMM. The ellipsoid shapes for the left hand is derived from the upper 3×3 diagonal block of the full covariance matrices, and the lower 3×3 diagonal block for the right hand.

the PHMM to more accurately model parameterized gesture that enhances its recognition performance.

4.3 Experiment 3: Robustness to Noise, Bounds on θ

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 θ. We also demonstrate using the bounds of the uniform distribution of θ to enhance the recognition capability of the PHMM.

4.3.1 Pointing Gesture

Another gesture that requires multidimensional 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. θ 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 $\theta = (x, y)$ position in a plane in front of the user (see Fig. 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 θ 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 20 sequences randomly selected from the pool of 50; again, the choice of number of states was done via cross validation. The remaining 30 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

4-state HMMs

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

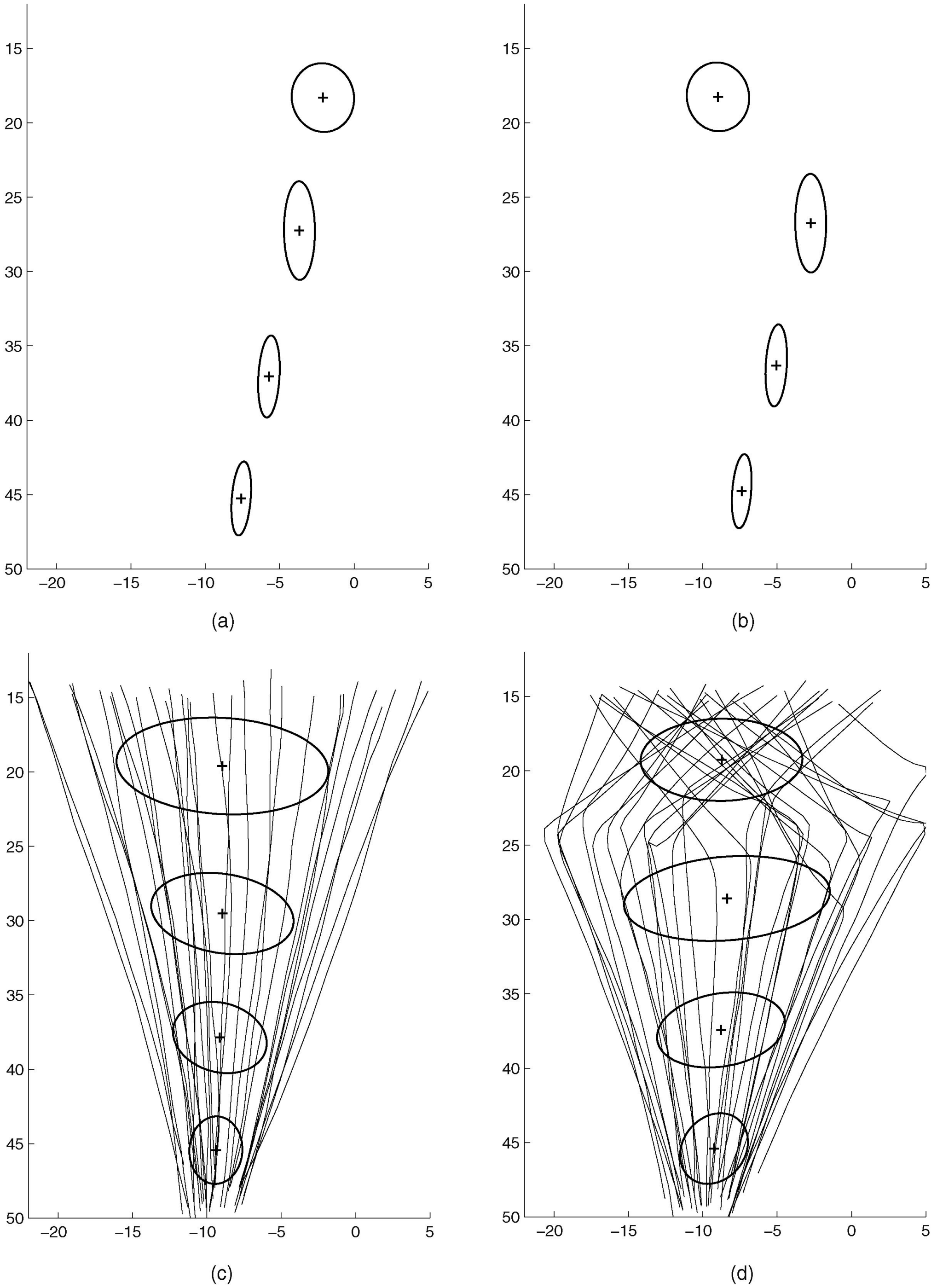
20-state HMMs

	point	*direct*
point	10	0
direct	6	4

4-state PHMM

	point	*direct*
point	10	0
direct	0	10

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



(a)

