M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 308
Abbreviated version appears in ICCV '95

Using Configuration States for the Representation and Recognition of Gesture

Aaron Bobick **Andy Wilson**
Room E15-383, The Media Laboratory
Massachusetts Institute of Technology
20 Ames St., Cambridge, MA 02139
E-mail: drew, bobick@media.mit.edu

Abstract

A state-based technique for the summarization and recognition of gesture is presented. We define a gesture to be a sequence of states in a measurement or configuration space. For a given gesture, these states are used to capture both the repeatability and variability evidenced in a training set of example trajectories. The states are positioned along a prototype of the gesture, and shaped such that they are narrow in the directions in which the ensemble of examples is tightly constrained, and wide in directions in which a great deal of variability is observed. We develop techniques for computing a prototype trajectory of an ensemble of trajectories, for defining configuration states along the prototype, and for recognizing gestures from an unsegmented, continuous stream of sensor data. The approach is illustrated by application to a range of gesture-related sensory data: the two-dimensional movements of a mouse input device, the movement of the hand measured by a magnetic spatial position and orientation sensor, and, lastly, the changing eigenvector projection coefficients computed from an image sequence.

1 Background

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

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

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

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

2 Related Work

Early experiments by Johansson with Moving Light Display (MLD) images suggest that many movements or gestures may be recognized by motion information alone. Work with very low resolution American Sign Language images by Sperling et al. [13] further supports the notion that in many domains a full geometric reconstruction of the moving object is unnecessary for recognition. For example, Polana and Nelson [10] use low level features of motion to recognize periodic motions such as walking.

A number of researchers have developed novel representations for motion trajectories that are useful in gesture recognition. Gould and Shah [5] show the analysis of motion trajectories to identify event boundaries. These are recorded in their trajectory primal sketch to be used for motion recognition. Rangarajan et al. [11] demonstrate two-dimensional motion trajectory matching through scale-space. Davis and Shah [4] recognize simple hand gestures by matching two-dimensional trajectories made by markers on the fingertips. Rohr [12] smooths a number of example joint angle trajectories to build a representation of one walking cycle parameterized by a pose variable.

Others have concentrated on capturing how the output of various image operators change over the course of movement. Darrell and Pentland [3] use dynamic time warping to match changing normalized image correlation template scores to learned models. The correlation templates are evenly distributed over the length of the model gesture so that it is always the case that some template has a high match score. Murase and Nayar [9] match changing eigenvector projection coefficients taken from the image data to determine the orientation and illumination angle of the object. Bregler and Omohundro [1] learn a surface representing constraints on the image sequence for their nonlinear image interpolation task. Yamato et al. [16] compute a simple region-based statistic from each frame of an image sequence, which is then vector-quantized. Sequences of the discrete symbols are then identified by a trained Hidden Markov Model.

Lastly, some work has been done in the real time recognition of simple mouse input device gestures. Tew and Gray [14] use dynamic programming to match mouse trajectories to prototype trajectories. Lipscomb [7] concentrates on filtering the mouse movement data to obtain robust recognition of similarly filtered models. Mardia et al. [8] compute many features of each trajectory and use a learned decision tree for each gesture to best utilizes the features for recognition.

3 Motivation for a Representation

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

Secondly, we desire a representation that is invariant to nonuniform changes in the speed of the gesture to be recognized. These shifts may also be thought of as non-linear shifts in time. A global shift in time caused by a slight pause early in the gesture should not affect the recognition of most of the gesture. Let us call the space of measurements that define each point of an example gesture a *configuration space*. The goal of time invariance is motivated by the informal observation that the important quality in a gesture is how it traverses configuration space and not exactly when it reaches a certain point in the space. In particular, we would like the representation to be time invariant but order-preserving: e.g. first the hand goes up, then it goes down. Finally, strong time invariance is probably not required because in the case of natural gestures, many kinds of time shifts are simply not physically plausible.

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

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

We can now consider the state membership function of an entire trajectory. Let $T_i(t)$ be the i^{th} trajectory. We need to choose a *combination rule* that defines the state membership of a point x in configuration space with respect to a group of states. For convenience let us choose `max`, which assigns the combined membership of x, $M_S(x)$, the value $\max_i(\mu_{s_i}(x))$. The combined membership value of a trajectory is a function of time while the assigned state of the trajectory at each time instant is the state whose membership is greatest. Thus, a set of configuration states translates a trajectory into a symbolic description, namely a sequence of states.

