M.I.T. Media Laboratory Vision and Modeling Group Technical Report No. 197

Recognition of Space-Time Gestures using a Distributed Representation

Trevor J. Darrell and Alex P. Pentland

ABSTRACT

This paper presents a method for learning, tracking, and recognizing human gestures using a view-based approach to model both object and behavior. Object views are represented using *sets* of view models, rather than single templates. Stereotypical space-time patterns, i.e. gestures, are then matched to stored gesture patterns using dynamic time warping. Real-time performance is achieved by using special-purpose correlation hardware and view prediction to prune as much of the search space as possible. Both view models and view predictions are learned from examples. We present results showing tracking and recognition of human hand gestures at over 10Hz.

1 Introduction

The location and orientation of head, hand, and eyes is a critical element of all human dialog. The ability to follow objects moving through space and recognize particular motions as meaningful gestures is therefore essential if computer systems are to interact naturally with human users [2, 6]. Currently, however, this information is largely unavailable to computing machines. An important new application of machine vision, therefore, is to extend the interface between man and machine, allowing the machine to directly perceive what its user is doing [5, 10, 12, 3]. This paper reports a system that can recover such information in real time, so that it can help mediate human-machine interaction.

To accomplish this goal, we have adopted a view-based representation of objects and actions, since such an approach places few restrictions on the range of shapes and behaviors that can be described. This allows us to model complex, articulated objects for which no simple 3-D model or recovery method is available. Moreover, the use of a view-based approach allows us to learn our models by observation, rather than needing precise CAD models.

The drawback of such an approach is that complex, articulated objects (such as hands) have a very large range of appearances, making traditional approaches to view-based matching difficult. We have therefore adopted a representation based on *interpolation* of appearance from a relatively small number of views. This is similar to the ideas of Ullman [11] and Poggio [7] for representing 3-D objects by interpolating between a small set of 2-D views. We are extending this idea in two ways: first, by building the representation using an ensemble of matched filters rather than feature sets, and second, by addressing the problem of recognizing the behavior of articulated objects in space-time.

In our system the match statistic is the normalized correlation between the image and a set of learned *view models*. Each model is based on one or more example images of a view of an object, from which mean and variance statistics about each pixel in the view are computed. The view models capture the reliable portions of the appearance of a particular view of an object, and exclude pattern details that vary across learning trials. Areas with highlights, and high-frequency geometric details are typically excluded from a view model.

A particular view model will have a range of parameter values of a given transformation (e.g., rotation, scale, articulation) for which the match statistic shows a roughly convex "tuning curve". If we have a set of views models which sample the transformation parameter finely enough, we can infer the actual transform parameters for new views by examining the *set* of model correlation scores. For example, Figure 1a shows three images of an eyeball that were used to create view models for gaze tracking; one looking 30 degrees left, one looking center-on, and one looking 30 degrees to the right. The three views span a ± 30 degree subspace of the gaze direction parameter. Figure 1(b,c,d) shows the normalized correlation score for each view model when tracking a rotating eyeball. Since the tuning curves produced by these models are fairly broad with respect to gaze angle, one could interpolate from their responses to obtain a good estimate of the true angle.

For a correctly-chosen set of view models, the set of correlation scores is theoretically adequate to permit recovery of the object's parameter values. However this requires knowing the parameter values associated with each model, information that is often difficult to obtain. Fortunately, this information is not required to recognize stereotyped behaviors, such as the hand gestures shown in Figure 2. Because the parameter values underlying the object pose are associated with the set of correlation scores, we can characterize a behavior by the pattern of view model scores directly.

To recognize a gesture in this way we first measure the scores of a set of view models for each frame of an image sequence. Then instead of deriving parameter values from this data, we ask which known behavior (gesture) most closely matches the observed *space-time* pattern of model scores. That is, rather than transforming sets of view model scores to parameters and then matching in parameter space, we match in the model-score space directly. This allows us to recognize gestures without explicit knowledge of the transformation parameters.

The domain of human gesture recognition seems particularly promising for this approach. It is known, for instance, that the human ability to interpret American Sign Language remains accurate even for 32 x 32 pixel high-contrast images with a frame rate of 16 images per second [9]. The ability to "read" sign language from very low-resolution, poor quality imagery indicates that humans do not require precise contours, shading, texture, or 3-D properties. What they require is a coarse 2-D description of hand appearance and an accurate representation of the hand's 2-D trajectory. This mix of coarse 2-D shape information and precise trajectory information is exactly what a fast, view-based interpolation approach can hope to provide.

We have implemented a system which automatically builds a view-based object model and its contextual dependencies while tracking the object through space and time. Special hardware for real-time normalized correlation is used to match views in 2-D images, the