(b)

(c)

(d)

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

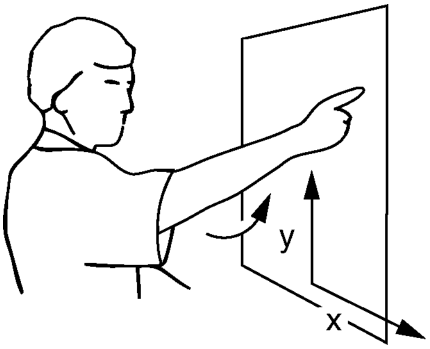


Fig. 8. The point gesture used in Section 4.3. The movement is parameterized by the coordinates of the target $\theta = (x, y)$ within a plane in front of the user. The gesture consists of a preparation phase, a stroke phase (shown here), and a retraction.

(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 θ. When the number of training examples was cut to five randomly selected sequences, the error increased to 0.82 inches (about 1.1°), demonstrating how the PHMM can exploit interpolation to reduce the amount of training data necessary. The approach discussed in Section 2.3 of tiling the parameter space with multiple unrelated HMMs would require many more training examples to match the performance of the PHMM on the same task.

4.3.2 Robustness to Noise

Because of the impact of θ on all the states of the PHMM, the entire sequence contributes evidence as to the value of θ. 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 nontrivial for many j, PHMMs should estimate θ 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 θ 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 Fig. 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, [17]) for real-world applications.

4.3.3 Bounding θ

Using the pointing data, we demonstrate how the bounds on the prior uniform density on θ 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 presegmentation) for a fixed gesture duration. Fig. 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 θ 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 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 θ estimated for each position of the sliding window, we can eliminate many of the false peaks. Recall that we assume θ has a uniform prior distribution over some allowed range. We can estimate that range from the training data either by simply taking the extremes of the training set, or by estimating the density using a ML or MAP estimate [8]. Given such bounds, we can postprocess the results of applying the PHMM by eliminating those windows which select an illegal value of θ. Fig. 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.

4.3.4 Local vs. Global Maxima

One concern in the use of EM for optimization is that, while each EM iteration will increase the probability of the observations, there is no guarantee that EM will find the global maximum of the probability surface. To show that this is not a problem in practice for the point gesture testing, we computed the log probability of a testing sequence for all legal values of θ. This log probability surface, shown in Fig. 11, is unimodal, such that for any reasonable initial value of θ the testing EM will converge on the maximum corresponding to the correct value of θ. The probability surfaces of the other test sequences in our experiments are similarly unimodal.[3]

5 Nonlinear PHMMs

5.1 Nonlinear Dependencies

The model derived in the previous section is applicable only when the output distributions of each state of the HMM are linearly dependent upon θ. 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 Section 3.2 is appropriate.

When this relation does not hold, there are at least three courses of action: 1) Find an analytical function which when applied to the feature space makes the dependence of the output distributions linear in θ, 2) find some intermediate parameterization that is linear in the feature space and then use some other technique to map

[3] Given the graphical model equivalent in Fig. 2, it is possible to exactly solve for the best value of θ using the standard inference algorithm [16]. The computational complexity of that algorithm is equivalent to that of evaluating the likelihood of the model for all value of θ, where θ is discretized to some adequate precision. Particularly for multidimensional θ, the exact inference algorithm for Bayes nets will thus involve many more computations than the EM algorithm outlined.