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

4 Computing the Representation

The technique presented to compute the configuration state representation proceeds in three steps. Beginning with the sample points of a number of example motion trajectories, the first step computes a single prototype configuration space curve for the ensemble of trajectories. The prototype curve removes time as an axis, and is simply parameterized by arc length λ as it moves through configuration space, $P(\lambda) \in \Re^d$. The goal of this parameterization is to group sample points that are nearby in configuration space and to preserve the temporal order along each of the example trajectories.

In practice, the prototype curve is represented by a series of discrete points in configuration space. The second step treats each of the segments of the prototype curve as vectors in the space. These vectors are clustered according to their proximity to each other in configuration space and their similarity in direction.

In the third and last step, each of the clusters of prototype curve vectors is used to define a fuzzy state. The arc length parameterization of the prototype curve vectors in each cluster is known. The trajectory sample points that have the same arc length projections are collected for each cluster; the distribution of the points determines the membership function for that state. By clustering the prototype curve vectors rather than the sample points directly, we obtain states that fall one after the other along the prototype. Furthermore, the shape of each state is determined by the points that project to similar arc lengths of the prototype, and not simply nearby points.

The following subsections detail the steps of the algorithm for computing the representation.

4.1 Computing the prototype

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

Our definition of a prototype curve of an ensemble of training trajectories $T_i(t)$ is a continuous one-dimensional curve that best fits the sample points in configuration space according to a least squares criterion. For ensembles of space curves in metric spaces there are several well known techniques that compute a "principal curve" [6] that attempts to minimize distance between each point of each of the trajectories and the nearest point on the principal curve.

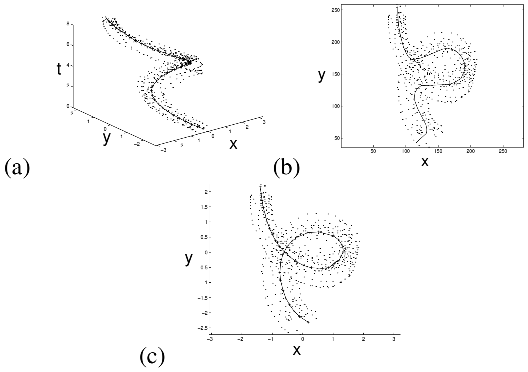


Figure 1: (a) Example trajectories as a function of time. (b) Projection of trajectory points into configuration space. Normal principal curve routines would lose the intersection. (c) Prototype curve recovered using the time-collapsing technique (see Appendix).

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

The problem of computing a prototype curve P in configuration space is how to collapse time from the trajectories $T_i(t)$. Figure 1 illustrates the difficulty. If the points that make up the trajectories (a) are simply projected into configuration space by removing time (b), there is no clear way to generate a connected curve that preserves the temporal coherence of the path through configuration space. Likewise, if each of the trajectories is projected into configuration space small variations in temporal alignment make it impossible to group corresponding sections of the trajectories without the consideration of time.

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

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

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

4.2 Clustering the sample points

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

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

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

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

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

4.3 Determining state shapes

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

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

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

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

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

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

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

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

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

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

5 Recognition

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6 Experiments

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

In each case, the measurements collected were uniformly scaled to fall within the interval from -1 to 1. With this scaling, the same initial time scaling s of 4.0 was found to work well in computing the prototype trajectory (see Appendix). The prototype curve was computed in turn for s = 4,3,2,1,0 in each case. In some cases it was necessary to change the value of the smoothing kernel parameter h and the values of σ_c and σ_a. Otherwise no adjustment of parameters was made between the experiments.

6.1 Mouse Gestures

In the first experiment, we demonstrate how the representation permits the combination of multiple gestures, and generalizes to unseen examples for recognition. Furthermore, by considering two dimensional data, the cluster points and the resulting states are easily visualized.

