To appear, *IEEE Conference on Face and Gesture Recognition*, March 2000.

Gesture Modeling and Recognition Using Finite State Machines

Pengyu Hong*, Matthew Turk[+], Thomas S. Huang*

* Beckman Institute
University of Illinois at Urbana
Champaign, 405 N. Mathews
Urbana IL 61801, USA
{hong, huang}@ifp.uiuc.edu

[+]Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399, USA
mturk@microsoft.com

Abstract

This paper proposes a state based approach to gesture learning and recognition. Using spatial clustering and temporal alignment, each gesture is defined to be an ordered sequence of states in spatial-temporal space. The 2D image positions of the centers of the head and both hands of the user are used as features; these are located by a color based tracking method. From training data of a given gesture, we first learn the spatial information without doing data segmentation and alignment, and then group the data into segments that are automatically associated with information for temporal alignment. The temporal information is further integrated to build a Finite State Machine (FSM) recognizer. Each gesture has a FSM corresponding to it. The computational efficiency of the FSM recognizers allows us to achieve real-time on-line performance. We apply the proposed technique to build an experimental system that plays a game of "Simon Says" with the user.

1. Introduction

Gestures are expressive and meaningful body motions used in daily life as a means of communication. Automatic gesture recognition systems using computer vision techniques may be useful in many contexts, including non-obtrusive human computer interfaces. For such environments, it is important that the systems are easily trained and fast enough to support interactive behavior. In this paper, we present a technique for gesture modeling and recognition, developed to support these requirements. The system is used to support a game of *Simon Says*, where the computer plays the role of Simon, issuing commands and checking to see if the user complies.

We use a simple feature set as input to the gesture modeling and recognition: 2D points representing the centers of the user's head and hands in the image sequence. These are obtained by a real-time skin-color tracking method. The tracking is relatively insensitive to variations in lighting conditions. The trajectories of the hands relative to the head position provide translation-independent input, and our initialization procedure allows scale independence as well.

Learning and recognizing even 2D gestures is difficult since the position data sampled from the trajectory of any given gesture varies from instance to instance. There are many reasons for this, such as sampling frequency, tracking errors or noise, and, most notably, human variation in performing the gesture, both temporally and spatially. Many conventional gesture modeling techniques require labor-intensive data segmentation and alignment work. We desire a technique to help segment and align the data automatically, without involving exhaustive human labor. At the same time, the representation used by the method should capture the variance of the gestures in spatial-temporal space.

Toward this goal, we modeled gestures as sequences of states in spatial-temporal space. Each state is modeled as a multidimensional Gaussian. The gesture recognition model is represented by a Finite State Machine (FSM) and built by the following procedures. We assume that the trajectories of a gesture are set of points distributed spatially. The distribution of the data can be represented by a set of Gaussian spatial regions. A threshold is selected to represent the spatial variance allowed for each state. These thresholds determine the spatial variance of the gesture. The number of the states and their coarse spatial parameters are calculated by dynamic k-means clustering on the training data of the gesture *without* temporal information.

Training is done offline, using perhaps several examples of each gesture, repeated continuously by the user when requested, as training data. After learning the spatial information, data segmentation and alignment become easy. The temporal information from the segmented data is then added to the states. The spatial

information is also updated. This produces the state sequence that represents the gesture. Each state sequence is a FSM recognizer for a gesture.

The recognition is performed online at frame rate. When a new feature vector arrives, each gesture recognizer decides whether to stay at the current state or to jump to the next state based on the spatial parameters and the time variable. If a recognizer reaches it final state, then we say a gesture is recognized. If more than one gesture recognizer reach their final states at the same time, the one with minimum average accumulate distance is chosen as the winner.

In the remainder of this paper, we discuss related work (Section 2), details of the gesture modeling and recognition (Section 3), the example application (Section 4), and end with conclusions and discussion (Section 5).

2. Related work

Since the Moving Light Display experiments by Johansson [1] suggested that many human gestures could be recognized solely by motion information, motion profiles and trajectories have been investigated to recognize human gestures.

Bobick and Wilson [2] proposed a state-based technique for the representation and recognition of gesture. In their approach, a gesture is defined to be a sequence of states in a configuration space. The training gesture data is first reduced to a prototype curve through configuration space, and the prototype curve is parameterized only according to arc length. The prototype curve is manually partitioned into regions according to spatial extent and variance of direction. Each region of the prototype is used to define a fuzzy state representing traversal through that phase of the gesture. Recognition is done by using the dynamic programming technique to compute the average combined membership for a gesture.