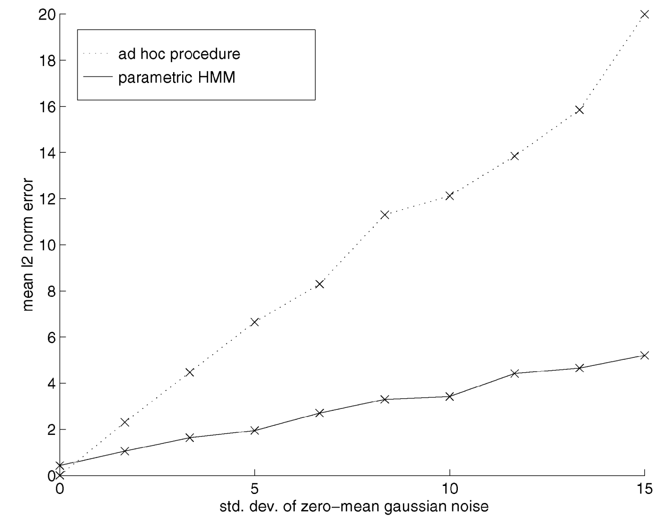


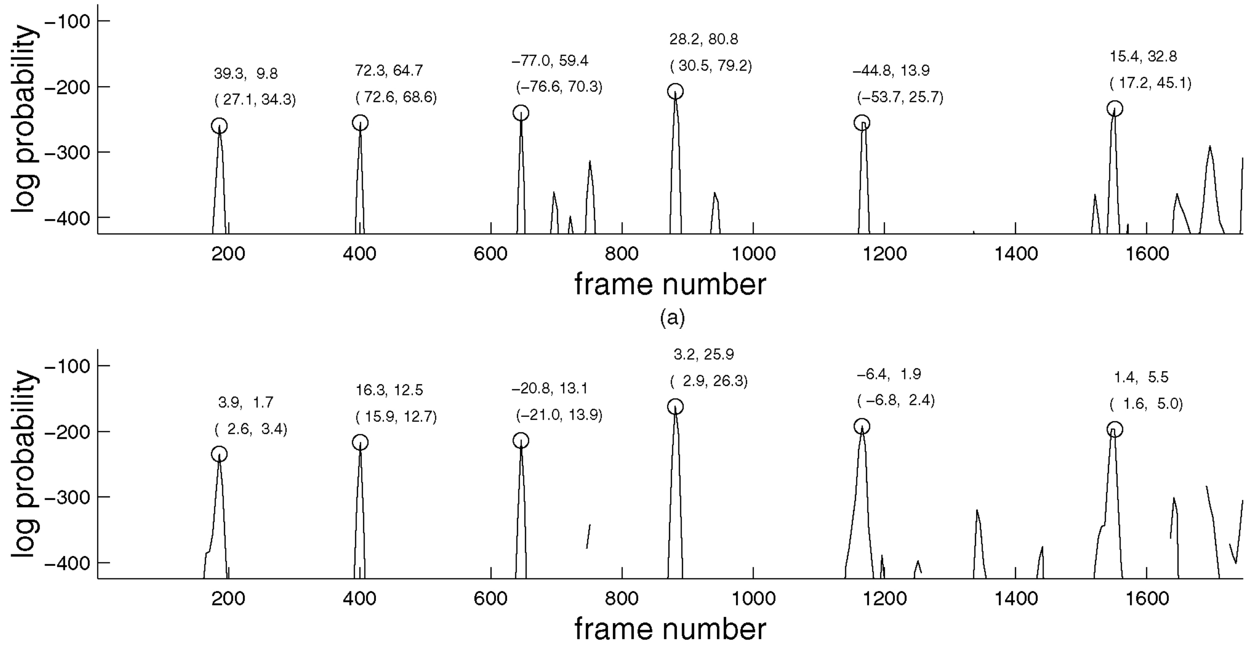
Fig. 9. Average error over the entire pointing test set as a function of noise. The value of θ was estimated by an direct measurement and by a parametric HMM retrained for each noise condition. The average error was computed by comparing the estimate of θ to the value recovered by direct measurement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

to the final parameterization, and 3) use a more general modeling technique, such as neural or radial basis function networks, to model the parametric variation of the state output densities with respect to θ.

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

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

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



(a)

(b)

Fig. 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 θ recovered by the EM testing algorithm and the ground truth value computed by direct measurement in parentheses. (a) Maximum likelihood estimate. (b) Maximum a posteriori estimate for which a uniform prior probability on θ was determined by the bounds of the training set. The MAP estimate was computed by simply disallowing sequences for which the EM estimate of θ is outside the uniform density bounds. This postprocessing step is equivalent to establishing a prior on θ in the framework presented in the Appendix.