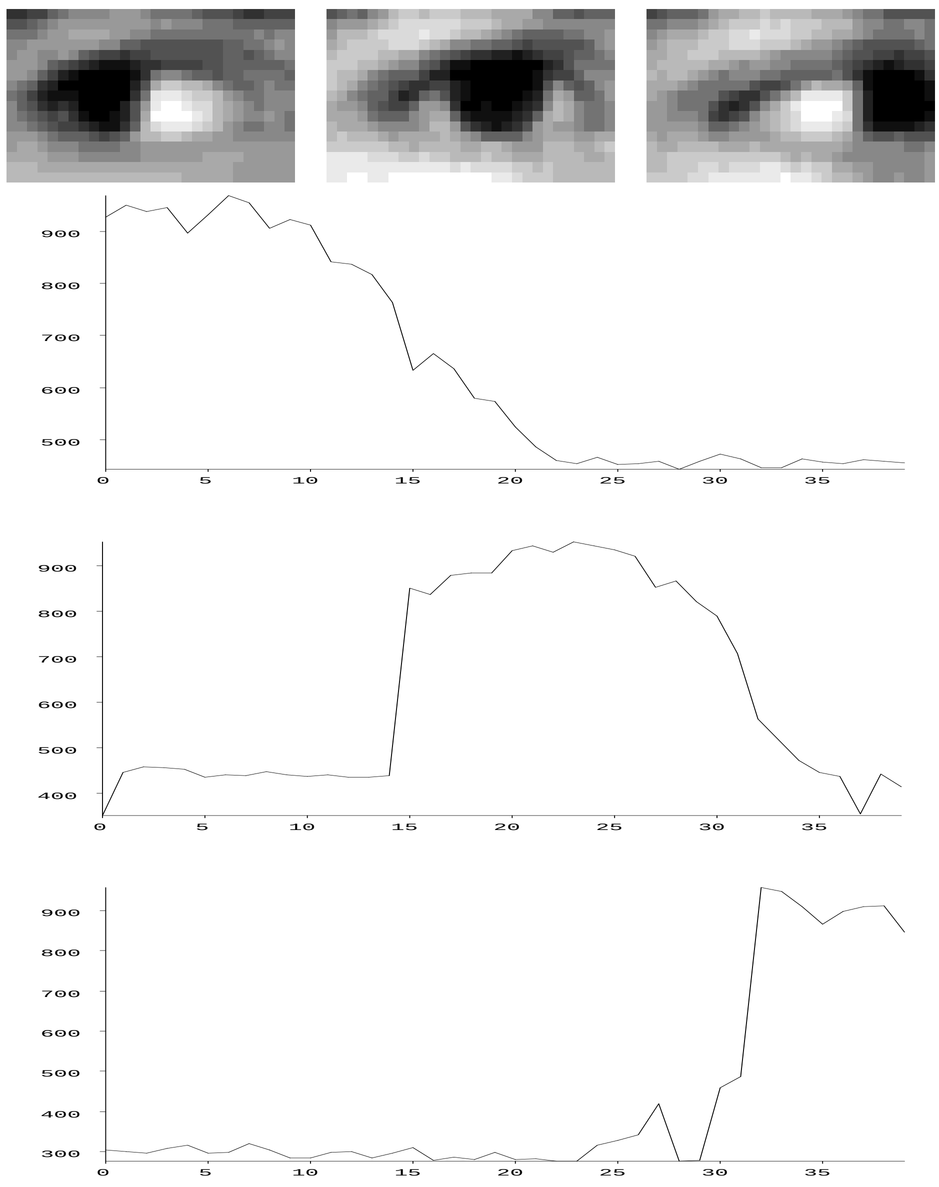

Figure 1: (a) Three views of an eyeball: +30, 0, and −30 of gaze angle. (a) Normalized correlation scores of the +30 degree view model when tracking a eyeball rotating from approximately −30 to +30 degrees of gaze angle. (b) Score for 0 degree view model. (c) Score for −30 degree model.

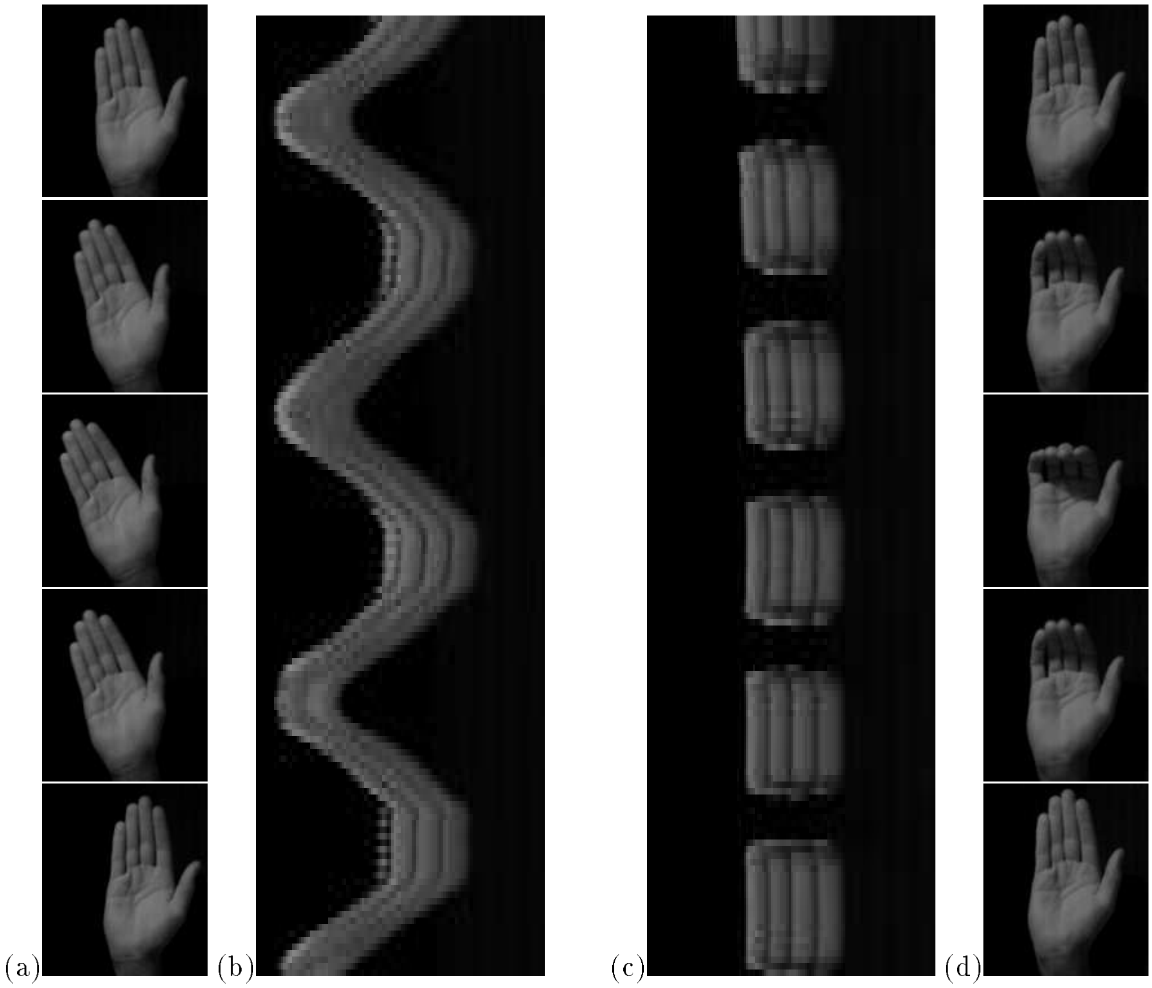

Figure 2: Space-time pattern of two hand waving gestures: (a,b) "hello", (c,d) "goodbye". (a) and (d) show frame numbers 0, 5, 10, 15, 20 sampled from a sequence of 70 frames; (b) and (c) show a horizontal spatio-temporal slice through the sequence, taken just above the position of the thumb in the image.

results of which are integrated over time to match gestures in 3-D (e.g., to match a space-time sequence of views). The following sections discuss in detail the view-based model acquisition and context sensitive searching, and then the extension to model and recognize gestures. We will show results of a real-time implementation tracking simple and complex objects, and on recognizing the hand gestures shown in Figure 2.