The (x, y) position of a mouse input device was sampled at a rate of 20Hz for two different gestures G_α and G_β. Ten different examples of each gesture were collected; each consisted of about one hundred sample points. Half of the examples were used to

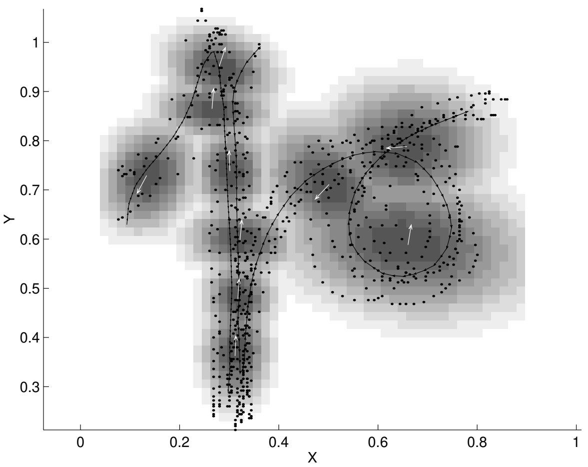


Figure 2: The prototype curves (black) for G_α and G_β are combined to find a set of states to describe both gestures. The clustering of the prototype curve vectors for G_α and G_β gives the clustering of all sample points. Each of the ten clusters depicted here is used to define a state centered about each of the white vectors. The combined membership of the 10 states in G_α and G_β is depicted under the curves; darker regions indicate high membership.

compute the prototype curve for each gesture. The smoothing kernel parameter h had a value of 0.2. The vectors along both of the curves were then clustered to find a single set of ten states useful in tracking either gesture. Because the gestures overlap, six of the states are shared by the two gestures. The prototype curves and the assignments of the sample points to the clusters are shown in Figure 2.

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

The match scores of the hand-segmented test gestures were then computed using the dynamic programming algorithm outlined in Section 5. In computing the maximum match score, the algorithm assigns each point to a state consistent with the sequence G_α. As described, a match is made only if it is considered complete.

The state transitions and the membership values computed for each sample are shown for the new examples of G_α and G_β in Figure 3. In the plots, the state transitions are marked as vertical bars; thus all sample points between two adjacent vertical bars are attributed to the same state. As described in Section 5, a transition to a state is only allowed if it is consistent with G_α. A match was considered complete if the minimum of the maximum memberships between each transition was greater than 0.4. The state transition plots for this experiment and the others graphically depict the time-invariant but order-preserving nature of the representation.

The representation thus provides a convenient way to specify a gesture as an ordered sequence of states while also permitting the combination of states shared by multiple gestures. By using the sequence of states in the recognition task, nearby and possibly overlapping states belonging to another gesture or another part of the same do not confuse the matching process.

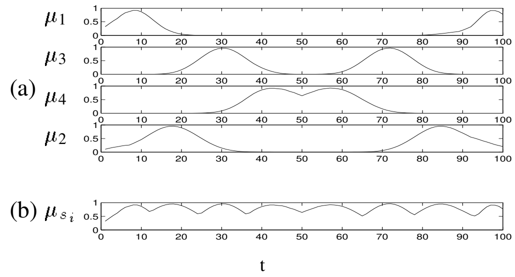


Figure 4: (a) The membership plot for the prototype curves for G_{wave} show for each of the configuration states how the states lie along the prototype. (b) Combined membership (maximum) of all states at each point along the prototype. The prototype for G_{point} is similar.

6.2 Spatial Position and Orientation Sensor Gestures

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

An Ascension Technology Flock of Birds magnetic position and orientation sensor was worn on the back of the hand and polled at 20Hz. For each sample, the position of the hand and the normal vector out of the palm of the hand was recorded (six dimensions). Ten large wave gestures (about 40 samples each) and ten pointing gestures (about 70 samples each) were collected. To insure that there were enough points available to compute the prototype curve, each example was upsampled using Catmull-Rom [2] splines so that each wave gesture is about 40 samples and each point gesture about 70 samples.

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

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

A stream of samples consisting of an alternating sequence of all the example wave and point gestures was causally segmented to find all wave and point gestures. For a matching threshold of $\eta = 0.5$, all the examples were correctly segmented (Figure 6).