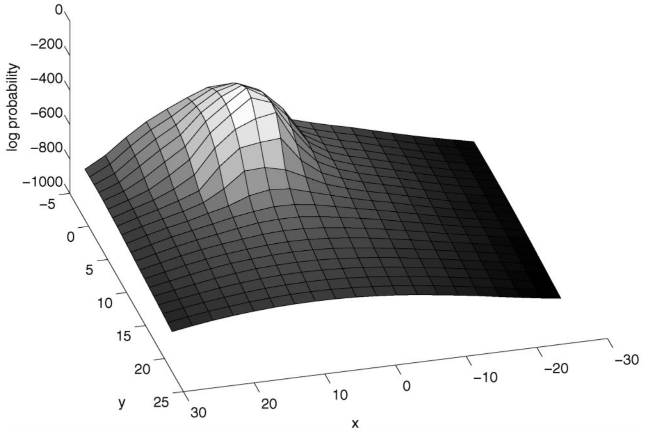


Fig. 11. Log probability as a function of $\boldsymbol{\theta} = (x, y)$ for a pointing test sequence. The smoothness of the surface makes possible to use iterative optimization techniques such as EM to find the maximum.

5.2 Nonlinear Model

Nonlinear PHMMs replace the linear model of Section 3.2 with a logistic neural network with one hidden layer. There is one neural network for each state whose function is to map the value of $\boldsymbol{\theta}$ to the output density parameters of that state. As with linear PHMMs, the output of each state is assumed to be Gaussian, with the variation of the density encoded in the mean $\hat{\boldsymbol{\mu}}_j$:

$$P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t, \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}), \Sigma_j). \tag{11}$$

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

$$\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}) = W^{(2,j)} g(W^{(1,j)})\boldsymbol{\theta} + b^{(1,j)}) + b^{(2,j)}, \tag{12}$$

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

5.3 Training

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

The expectation step of the nonlinear PHMM is the same as that of the linear PHMM. In the EM maximization step, we maximize Q. From the appendix, we have

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')}, \tag{13}$$

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

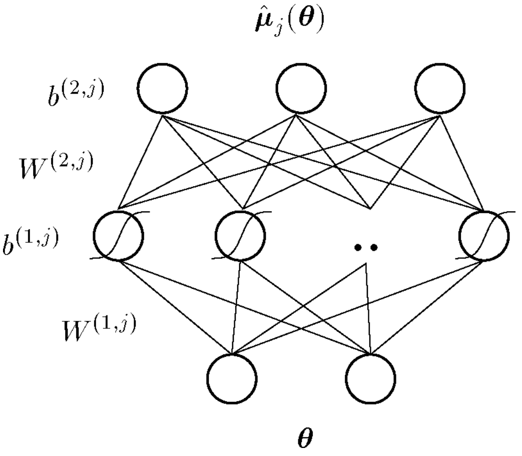


Fig. 12. Neural net architecture of the nonlinear PHMM used to map the values of $\boldsymbol{\theta}$ to $\hat{\boldsymbol{\mu}}$. There is a separate network for each state j for which the weights $W^{(i,j)}$, $i = 1, 2$, and the biases $b^{(i,j)}$, $i = 1, 2$, must be learned in training.

$$\frac{\partial Q}{\partial \phi'} = \sum_{K,t} \gamma_{tj} \Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j) \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k)}{\partial \phi'}. \tag{14}$$

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

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

$$\begin{aligned} \frac{\partial J}{\partial w_{ij}^l} &= \frac{\partial J}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l} \\ &= \delta_i^l x_j^{l-1}, \end{aligned} \tag{15}$$

where

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = g'(z_i^l) \sum_j \delta_j^{l+1} w_{ij}^{l+1} \tag{16}$$

$$\delta_i^L = y_i^* - y_i. \tag{17}$$

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

$$\Delta w_{ij}^{l} = \rho \delta_i^l x_j^{l-1}. \tag{18}$$

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

$$J = \sum_{k,t,j} \gamma_{ktj} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j)^T \Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j). \tag{19}$$

From (16), we may thus derive a new "δ rule":

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = g'(z_i^l) \sum_j \delta_j^{l+1} w_{ij}^{l+1} \tag{20}$$

$$\delta_i^L = \gamma_{tj} \Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j). \tag{21}$$