2 Learning View Models

A key problem for this approach is how to learn an appropriate set of view models, given the requirement that they adequately sample the parameters of the transformations an object may undergo. For simple transforms such as rotation or scale, analytic methods can often provide an optimal set of sample points. For more complex situations, such as articulated or unfamiliar objects, we have no adequate analytic model. Therefore we have developed a data-driven method of automatically constructing a set of view models whose normalized correlation scores can be meaningfully interpolated.

2.1 Correlation-Based Search

Our tracking system performs a normalized correlation search using a set of view models. Given a set of models indexized by the variable m, $1 \leq m \leq M$, the search function is

$$R_m(i,j) = \frac{n_m \sum_{V_m(u,v)<\gamma} T_m(u,v) I(i+u,j+v) - \sum_{V_m(u,v)<\gamma} T_m(u,v) \sum_{V_m(u,v)<\gamma} I(i+u,j+v)}{n_m^2 \sigma_I \sigma_{T_m}} \tag{1}$$

where $T_m()$, $V_m()$ are the mean and variance of pixels in the set of example images for model m, $I()$ is the new image being searched, n_m is the number of valid pixel locations in the view model, γ is the "don't-care" threshold on model variance, and σ_I,σ_{T_m} are the standard deviation of the offset and model images, respectively:

$$\sigma_I = \sqrt{\frac{1}{n_m} \sum_{V_m(u,v)<\gamma} I^2(i+u,j+v) - (\frac{1}{n_m} \sum_{V_m(u,v)<\gamma} I(i+u,j+v))^2}$$

$$\sigma_{T_m} = \sqrt{\frac{1}{n_m} \sum_{V_m(u,v)<\gamma} T_m^2(u,v) - (\frac{1}{n_m} \sum_{V_m(u,v)<\gamma} T_m(u,v))^2}$$

We find the spatial maxima and the corresponding image offsets for each view model:

$$r_m = \max_{i,j} r_m(i,j); \quad (x_m, y_m) = \arg\max_{i,j} r_m(i,j).$$

2.2 Adaptive Model Acquisition

The system begins with one initial model ($M = 1$), specified by the user using a cursor. The target object is then tracked by using the search function given above. When the search

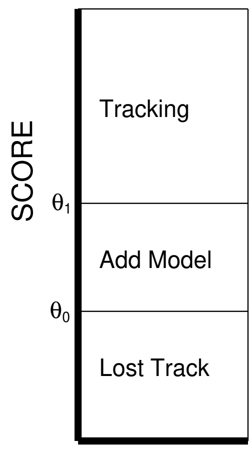

Figure 3: States in adaptive model training as a function of the current maximum model score. When the current score is degraded but still above a minimum "lost-track" threshold, a new model is added using a image centered at the current track position.

function score falls below a certain threshold a new model is added to the search set, using the image at the offset associated with the current best score. Over time, this results in a family of view models that sample the aspect space. Once a fixed set of models has been acquired, the system can update the pixel statistics for each model using a nearest-neighbor refinement rule.

2.2.1 Adding View Models

A new view model is added when the the current maximum correlation score is degraded, but the the system has not completely lost track of the model (i.e., if the object is completely removed from the field of view). A threshold θ_0 establishes the "base" tracking level, below which we assume we have lost track of the object. A second threshold, θ_1, $\theta_1 < \theta_0$ sets the level at which a new model should be added. The full rule is given by:

$$\begin{array}{ll} \text{Let} & m^* = \arg \max_{1 \leq m \leq M} r_m \\ \text{If} & \theta_0 < r_{m^*} < \theta_1 \\ \text{Then} & T_M(i,j) = I(x_{m^*} + i, y_{m^*} + j), \quad 1 \leq i,j \leq \sqrt{n_M} \\ & M = M + 1 \end{array} \tag{2}$$

where n_M is the number of pixels in the initial view model. In the current system the size of the view model is set by the user when specifying the initial model. However, during the refinement phase described below pixel variances are computed and unreliable pixels effectively removed from the model, thus allowing it to have variable size. (Durring model acquisition the model variance is undefined, and is ignored by the search function.)

The relationship of these thresholds is shown diagrammatically in Figure 3. Currently they are set by the user after an initial training phase which establishes the sensitivity of the view model matches. For all of the examples in this paper we used $\theta_0 = 0.6$ and $\theta_1 = 0.7$. We are currently exploring additional techniques to automatically set these thresholds by specifying a desired number of view models.

2.2.2 Refining View Models

Once a fixed number of view models has been found to coarsely interpolate the set of observed images, one can refine the view models by collecting statistics about multiple images of each view. Using a nearest-neighbor update rule, the models can be modified so that they optimally cover the view space in terms of RMS error [4]. The refinement stage functions by first classifying new images as nearest to one of the existing models, and then modifying the pixel statistics for that model. In this way we compute the pixel variances $V_m(u, v)$ used by the search function.

2.2.3 Example of data-driven model acquisition

For simple objects and transformations, this adaptive scheme can build a model which adequately covers the entire space of possible views. For example, for a convex rigid body undergoing a 1D rotation with fixed relative illumination, a relatively small number of view models can suffice to track and interpolate the position of the object at any rotation. Figures 4 illustrates this with a simple example of a rotating box. The adaptive tracking scheme was run with a camera viewing a box suspended by wire. (The axis of rotation was not parallel to any of the three principal axes of the box.) Figure 4a shows the view models in use when the algorithm converged, and all possible rotations were matched with $r_{m^*} > \theta_0$. To demonstrate the tuning properties of each model under rotation, Figure 4b shows the correlation scores for each model plotted as a function of input frame number of a demonstration sequence. In this sequence the box was held fixed at its initial position for the first 5 frames, and then rotated continuously from 0 to 340 degrees. The responses of each model are broadly tuned as a function of object angle, with a small number of models sufficing to represent/interpolate the object at all rotations.

3 Using Temporal Context to guide search

For a static search task, the search rule specified above evaluates the correlation score for each model and picks the maximum. If, however, we are processing a stream of video images, we can exploit temporal correlation between frames of an image sequence to make the search task more efficient.