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

6.3 Imagespace Gestures

As a final experiment, we consider a digitized image sequence of a waving hand (Figure 7). Given a set of smoothly varying measurements taken from the images and a few hand-segmented

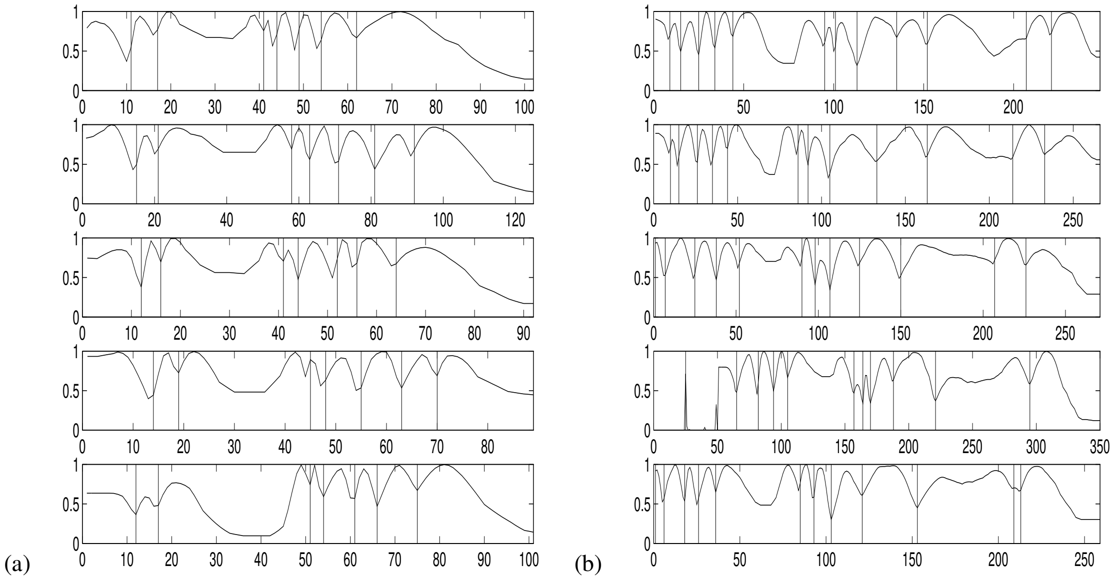


Figure 3: The state transition and membership plots for the testing examples for G_α (a) and G_β (b). The state transitions are marked by vertical bars. The transitions are calculated to maximize the average membership while still giving an interpretation that is complete with respect to G_α and G_β. No complete and consistent interpretation could be found for the fourth example of G_β.

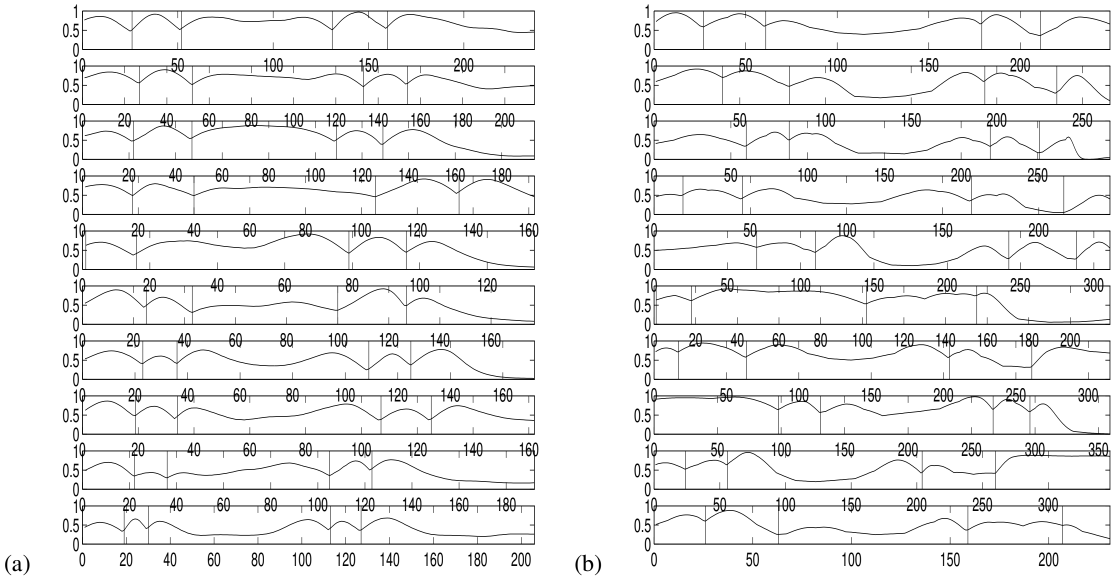


Figure 5: The state transition and membership plots for the testing examples for G_{wave} (a) and G_{point} (b). The state transitions are marked by vertical bars.