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

5.4 Testing

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

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

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} = \sum_t \sum_j \gamma_{tj} (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{22}$$

$$\frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = W^{(2,j)} \Lambda(g'(W^{(1,j)} + b^{(1,j)})) W^{(1,j)}, \tag{23}$$

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

In the results presented in this paper, we use a gradient ascent algorithm with adaptive step size [26]. In addition, it was found necessary to constrain the gradient ascent step to prevent the algorithm from wandering outside the bounds of the training data, where the output of the neural networks is essentially undefined. This constraint is implemented by simply limiting any component of the step that takes the value of $\boldsymbol{\theta}$ outside the bounds of the training data, established by the minimum and maximum $\boldsymbol{\theta}$ training values.

As with the EM training algorithm of the linear parametric case, for all of our experiments less than 10 GEM iterations are required.

5.5 Easing the Choice of Parameterization

In Section 4.3, we presented an example of a pointing gesture parameterized by projection of hand position onto the plane parallel and in front of the user at the moment that the arm is fully extended. The linear PHMM works well since the projection is a linear operation over the range of angles used in the experiment.

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

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

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

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

6 Results of Nonlinear Model

To test the performance of the nonlinear PHMM, we conducted an experiment similar to the pointing experiment of Section 4.3, but with a spherical coordinate parameterization rather than the projection onto a plane in front of the user.

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

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

Note that, for pointing gestures that are confined to a small area in front of the user (as in the experiment presented in Section 4.3), the linear parametric HMM approach will work well enough since, for small values, the tangent function is approximately linear. The pointing gestures used in the present experiment were more broad,

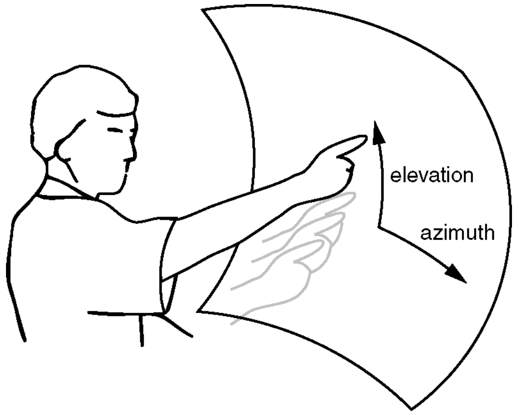


Fig. 13. The spherical coordinate system is a natural parameterization of pointing direction.

ranging from -36° to $+81^\circ$ elevation and -77° to $+80^\circ$ azimuth.

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

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

Last, we demonstrate detection performance of the nonlinear PHMM on our pointing data. A one minute sequence was collected that contained a variety of movements, including six pointing movements, distributed throughout. To simultaneously detect the gesture and recover $\boldsymbol{\theta}$, we used a 30 sample (one second) sliding window on the sequence. Fig. 15 shows the log probability as a function of time and the value of $\boldsymbol{\theta}$ recovered for a number of recovered pointing gestures. All of the pointing gestures were correctly detected and the value of $\boldsymbol{\theta}$ accurately recovered.

7 Conclusion

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

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

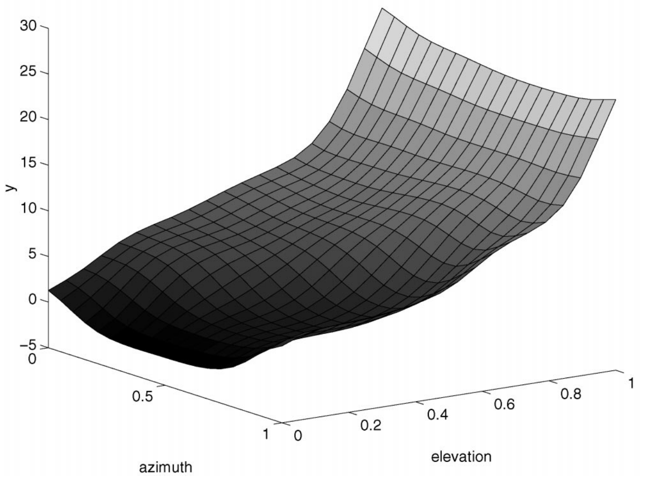


Fig. 14. The output of the logistic network corresponding to state $j = 5$ displayed as a surface. State 5 is near the apex of the gesture and shows the greatest sensitivity to pointing angle. Only the y coordinate of the output is shown; the x coordinate is similarly nonlinear.