If the rate of temporal sampling is fast enough, an object can only move a limited amount from frame to frame. Thus we can spatially restrict our search to be within a window around x_{m^*}, y_{m^*}, if our current match score is above the base threshold ($r_{m^*} > \theta_0$). (If we have more prior knowledge about the object being tracked, we can also use a more powerful predictive model incorporating the object dynamics.) If the object is not found

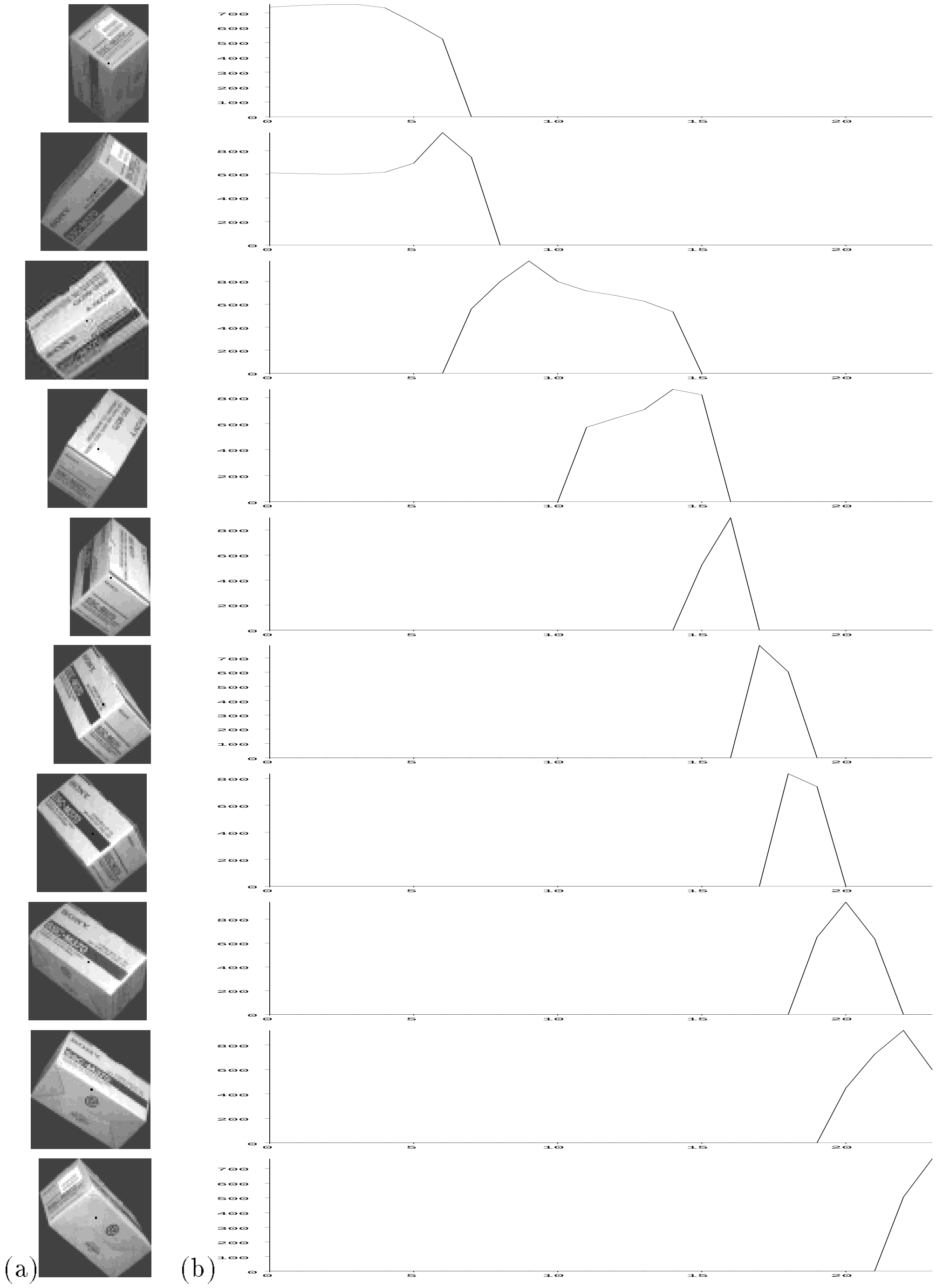

Figure 4: (a) Models automatically acquired from a sequence of images of a rotating box. (b) Normalized correlation scores for each model as a function of image sequence frame number.

within the reduced window, then the current position information is considered invalid and an exhaustive search is conducted over the entire image.

Similarly, by modeling the expected view model score conditional on the current scores, we can predict which views will most likely be seen at the next instant. When a particular view of an object is seen at one instant, not all other views are equally likely to be seen at the next instant (except for certain pathologic objects like a sphere!) We can use this information to order the computation of match scores and/or only compute scores for a subset of the view models.

In essence, we wish to compute the conditional expected score and position of *all* of the view models at time $t+1$, given the current vector of *all* the view model responses at time t: $E(\mathbf{r}[t+1] \mid \mathbf{r}[t])$, where $\mathbf{r} = \{r_1, ..., r_M\}$. To estimate this numerically we represent the vector of responses $\mathbf{r}$ as a discrete set of states. The simplest discretization is to define the match state to be the view model with the current maximum score, $m^*[t] = \arg\max_m r_m[t]$. More complex discretizations are also possible, but we have found this one to be intuitive and give good results.

When the system is in context training mode, it collects the following simple statistical observations:

$$A(m, m^*[t-1]) = A(m, m^*[t-1]) + r_m[t] \qquad \forall m \ s.t. \ r_m[t] > \theta_0$$

$$B(m^*[t-1]) = B(m^*[t-1]) + 1$$

The expected score of the m-th model in the next time step is therefore:

$$E(r_m[t+1] \mid m^*[t]) = \frac{A(m, m^*[t])}{B(m^*[t])}$$

When being run in matching mode (non-training), this information guides the search computation. We search a reduced set of view models expected to exceed a certain score first, and if we find above-threshold scores within this set we return only the subset of their scores and positions. If no match is found in this subset, we evaluate the search scores of the remaining view models, and return the scores for the full set.

4 Non-rigid Objects and Sparse view spaces

When objects are non-rigid, either constructed out of flexible materials or an articulated collection of rigid parts (like a hand), then the dimensionality of the space of possible views becomes much larger. Full coverage of the view space in these cases is usually not possible since enumerating it even with very coarse sampling would be prohibitively expensive in terms of storage and search computation required.

However, many parts of a high dimensional view space may never be encountered when processing real sequences, due to unforeseen additional constraints. These may be physical (some joints may not be completely independent), or behavioral (some views may never be used in the actual communication between user and machine). A major advantage of our adaptive scheme is that it has no difficulty with sparse view spaces, and derives from the data which regions of the space are full.