Davis and Shah [3] proposed a method to recognize human-hand gestures using a model-based approach. A finite state machine (FSM) is used to model four qualitatively distinct phases of a generic gesture: (1) static start position, for at least three frames; (2) smooth motion of the hand and fingers until the end of the gesture; (3) static end position, for at least tree frames; (4) smooth motion of the hand back to the start position. Gestures are represented as a list of vectors and are then matched to the stored gesture vector models using table lookup based on vector displacements.

McKenna and Gong [4] modeled gestures as sequences of visual events. Given temporally segmented trajectories, each segment is associated with an event. Each event is represented by a probabilistic model of feature trajectories and matched independently with its own event model by linear time scaling. A gesture is thus a piecewise linear time warped trajectory. The event duration probability density function (p.d.f.) is estimated from the training examples as either a Gaussian or a uniform distribution over the interval. Gesture recognition is performed using a probabilistic finite state (event) machine. State transitions depend on both the observed model likelihood and the estimated state duration p.d.f.

HMMs have been used extensively in visual gesture recognition recently [5,6,7,8,9]. HMMs are trained on data that is temporally well-aligned. Given the sample data of a gesture trajectory, HMMs use dynamic programming to output the probability of the observation sequence. The maximum probability is compared with a threshold to decide if a gesture is recognized. In the conclusion of this paper, we will briefly discuss the relation between our technique and HMMs.

3. FSMs for gesture recognition

3.1. Gesture modeling

The features that we compute from the input video images, and use for input to gesture modeling and recognition, are the 2D positions of the centers of the user's face and hands. These are acquired from a real-time skin-color tracking algorithm, similar to the methods proposed by Yang [10] and others. The data obtained from our tracking implementation is noisy. Thus we do not use the speed, the direction of motion, or the area of the skin as features. This is also a motivation for using the following representation.

Basically, a gesture is defined as an ordered sequence of states in the spatial-temporal space. Each state S has parameters $< \vec{\mu}_s, \Sigma_s, d_s, T_s^{\min}, T_s^{\max} >$ to specify the spatial-temporal information captured by it, where $\vec{\mu}_s$ is the 2D centroid of a state, Σ_s is the 2x2 spatial covariance matrix, d_s is the distance threshold, and $[T_s^{\min}, T_s^{\max}]$ is a duration interval. The spatial-temporal information of a state and its neighbor states specifies the motion and the speed of the trajectory within a certain range of variance. In the training phase, the function of the states is to help to segment the data and temporally align the training data automatically.

3.2. Computing the gesture model

Without appropriate alignment of the training data, it is very difficult to learn the spatial-temporal information at the same time if the data is not well behaved in time. To solve this problem, we decouple the temporal information from the spatial information, roughly learn the spatial information, incorporate the temporal information, and then refine the spatial information. The training data is captured by observing a gesture repeated several times

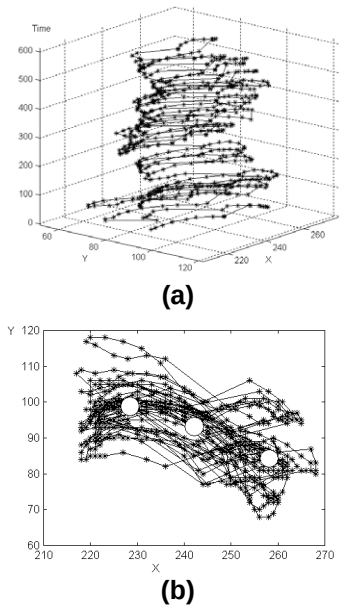


Figure 1. "Wave left hand" gesture. (a) With temporal information. (b) Without temporal information.

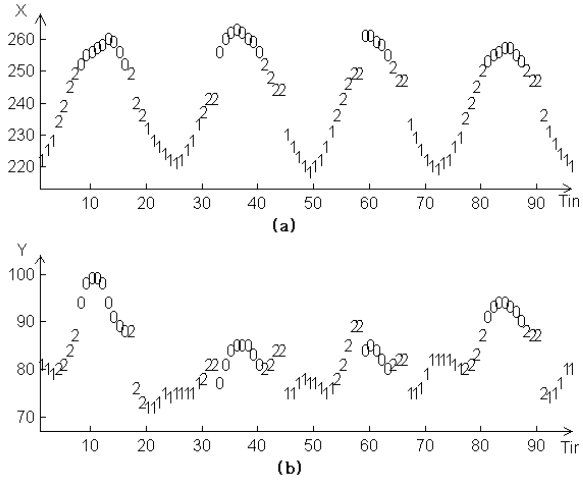


Figure 2. State sequence correspondent to part of the data shown in Figure 1. (a) The state sequence plotted by x position along the time. (b) The state sequence plotted by y position along the time.

continuously. The learning is divided into two phases as follows.