Second, the parameterized technique presented is domain-independent and is applicable to any sequence parsing problem where some context or style ([25]) spans an entire sequence.

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

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

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

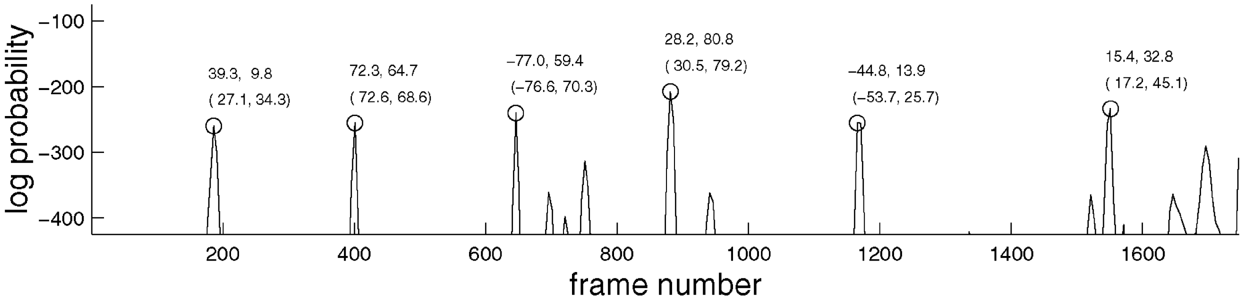


Fig. 15. Recognition results are shown by the log probability of the windowed sequence beginning at each frame number. The true positive sequences are labeled by the value of θ recovered by the EM testing algorithm and the value computed by direct measurement (in parentheses).

Appendix

Expectation-Maximization Algorithm for Hidden Markov Models

In this section, we derive (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:

$$Q(\phi' \mid \phi) = \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}[\log P(\mathbf{x}, \mathbf{q}, \phi')] \tag{24}$$

$$= \sum_{\mathbf{q}} P(\mathbf{q} \mid \mathbf{x}, \phi) \log P(\mathbf{x}, \mathbf{q}, \phi'), \tag{25}$$

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 Fig. 10.

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

$$Q(\phi' \mid \phi) = \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log \prod_t a_{q_{t-1}q_t} P(\mathbf{x}_t \mid q_t, \phi')\right]. \tag{26}$$

The expectation is carried out using the Markov property:

$$\begin{aligned} Q(\phi' \mid \phi) =&\ \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\sum_t \log a_{q_{t-1}q_t} + \sum_t \log P(\mathbf{x}_t \mid q_t, \phi')\right] \\ &= \sum_t \mathrm{E}_{\mathbf{q}|\mathbf{x},\phi}\left[\log a_{q_{t-1}q_t} + \log P(\mathbf{x}_t \mid q_t, \phi')\right] \\ &= \sum_{t,j} P(q_t = j \mid \mathbf{x}, \phi) \\ &\left[\sum_i P(q_{t-1} = i \mid \mathbf{x}, \phi) \log a_{ij} + \log P(\mathbf{x}_t \mid q_t = j, \phi')\right]. \end{aligned} \tag{27}$$

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

$$\alpha_1(j) = \pi_j b_j(\mathbf{x}_t) \tag{28}$$

$$\alpha_t(j) = \left[\sum_i \alpha_t(i) a_{ij}\right] b_j(\mathbf{x}_t) \tag{29}$$

$$\beta_T(j) = 1 \tag{30}$$

$$\beta_t(j) = \sum_j a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j) \tag{31}$$

from which γ_{tj} may be computed:

$$\gamma_{tj} = \frac{\alpha_t(j)\beta_t(j)}{P(\mathbf{x} \mid \phi)}. \tag{32}$$

In the "maximization" step, we compute ϕ' to increase Q. Taking the derivative of (27) and writing $P(q_t = j \mid \mathbf{x}, \phi)$ as γ_{tj}, we arrive at:

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')}, \tag{33}$$

which we set to zero and solve for ϕ'.