In the context of developing a user interface, the ability to only model the relevant information can be of critical importance. If there are only a small number of gestures that need to be recognized, then it would be wasteful to model views not actually used by these gestures. To cover completely the possible view-space of a human hand would require a prohibitively large number of view models. However if there are only N gestures to be recognized, each with M frames then at worst we will need to cover a space of $N * M$ views. In general this will require far fewer than $N * M$ view models, since there is almost always some correlation between frames in a sequence which can be exploited.

5 Training Gestures

A gesture can be thought of as a set of views observed over time. Since each view is characterized by the outputs of the view models used in tracking, a gesture can be modeled as the set of model outputs (both score and position) over time. We thus recognize previously trained gestures by collecting the model outputs produced when viewing a novel sequence and comparing them to a library of stored patterns.

To compare a novel sequence of view model outputs to a stored gesture model we must take into account expected variations in those outputs. Our gesture representation incorporates a statistical model of correlation scores, so view models which are less highly correlated with the actual gesture are given less weight in the matching. To account for variation in the overall length of the gesture, we allow the observed sequence to be arbitrarily time-warped to the stored gesture models.

We train a gesture model by providing several temporally segmented examples of the gesture, and computing the mean and variance of the view model correlation scores as a function of time. Each example is time-warped to be the same length as the the longest example before computing statistics on the correlation scores.

5.1 Dynamic Time Warping

The dynamic time warping method involves the use of dynamic programming techniques [1] to solve an elastic pattern matching task. This method was developed to solve the time alignment problem in the speech and signal processing literature [8], and is easily applied to our task. To temporally align two sequences, $\mathbf{r}[t]$, $0 < t < T$, and $\mathbf{r}'[t]$, $0 < t < T'$, using the dynamic time warping (DTW) method, we consider the grid whose horizontal axis is associated with $\mathbf{r}$ and whose vertical axis is associated with $\mathbf{r}'$. Each element of a grid contains a distance measure $D_{i,j}$ measuring the distance between $\mathbf{r}[i]$ and $\mathbf{r}'[j]$. During the gesture training phase we have used euclidean distance for $D_{i,j}$, however other metrics are possible.

The best time warp will minimize the accumulated distance along a monotonic path through the grid from $(0,0)$ to (T,T'). The DTW algorithm uses a partial sum variable, $C_{i,j}$, to recursively compute a minimal solution; $C_{i,j}$ is defined to be the minimum cost to align $\mathbf{r}[0..i]$ with $\mathbf{r}'[0..j]$. Using a simple form of the DTW method, we have

$$C_{i,j} = D_{i,j} + \min(C_{i-1,j-1}, C_{i-1,j}, C_{i,j-1}), \tag{3}$$

where

$$\begin{aligned} C_{1,1} &= D_{1,1} \\ C_{i,1} &= D_{i,1} + C_{i-1,1} \\ C_{1,j} &= D_{1,j} + C_{1,j-1}. \end{aligned}$$

When the recursion is complete, the minimum distance between the two patterns is simply $C_{T,T'}$. To find the optimal path and this the actual alignment we simply backtrack through $C_{i,j}$ along the directions of partial sum minima.

5.2 Classifying time-aligned patterns

Once the view model outputs for each example sequence have been time-aligned, we can build a new gesture model $\mathbf{g}$, by computing the mean values and variance of the model outputs at each time step. Denoting the mean value of view model m at time t for gesture $\mathbf{g}$ as $\tilde{g}_m[t]$, we also find its corresponding variance, $\sigma^2(g_m[t])$.

Given a set of gesture models we detect their presence in a novel sequence by matching each gesture against the most recently observed view model outputs ($\mathbf{r}[t]$). We apply the DTW method to compute the gesture model scores, using a distance function that takes into account the statistics of the gesture examples:

$$D_{i,j} = \sum_{m<M} \frac{1}{\sigma^2(g_m[i])}(\tilde{g}_m[i] - r_m[j])^2 \tag{4}$$

When time-warping gesture models to new observations, we perform the matching backwards in time, so that the current time step is zero for the time-warping algorithm. Thus, both the observed sequence and the gesture pattern must be aligned at the current time step. We relax the requirement that a fixed amount of the observations be used in the time-warped match; the score of a gesture model is defined to to be the minimum of any of the partial sums which account for all of the time samples in the gesture:

$$\mathcal{D}(\mathbf{g}, \mathbf{r}') = \min_{0 \geq t \geq T'} C_{T,t}$$

where T is the number of time samples in $\mathbf{g}$ and T' is the number of time steps in the input buffer $\mathbf{r}'$. Finally, the score function for a given gesture is defined as the inverse distance to the current sequence of observations.

user	*match*	*false alarm*
TD	4	0
AP	4	1
AB	4	0
JB	3	1
LC	4	0
TS	4	0
SS	4	0
total	27	2
possible	28	28
rate	96%	7%

Table 1: Results of "hello" gesture recognition experiment for multiple users; see text for details

6 A Gesture Recognition Example

As an example, we trained our system to recognize the gestures shown in Figure 2. The first gesture was of a person waving "hello" to a camera; in the second gesture a person waved "goodbye". The adaptive training procedure described in section 2 was used to acquire a set of view models that covered (i.e. was able to interpolate) all of the views generated by the two gestures; these are shown in Figure 5.

Figure 6 shows the correlation scores for these models for an example of the "hello" gesture. In this gesture, one can see the high scores oscillating between the view models which contain the rotated versions of the hand. Figure 7 shows the correlation scores for an example of the "bye" gesture; one can see the output oscillating between the view for the extended hand and the view of the compressed hand. Figure 8 shows the gesture data displayed as a surface plot depicting "view-space-time", the pattern of view scores over time.

Figure 9 shows the pattern of view scores for novel sequence which contained three instances of each gesture. At each time step of the observed sequence, the time-warped distance metric was evaluated (in real-time) for each stored gesture pattern. The inverse distance is plotted as a function of frame number next to each of the gesture models. Each gesture in the observed sequence yields a distinct peak in the score of the appropriate gesture model.

We tested the ability of our system to recognize "hello" gestures performed by different users. Each user (from a total of seven, two familiar with the system, five unfamiliar) was asked to provide a training sequence consisting of four examples of the "hello" gesture, interleaved with performing three other gestures ("good bye", "come closer", and "go there" were suggested) and finally keeping the hand stationary for a short period (the null gesture). Temporal segmentation cues for the "hello" examples were provided by the operator, and a gesture model for that user's "hello" was constructed. Each user was then asked to repeat the same sequence of gestures, and the recognition performance was measured. The system produced a “hello” gesture score during the second sequence, and a threshold of 0.3 was used to determine when matches occurred. (None of the system parameters were changed on the system between users.) As shown in Table **??**, the system was able to correctly match all but one of the “hello” gestures while experiencing only 2 false matches.