In the first phase (spatial clustering), we learn the distribution of the data distributions without temporal information. An example of data from a "wave left hand" gesture is shown in Figure 1. A threshold is defined to specify the coarse spatial variance allowed for a gesture. Currently this threshold is fixed, although it could eventually be computed from the data and prior information about the user and/or the gesture. Each state *S* indicates a region in the 2D space that is represented by the centroid $\vec{\mu}_s = E(\vec{x})$ and the covariance matrix $\Sigma_s = E((\vec{x} - \vec{\mu})^T (\vec{x} - \vec{\mu}))$, where $\vec{x} = (x, y)$. Given input data $\vec{x}$, the distance from the data to the state *S* is defined as the Mahalanobis distance:

$$D(\vec{x}, S) = \sqrt{(\vec{x} - \vec{\mu}_s)\Sigma_s^{-1}(\vec{x} - \vec{\mu}_s)^T}$$

At the beginning, we assume that the variance of each state is isotropic. Beginning with a model of two states, we train the model to represent the data using dynamic *k*-means. Whenever the error improvement is very small, we split the state with the largest variance that is higher than the chosen threshold. The training stops after all the variances of the states drop below the threshold. For example, after training, the data in Figure 1 has three states whose centroids are represented by the white circles in Figure 1(b).

In the second phase (temporal alignment), each data point is assigned a label corresponding to the state to which it belongs. Thus we get a state sequence corresponding to the data sequence, as illustrated in Figure 2. By manually specifying the temporal sequence of states from the gesture examples, we obtain the structure of the Finite State Machine (FSM) for the gesture. For example, the state sequence for one cycle of the "wave left hand" gesture, shown in Figure 2, is [1 2 0 2 1]. Once this is determined, the training data is segmented into gesture samples. A sample of [1 1 1 2 2 2 2 0 0 0 0 2 2 2 1 1], for example, consists of the five states with (3, 4, 4, 3, and 2) samples per state, respectively. The number of samples in a state is proportional to the duration of the state. The example gestures are all aligned in this manner, resulting in N examples with the same state sequence, differing only in the number of samples per state.

The duration interval $[T_s^{\min}, T_s^{\max}]$ is currently set by calculating the minimum and maximum number of samples per state over the training data. Since the user may stay at the first state of the FSM for an indefinite period of time, we set $T_0^{\max}$ to be infinite.

Once the data alignment is done, an HMM could be trained by the data. However, we instead use the simple FSM to model the spatial-temporal information of the gestures. The temporal information of each state is represented by a duration variable whose probability is modeled as uniform over a finite interval $[T_s^{\min}, T_s^{\max}]$. The duration variable tells approximately how long the trajectory should stay at a certain state.

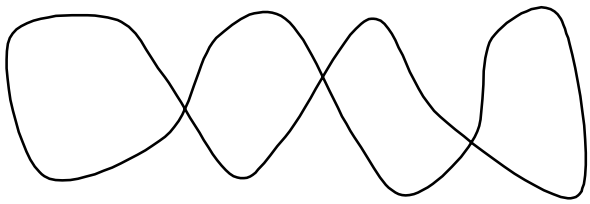


Figure 3. A gesture containing loops with multiple intersections.

For the data subset belonging to a state *S*, the mean *m* and the variance σ^2 of the distance of the data to the center of the state are calculated. The threshold d_s of the state *S* is set to be $m + k\sigma$.

A major advantage of the method is that it handles gestures which with different lengths (a different number of states). Our method quickly produces a recognizer for different gestures by specify only the variance, even when only a few training examples are available. Potentially, our approach is able to handle gestures with trajectories that contain loops with more than one intersection, such as the one shown in Figure 3.

3.3. Recognition

Real-time, online recognition is done by considering only the data acquired at the current time point. A gesture is recognized when all the states of a gesture recognizer are passed. This is different from the traditional approaches, which require that the data segment provided to the recognizer contains the complete gesture data. Although we only examine the data sample at current time point, we do use the context information stored in the FSM for recognition. The context information of a gesture recognizer *g* can be represented as:

$$c = < s_k, t >$$

where s_k is the current state of the recognizer *g*, *t* is the how long the recognizer has stayed at s_k. Since a FSM is an ordered state sequence, s_k stores the history of the trajectory.