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

$$\mu_j = \frac{\sum_t \gamma_{tj} \mathbf{x}_t}{\sum_t \gamma_{tj}} \tag{34}$$

$$\Sigma_j = \frac{\sum_t \gamma_{tj} (\mathbf{x}_t - \mu_j)(\mathbf{x}_t - \mu_j)^T}{\sum_t \gamma_{tj}}. \tag{35}$$

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

$$\begin{aligned} \xi_t(i,j) &= P(q_t = i, q_{t+1} = j \mid \mathbf{x}, \phi) \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{x}_t) \beta_{t+1}(j)}{P(\mathbf{x} \mid \phi)} \end{aligned} \tag{36}$$

$$a_{ij} = \frac{\sum_t^{T-1} \xi_t(i,j)}{\sum_t^{T-1} \gamma_{tj}}. \tag{37}$$

Acknowledgments

Portions of this paper appeared in the Proceedings of the Sixth International Conference on Computer Vision, Bombay [29], and in the Proceedings of the 1998 Conference on Computer Vision and Pattern Recognition, Santa Barbara, California [28].

References

[1] A. Azarbayejani and A. Pentland, "Real-Time Self-Calibrating Stereo Person Tracking Using 3-D Shape Estimation from Blob Features," *Proc. 13th Int'l Conf. Pattern Recognition,* Vienna, Aug. 1996.

[2] Y. Bengio and P. Frasconi, "An Input Output HMM Architecture," *Advances in Neural Information Processing Systems 7,* G. Tesauro, M.D.S. Touretzky, and T.K. Leen, ed., pp. 427-434. MIT Press, 1995.

[3] C.M. Bishop, *Neural Networks for Pattern Recognition.* Oxford: Clarendon Press, 1995.

[4] C.M. Bishop, M. Svensen, and C.K.I. Williams, "EM Optimization of Latent-Variable Density Models," *Advances in Neural Information Processing Systems 8,* M.C. Moser, D.S. Touretzky, and M.E. Hasselmo, eds., pp. 402-408. MIT Press, 1996.

[5] A.F. Bobick and A.D. Wilson, "A State-Based Approach to the Representation and Recognition of Gesture," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 12, pp. 1,325-1,337, Dec. 1997.

[6] A. Bobick and J. Davis, "An Appearance-Based Representation of Action," *Proc. Int'l Conf. Pattern Recognition,* vol. 1, pp. 307-312, Aug. 1996.

[7] C. Bregler and S.M. Omohundro, "Surface Learning with Applications to Lipreading," *Advances in Neural Information Processing Systems 6,* pp. 43-50, 1994.

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

[9] L.W. Campbell, D.A. Becker, A.J. Azarbayejani, A.F. Bobick, and A. Pentland, "Invariant Features for 3-D Gesture Recognition," *Proc. Second Int'l Conf. Face and Gesture Recognition,* pp. 157-162, Killington, Vt., 1996.

[10] L.W. Campbell and A.F. Bobick, "Recognition of Human Body Motion Using Phase Space Constraints," *Proc. Int'l Conf. Computer Vision,* 1995.

[11] J. Cassell and D. McNeill, "Gesture and the Poetics of Prose," *Poetics Today,* vol. 12, no. 3, pp. 375-404, 1991.

[12] T.J. Darrell and A.P. Pentland, "Space-Time Gestures," *Proc. Computer Vision and Pattern Recognition,* pp. 335-340, 1993.

[13] T. Darrell, P. Maes, B. Blumberg, and A. Pentland, "A Novel Environment for Situated Vision and Behavior," *Proc. Computer Vision and Pattern Recognition '94 Workshop Visual Behaviors,* pp. 68-72, Seattle, Wash., June 1994.

[14] M.J.F. Gales, "Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition," CUED/F-INFENG Technical Report 291, Cambridge Univ. Eng. Dept., 1997.

[15] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation,* vol. 3, pp. 79-87, 1991.

[16] F.V. Jensen, *An Introductions to Bayesian Networks.* New York: Springer, 1996.

[17] R.E. Kahn and M.J. Swain, "Understanding People Pointing: The Perseus System," *Proc. IEEE Int'l. Symp. Computer Vision,* pp. 569-574, Coral Gables, Fla., Nov. 1995.

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

[19] H. Murase and S. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," *Int'l J. Computer Vision,* vol. 14, pp. 5-24, 1995.