7 Implementation Issues and Real-time performance

We have implemented the method described above in a hybrid general/special purpose computing environment. Our system has been designed for real-time performance, and all of the examples demonstrated above ran at 10Hz or better (except during the training phase, which has latencies on the order of a second when a new view model is added.)

The majority of the computational burden in this method involves finding the peak view model correlation scores. To compute correlation scores, we rely on a special-purpose image processing computer which is designed for quick correlation searching. The view model acquisition, evaluation, and maxima finding are implemented on a Cognex 4400 vision processor developed by the Cognex Corporation. We have found that for the domain of recognizing hand gestures where the hand fills 1/8th to 1/4th of the frame, it is possible to subsample input images by 4 in both directions (for a final resolution of 128x120) and achieve good recognition performance. We have tested our system at this resolution on examples which used up to 40 models (on an modified version of the box example); the time required to exhaustively search all models in this case was on the order of 200-300ms. (At this resolution our system can store up to 100 view models in memory accessible by the searching hardware.) Using the predictive searching mechanism described in section 3 we were able to reduce the processing time for this example to under than 100ms.

The Cognex 4400 is connected via serial line and shared memory to a host Sun 4/330. The correlation scores are transmitted to the Sun workstation, which stores the gesture models and performs the dynamic time warping procedure. The ($2T^*$) most recent score vectors are buffered, where T^* is the length of the longest sequence in the gesture library. After each frame the DTW scores for each gesture are evaluated, matching backwards in time. Our implementation of DTW on the Sun takes considerably less than 100ms to warp two 20 element sequences, where each element is a vector of 10 model scores. The entire tracking and recognition system can thus perform in real time or near real time for non-trivial tasks.

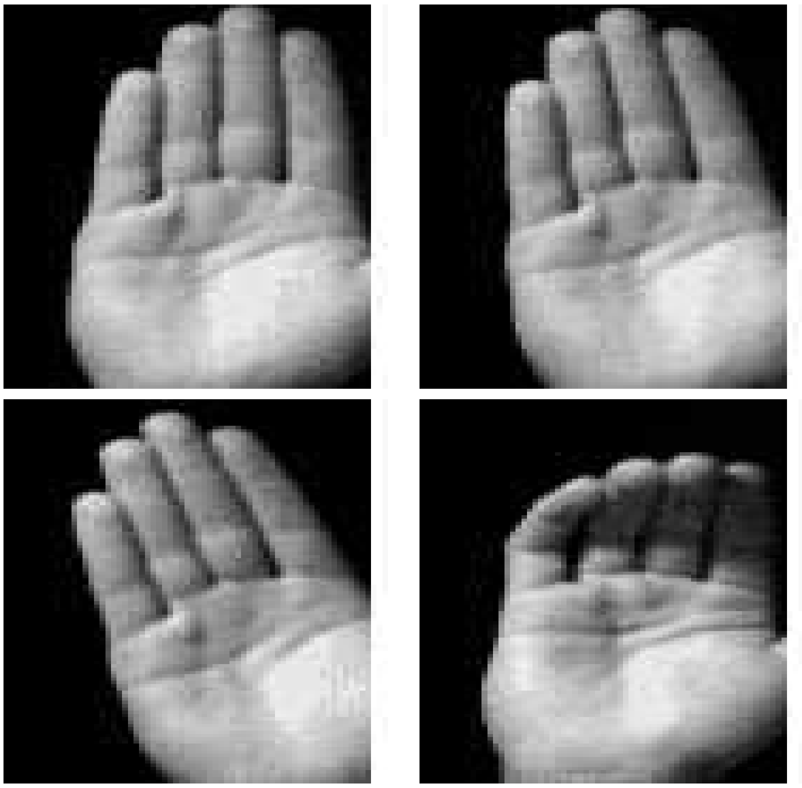

Figure 5: Four view models used in recognizing the gestures shown in Figure 2. The first model was specified with a window provided by the user, and the remaining models were adaptively acquired from a training sequence containing both gestures.