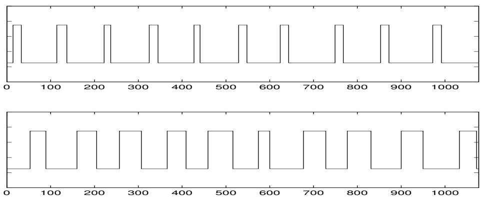


Figure 6: The causal segmentation of the stream of position and orientation data, for the wave gesture (top) and point gesture (bottom). A high value indicates the detection of a single complete and consistent gesture. All examples of both gesture were detected in this test.

example waves, the goal is to automatically segment the sequence to recover the remaining waves. This example shows how the measurements do not require a direct physical or geometric interpretation, but should vary smoothly in a meaningful and regular way.

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

The first five example waves were used in training. The three eigenvectors with the largest eigenvalues were computed by the Karhunen-Loeve Transform (as in [15, 9]) of the training frames, treating each frame as a column of pixel values. The first three eigenvectors accounted for 71% of the variance of the pixel intensity values of the training frames.

The training frames were then projected onto the eigenvectors to give the smooth trajectories shown in Figure 8. The membership along the prototype curve (h = 0.4) computed from the projection trajectories is shown in Figure 9. The recovered state sequence $G_{wave} = \langle s_1 s_4 s_3 s_2 s_3 s_4 \rangle$ again shows the periodicity of the motion. Figure 10 shows the state transitions and membership values for ten other examples projected onto the same eigenvectors. Again, the first and last states were ignored in the matching process due to variations in the exact beginning and ending of the motion. Lastly, Figure 11 shows the automatic causal segmentation of the whole image sequence was computed. Of the 32 waves in the sequence, all but one (the same incomplete example above) were correctly segmented.

This example demonstrates how the representation may be used in a broad range of tracking and segmentation tasks; namely, those in which smoothly varying and meaningful measurements can be collected from the example gestures.

7 Conclusion

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

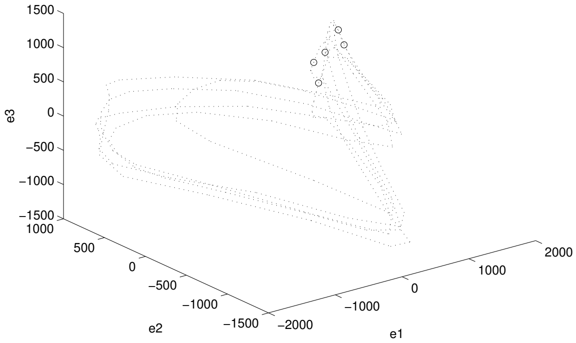


Figure 8: Each of the axes e_1, e_2, and e_3 represents the projection onto each of the three eigenvectors. The image sequences for the five training examples project to the smooth trajectories shown here. A circle represents the end of one example and the start of the next. The trajectory plots show the coherence of the examples as a group, as well as the periodicity of the movement.

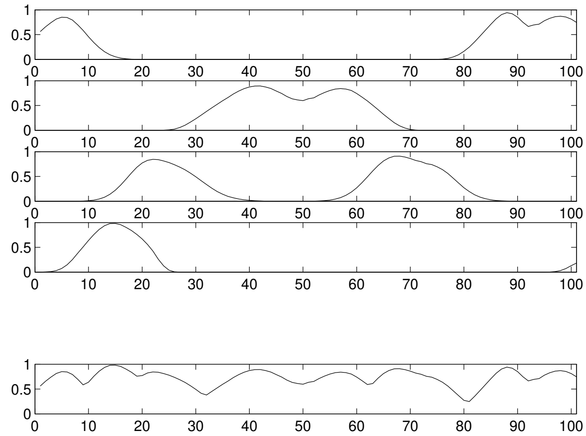


Figure 9: The membership along the prototype curve for each state is shown in the top four plots. The bottom plot shows the combined membership of all states along the prototype.