[20] S.M. Omohundro, "Family Discovery," *Advances in Neural Information Processing Systems 8,* D.S. Touretzky, M.C. Moser, and M.E. Hasselmo, eds., pp. 402-408, MIT Press, 1996.

[21] H. Poizner, E.S. Klima, U. Bellugi, and R.B. Livingston, "Motion Analysis of Grammatical Processes in a Visual-Gestural Language," *Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop, Motion: Representation and Perception,* pp. 148-171, Toronto, Apr. 1983.

[22] L.R. Rabiner and B.H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine,* pp. 4-16, Jan. 1986.

[23] J. Schlenzig, E. Hunter, and R. Jain, "Vision Based Hand Gesture Interpretation Using Recursive Estimation," *Proc. 28th Asilomar Conf. Signals, Systems, and Computers,* Oct. 1994.

[24] T.E. Starner and A. Pentland, "Visual Recognition of American Sign Language Using Hidden Markov Models," *Proc. Int'l Workshop Automatic Face- and Gesture-Recognition,* Zurich, 1995.

[25] J. Tenenbaum and W. Freeman, "Separating Style and Content," *Advances in Neural Information Processing Systems 9,* 1997.

[26] S.A. Teukolsky, W.H. Press, B.P. Flannery, and W.T. Vetterling, *Numerical Recipes in C.* Cambride, U.K.: Cambridge Univ. Press, 1991.

[27] A.D. Wilson and A.F. Bobick, "Learning Visual Behavior for Gesture Analysis," *Proc. IEEE Int'l. Symp. Computer Vision,* Coral Gables, Fla., Nov. 1995.

[28] A.D. Wilson and A.F. Bobick, "Nonlinear PHMMs for the Interpretation of Parameterized Gesture," *Proc. Computer Vision and Pattern Recognition,* 1998.

[29] A.D. Wilson and A.F. Bobick, "Recognition and Interpretation of Parametric Gesture," *Proc. Int'l Conf. Computer Vision,* pp. 329-336, 1998.

[30] A.D. Wilson, A.F. Bobick, and J. Cassell, "Temporal Classification of Natural Gesture and Application to Video Coding," *Proc. Computer Vision and Pattern Recognition,* pp. 948-954, 1997.

[31] Y. Yacoob and M.J. Black, "Parameterized Modeling and Recognition of Activities," *Computer Vision and Image Understanding,* vol. 73, no. 2, pp. 232-247, 1999.

[32] J. Yamato, J. Ohya, and K. Ishii, "Recognizing Human Action in Time-Sequential Images Using Hidden Markov Model," *Proc. Computer Vision and Pattern Recognition,* pp. 379-385, 1992.

Andrew D. Wilson received his BA in computer science from Cornell University, Ithaca, in 1993 and his MS in media arts and sciences from the Massachusetts Institue of Technology in 1995. He is currently a PhD candidate with the Vision and Modeling Group at the MIT Media Laboratory. His research activities include work on developing models for the representation of gesture and human motion, online adaptive models of learning, and realtime computer vision. The current emphasis of his work is on online adaptive learning techniques for robust and flexible gesture recognition systems.

Aaron F. Bobick received his PhD in cognitive science from the Massachusetts Institute of Technology in 1987 and also holds BS degrees from MIT in mathematics and computer science. In 1987, he joined the Perception Group of the Artificial Intelligence Laboratory at SRI International and, soon after, was jointly named a visiting scholar at Stanford University. From 1992 until July 1999, he served as an assistant and then associate professor in the Vision and Modeling Group of the MIT Media Laboratory. He has recently moved to the College of Computing at the Georgia Institute of Technology, where he is an associate professor in the GVU and Future Computing Environments Laboratories.

Professor Bobick has performed research in many areas of computer vision. His primary work has focused on video sequences where the imagery varies over time either because of change in camera viewpoint or change in the scene itself. He has published papers addressing many levels of the problem from validating low level optic flow algorithms to constructing multirepresentational systems for an autonomous vehicle to the representation and recognition of high level human activities. The current emphasis of his work is on action understanding, where the imagery is of a dynamic scene and the goal is to describe the action or behavior. Three examples are the basic recogniton of human movments, natural gesture understanding, and the classification of football plays. Each of these examples require describing human activity in a manner appropriate for the domain, and developing recognition techniques suitable for those representations.