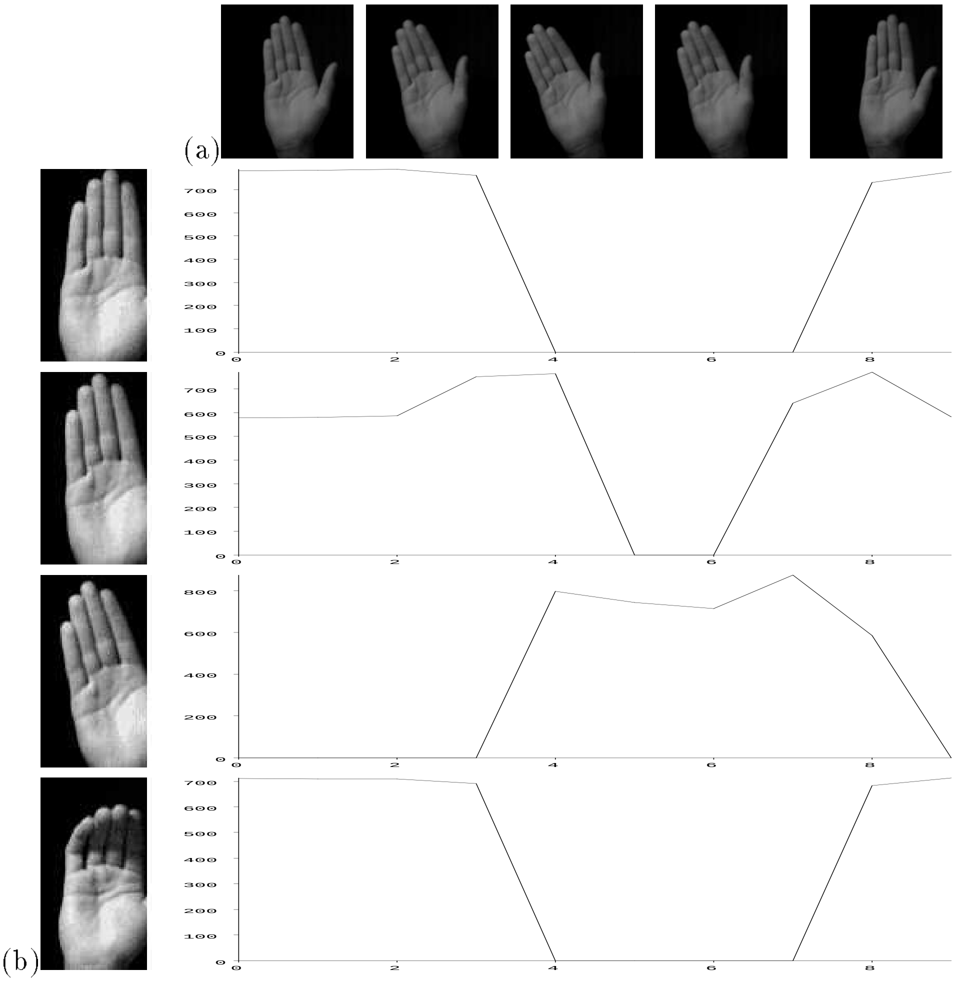

Figure 6: Training the "hello" gesture. (a) shows samples from sequence of frames used to train the gesture. (b) shows the scores for each of the model views as a function of frame number.

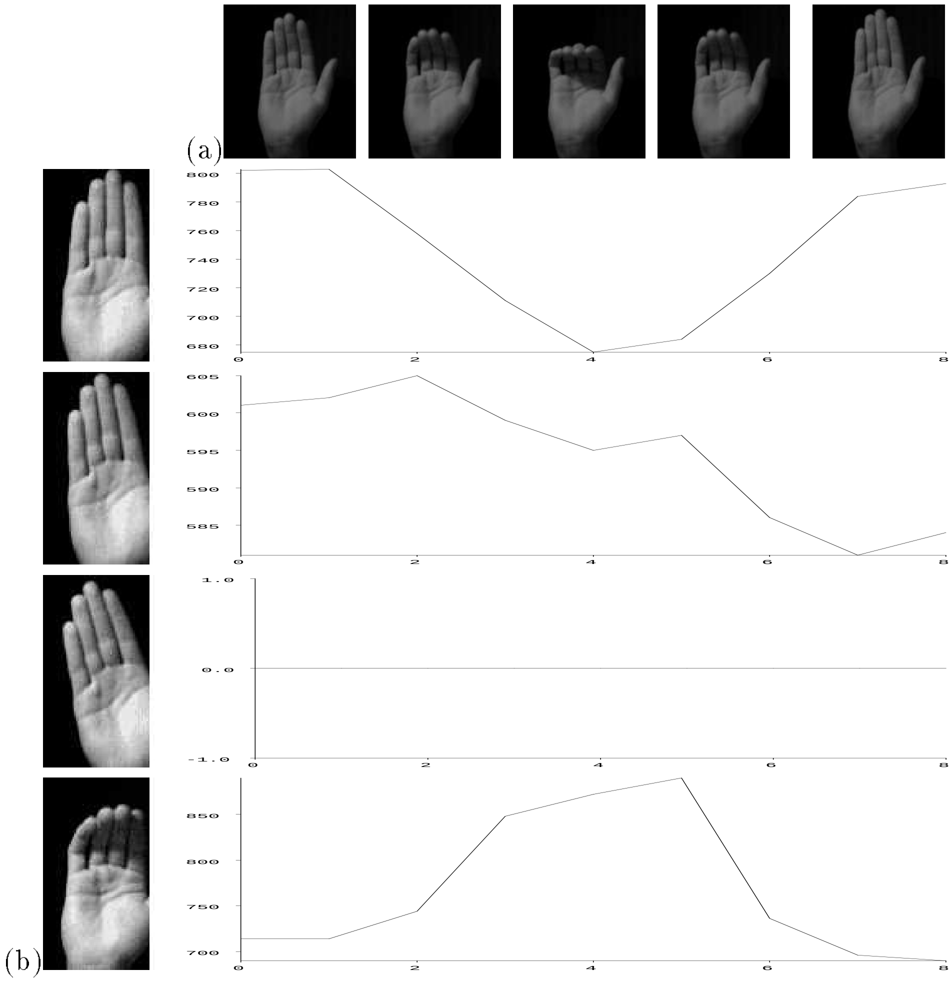

Figure 7: Training the "bye" gesture. (a) shows samples from sequence of frames used to train the gesture. (b) shows the scores for each of the model views as a function of frame number.

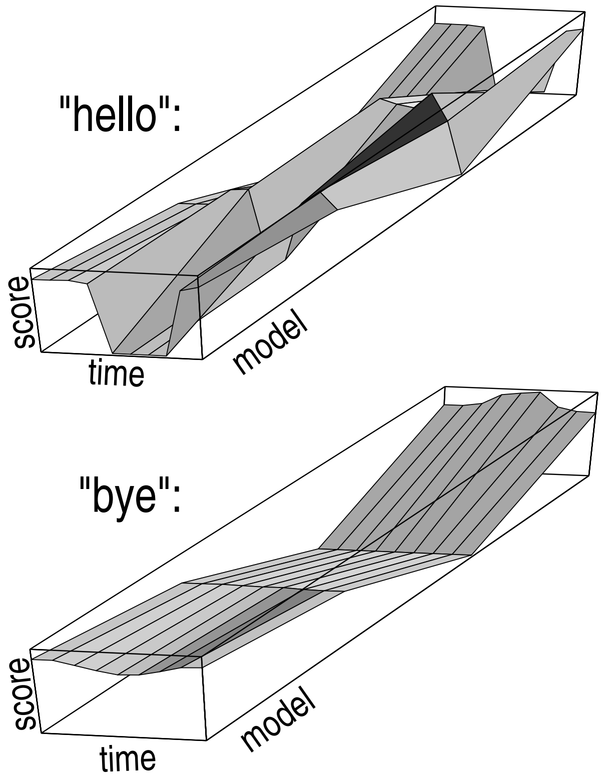

Figure 8: Surface plots of gesture models using data from Figures 6 and 7.

8 Summary and Conclusions

We have developed and implemented a real-time system for learning, tracking, and recognizing complex object and gestures. The use of a view-based representation allows us to model (and search) only the portion of an object's view-space which is actually used by a user in the gestures to be recognized. Capturing the temporal correlation of view model scores is an important portion of this work, since it can reduce the amount of computation that needs to be performed by an an order of magnitude. Together with the Cognex hardware described above, this allows us to achieve real-time performance when tracking objects with many views.

Modeling gestures as a set of view scores over time allows us to reduce the dimensionality of the space-time pattern without having to recover exactly the underlying object pose parameters. Using dynamic time warping and a statistical recognition approach, real-time recognition of human gestures is possible using computational equipment of modest power.