Figure 7: Each row of images depicts a complete wave sequence taken from the larger image sequence of 830 frames, 30 fps, 60 by 80 pixels each. Only 5 frames of each sequence is presented. The variation in appearance between each example (along columns) is typical of the entire sequence.

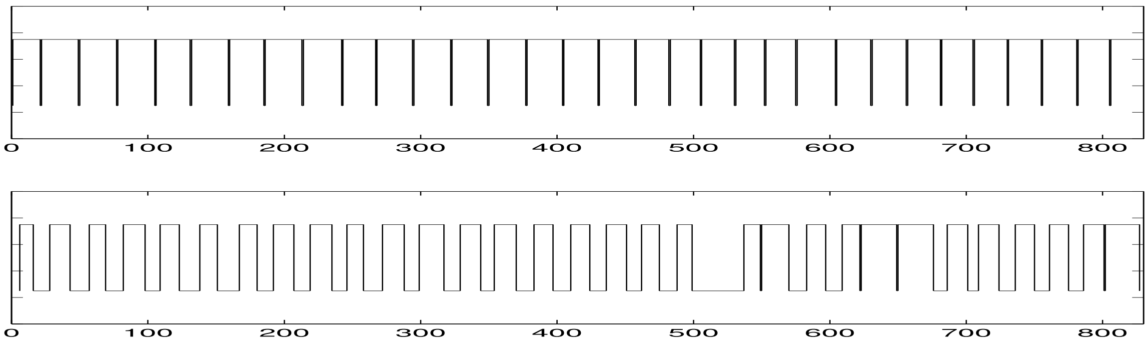


Figure 11: The stream recognition process results in a segmentation of the entire video sequence. A high value indicates a complete and consistent gesture. All but one of the 32 examples of the wave gesture are correctly identified.

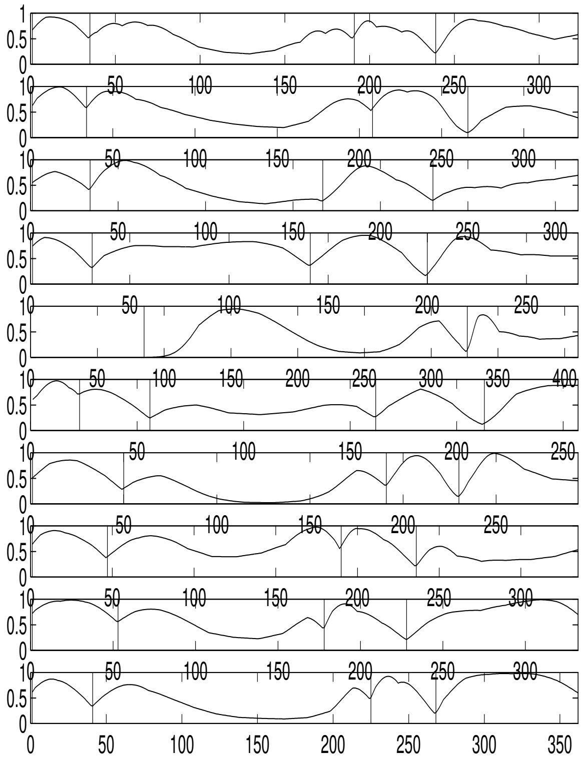


Figure 10: The state transition and membership plots for testing examples 16 through 25, taken from the image sequence of 32 examples. The state transitions are marked by vertical bars. The fifth example was deemed incomplete. On most of the successfully tracked examples, the dynamic programming algorithm chooses to make an immediate transition to the second state; this is caused by an overly high degree of overlap between states, controlled by σ_a.

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

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

The representation of a gesture as a sequence of predefined states along a prototype gesture is a convenient symbolic description, where each state is a symbol. In one experiment we demonstrated how two gestures can share states in their defining sequences. This description may also be useful in composing new gestures from previously defined ones, in detecting and allowing for periodicity in gesture, and in computing groups of states that are atomic with respect to a number of gestures. In short, the symbolic description permits a level of "understanding" of gesture that we have not explored.