When a new data sample *x* comes, if one of the following conditions is met, the state transition happens.

$$(1)\ (D(x, s_{k+1}) \le d_{k+1}) \,\&\, (t > t_k^{\max})$$

$$(2)\ (D(x, s_{k+1}) \le d_{k+1}) \,\&\, (D(x, s_{k+1}) \le D(x, s_k)) \,\&\, (t \ge t_k^{\min})$$

$$(3)\ (D(x, s_{k+1}) \le d_{k+1}) \,\&\, (D(x, s_k) \ge d_k)$$

The recognizer only takes into account the current data sample, since the past is modeled by the current state. Each state *S* has its own threshold d_s. If the new data *x* does not belong to the current state and the state transition cannot happen, the recognizer is reset, i.e., that particular FSM is eliminated from consideration. Thus the computation complexity at each time point is approximately O(n) where n is the total number of the FSM models.

If a data sample happens to make more than one gesture recognizer fire, there is an ambiguity. To resolve the ambiguity, we choose the gesture with the minimum average accumulate distance:

$$Gesture = \arg\min_g \left(\sum_{i=1}^{n_g} D(\vec{x}_i, s_{gi}) / n_g \right)$$

where s_{gi} is the state of the gesture *g* that the data sample $\vec{x}_i$ belongs to, n_g is the number of the data accepted by the recognizer of the gesture *g* up to the current time point.

Figure 4. Tracking screenshot during initialization.

4. "Simon Says" application

This approach to gesture representation and recognition was motivated by the desire to support real-time interactive systems. As an example of such an application, we built a system to play the game *Simon Says* with a user. In *Simon* Says, an on-screen character requests that the user make a particular gesture, and then checks to see if the user complies. This can be implemented as verification (is the specified gesture being done?) or as more general recognition (which of *N* gestures is the user doing?).

Our prototype system does not yet have an embodied Simon – rather, Simon's requests and evaluations are communicated via text and sound. The system runs at frame rate on a 350 MHz Pentium II system running Windows NT 4.0. Figure 4 shows a screen shot of the tracking window during initialization, in which the user is asked to hold his hands out to the side and above his head.

We tested the application using some simple gestures, such as left hand wave, right hand wave, drawing a circle, and drawing a figure eight (∞). Some examples are given in Figures 5, 6, and 7. Figure 5 shows data and the resulting FSM for waving the left hand. Figure 6 shows data and the resulting FSM for drawing a circle. Likewise, Figure 7 shows data and the resulting FSM for drawing a figure eight. In each of these figures, the data is plotted with labels of the state overlapped on the data points.

A left hand wave gesture is recognized only when the left hand moves through the 2D spatial regions represented by the state sequence [0, 2, 1, 2, 0], with the duration of each state falling in the allowable range of $[T_i^{\min}, T_i^{\max}]$. Any movement that violates these requirements will not be accepted as left hand wave. One drawback of this approach is that a very noisy data sample will cause the recognition to fail. In practice, we introduce a variable *OD* to the FSM to handle this situation of variable or noisy input. If a data sample does not fit the current state of a FSM model, the model stays at the current state and increase the value of *OD*. If the value of *OD* is bigger than a fixed, small threshold, then the FSM model is reset. This heuristic takes into account brief movement errors, short-term tracking failure, and mildly noisy tracking results.

5. Conclusions and discussion

We have developed a technique for gesture modeling and recognition in real-time, interactive environments. The training data consists of tracked 2D head and hand locations, captured while performing each repeatedly. The spatial and temporal information of the data are first decoupled. In the first phase, the algorithm learns the distribution of the data without temporal information via dynamic *k*-means. The result of the first phase provides support for data segmentation and alignment. The temporal information is then learned from the aligned data segments. The spatial information is then updated. This produces the final state sequence, which represents the gesture. Each state sequence is a FSM recognizer for a gesture. The technique has been successfully tested on a set of gestures, e.g., waving left hand, waving right hand, drawing a circle, drawing a figure eight, etc.

There are similarities between HMMs and our approach. One difference is that with HMMs the number of states and the structure of the HMM must be predefined. To train a HMM, well-aligned data segments are required. The FSM method we proposed segments and aligns the training data and simultaneously produces the gesture model. We are currently pursuing an unsupervised model construction method to make the training simpler and better.