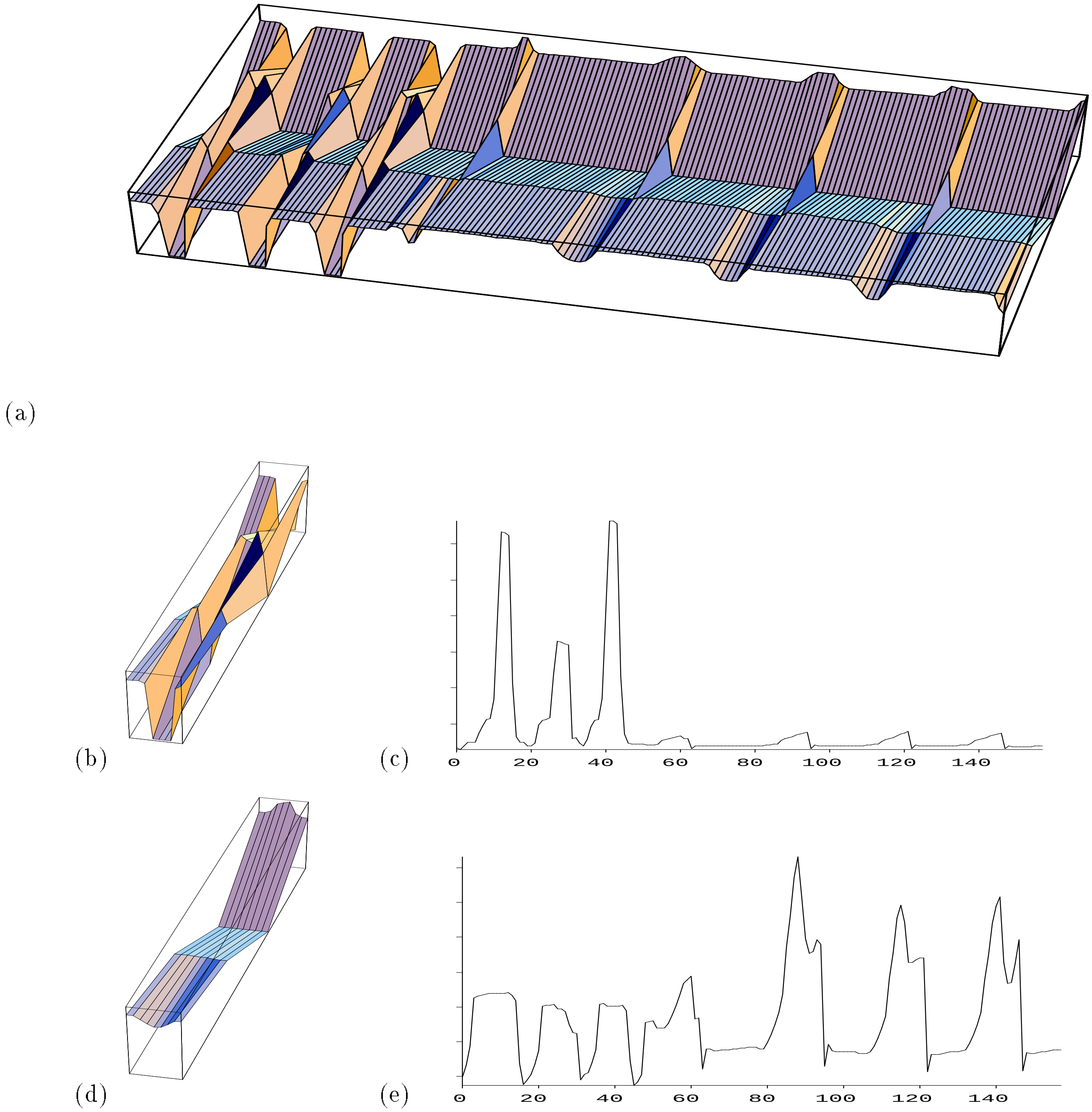

Figure 9: Matching the "hello" and "bye" gestures against observed data. (a) shows the output of the view models for a sequence in which the user waves "hello" three times and then "bye" three times. (b) "hello" gesture model; (c) gesture score as a function of time. (d) "hello" gesture model; (e) gesture score as a function of time. Three peaks are evident in each score plot, indicating a match for each of the gestures.

References

[1] Bellman, R. E., (1957) *Dynamic Programming.* Princeton, NJ: Princeton Univ. Press.

[2] Bolt, R. (1984) "The Human Interface, where people and computers meet," *Lifetime Learning Publications*, Belmont, California.

[3] Ishibuchi, K., Takemura, H., and Kishino, F., "Real Time Hand Shape Recognition using Pipe-line Image Processor", (1992) *IEEE Workshop on Robot and Human Communication*, pp. 111-116

[4] Makhoul, J., Roucos, S., and Gish, H., (1985) "Vector Quantization in Speech Coding" *Proc. IEEE*, Vol. 73, No. 11, pp. 1551-1587

[5] Mase, K., (1991) "Recognition of Facial Expression from Optical Flow", *ICICE Transactions*, Vol. E 74, No. 10, pp. 3474-3483

[6] Negroponte, N.,(1980) "The Architecture Machine," *M.I.T. Press*, Cambridge.

[7] Poggio, T., and Edelman, S., (1990) "A Network that Learns to Recognize Three Dimensional Objects," *Nature*, Vol. 343, No. 6255, pp. 263-266

[8] Sakoe, H., and Chiba, S., (1980) "Dynamic Programming optimization for spoken word recognition", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. 26, pp. 623-625

[9] Sperling, G., Landy, M. S., Cohen, Y. and Pavel, M. (1985). "Intelligible encoding of ASL image sequences at extremely low information rates," *Computer Vision, Graphics, and Image Processing*, Vol. 31, pp. 335-391.

[10] Torige, A., and Kono, T., "Human-Interface by Recognition of Human Gesture with Image Processing Recognition of Gesture to Specify Moving Direction", (1992) *IEEE Workshop on Robot and Human Communication*, pp. 105-110

[11] Ullman, S., and Basri, R., (1991)"Recognition by Linear Combinations of Models," *IEEE PAMI*, Vol. 13, No. 10, pp. 992-1007

[12] Yamato, J., Ohya, J., Ishii, K., (1992) "Recognizing Human Action in Time-Sequential Images using Hidden Markov Model", *IEEE Proc. CVPR-92*, pp. 379-385.