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

References

[1] C. Bregler and S. M. Omohundro. Nonlinear image interpolation using surface learning. In G. Tesauro J. D. Cowan and J. Alspector, editors, *Advances in neural information processing systems 6*, pages 43–50, San Fransisco, CA, 1994. Morgan Kaufmann Publishers.

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

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

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

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

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

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

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

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

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

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

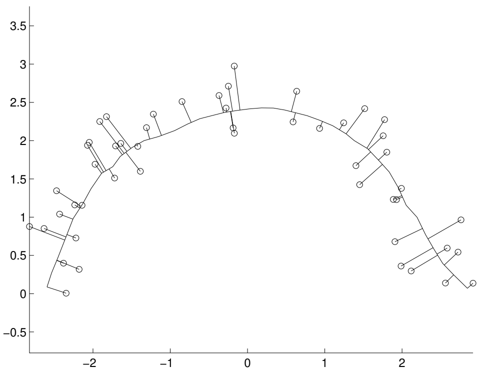


Figure 12: A principal curve calculated from a number of points randomly scattered about an arc. Only a fraction of the data points are shown; the distances of these points to the closest points on the curve are indicated.

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

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

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

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

[16] 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.

A Computation of time collapsed prototype curve

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

To compute the prototype curve $P(\lambda)$, we use Hastie and Stuetzle's "principal curves" [6]. Their technique results in a smooth curve which minimizes the sum of perpendicular distances of each sample to the nearest point on the curve. The arc length along the prototype of the nearest point is a useful way to parameterize the samples independently of the time of each sample. That is, for each sample x_i there is a lambda which minimizes the distance to $P(\lambda)$: $\lambda(x_i) = \underset{\lambda}{\arg\min} \|P(\lambda) - x_i\|$. An example is shown in Figure 12.

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

In practice, only one sample point will project to a particular arc length along the curve. Therefore, a number of points that project to approximately equal arc lengths are averaged. The approach suggested by Hastie and Stuetzle and used here is to compute a weighted least squares line fit of the nearby points, where the weights are derived from a smooth, symmetric and decreasing kernel centered about the target arc length. The weight $w$for a sample x_i and curve point $p = P(\lambda)$ is given by

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

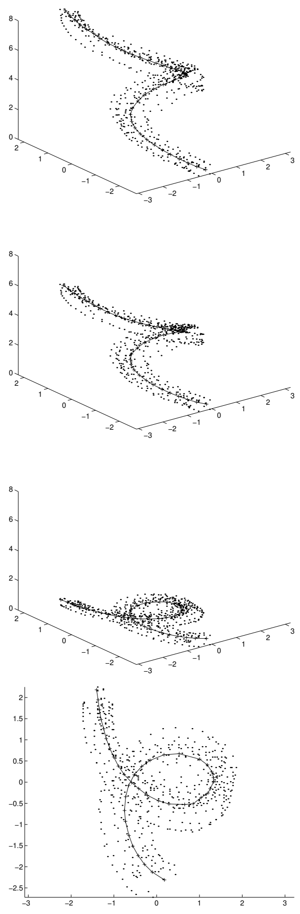
where h controls the width of the kernel.

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

By starting with a time scaling s which renders the trajectories slowly varying in the configuration space parameters as a function of arc length, the principal curve algorithm computes a curve which is consistent with the temporal order of the trajectory samples. Then the time scale s can be reduced somewhat and the algorithm run again, starting with the previous curve. In the style of a continuation method, this process of computing the curve and rescaling time repeats until the time scale is zero, and the curve is in the original configuration space. To ensure that points along the prototype do not coincide nor spread too far from one another as the curve assumes its final shape, the principal curve is resampled between time scaling iterations so that the distance between adjacent points is constant.

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

By repeatedly applying the principal curve algorithm and collapsing time, a temporally consistent prototype $P(\lambda)$ is found in configuration space. Additionally, the arc length associated with each projected point, $\lambda(x_i)$, is a useful time-invariant but order-preserving parameterization of the samples. An example of this time-collapsing process is shown in Figure 13.



(a)

(b)

(c)

(d)

Figure 13: The principal curve is tracked while time is slowly collapsed in this series: (a) s = 8, (b) s = 5, (c) s = 0. In each of these graphs, the vertical axis is time. (d) shows the final, temporally consistent curve.