During the recognition phase of an HMM, the system takes a segment of data as input, calculates the combined probability of the membership, compares the probability with a threshold, and decides whether the data is accepted or rejected. In our approach, since each state is associated with a threshold that is learned from the data, recognition is done based on the data at current point in time and the context information that is stored in the FSM. This dramatically reduces the computation complexity.

The *Simon Says* application is an interesting domain in which to test gesture recognition – real-time demands are balanced with a cooperative user and a controlled context.

References

[1] G. Johansson, Visual Perception of Biological Motion and a Model for Its Analysis, *Perception and Psychophysics*, vol. 14, no. 2, pp.201-211. 1973.

[2] Aaron F. Bobick and Andrew D. Wilson. A State-Based Approach to the Representation and Recognition of Gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 12 Dec. 1997.

[3] James Davis and Mubarak Shah. Visual Gesture Recognition. *Vision, Image and Signal Processing*, 141(2), 1994, pp. 101-106.

[4] Stephen J. McKenna and Shaogang Gong. Gesture Recognition for Visually Mediated Interaction using Probabilistic Event Trajectories. *The Ninth British Machine Vision Conference*, Sep. 1999.

[5] J.L. Yamato, J. Ohya and K. Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *Proc. Conf. on Computer Vision and Pattern Recognition*, Champaign, IL 1992, pp. 379-385.

[6] J. Schlenzig, E. Hunter, and R. Jain. Recursive Identification of Gesture Inputs Using Hidden Markov Models. In *Proc. Second Annual Conf. Applications of Computer Vision*, pp. 187-194, Dec. 1994.

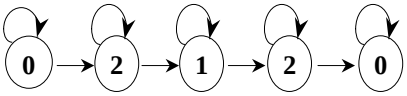
[7] T.E. Starner and A. Pentland. Visual Recognition of American Sign Language Using Hidden Markov Models. In *Proc. Int'l Workshop Automatic Face and Gesture Recognition*, Zurich, 1995.

[8] J.M. Siskind and Q. Morris. A maximum-likelihood approach to visual event classification. In *Proceedings of the Fourth European Conference on Computer Vision*, pp. 347-360, 1996.

[9] M.H. Yang and N. Ahuja, Recognizing Hand Gesture Using Motion Trajectories. CVPR99.

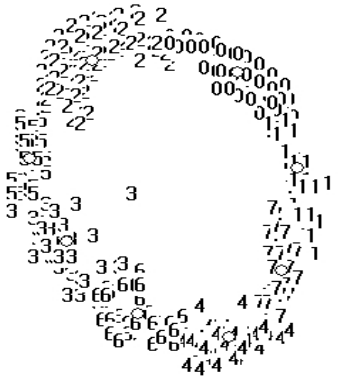
[10] J. Yang and A. Waibel. A real-time face tracker. In *Proceedings of the Third IEEE Workshop on Applications of Computer Vision*, pp. 142-147, 1996.

(a)

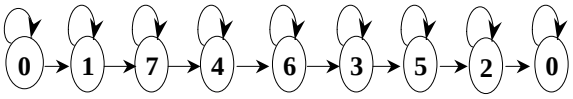


(b)

Figure 5. "Wave left hand" gesture. (a) Data. (b) The corresponding FSM.

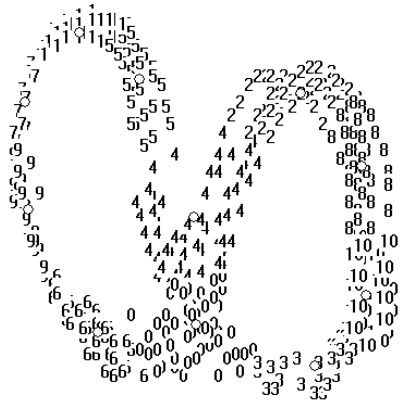


(a)

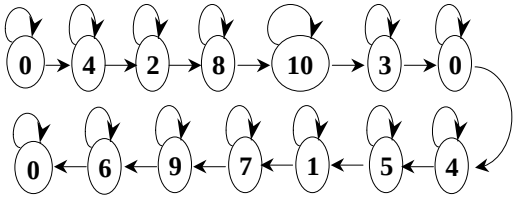


(b)

Figure 6. "Drawing a circle" gesture. (a) Data. (b) The corresponding FSM.



(a)



(b)

Figure 7. "Drawing a figure 8" gesture. (a) Data. (b) The corresponding FSM.