# Constructing Finite State Machines for Fast Gesture Recognition

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

[*]*Beckman Institute*
*University of Illinois at Urbana*
*Champaign, 405 N. Mathews*
*Urbana IL61801, USA*
*{hong, huang}@ifp.uiuc.edu*

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

## Abstract

*This paper proposes an approach to 2D gesture recognition that models each gesture as a Finite State Machine (FSM) in spatial-temporal space. The model construction works in a semi-automatic way. The structure of the model is first manually decided based on the observation of the spatial topology of the data. The model is refined iteratively between two stages: data segmentation and model training. Given the continuous training data of a single gesture, we roughly segment the gesture trajectory into phrases using the spatial information alone. The segmentation results are used to initialize an FSM. The model is used to re-segment the data. The results of the re-segmentation are used to refine the parameters of the model. After the FSM is trained, we incorporate a modified Knuth-Morris-Pratt algorithm into the FSM recognition procedure to speed up the gesture recognition. The computational efficiency of the FSM recognizers allows real-time on-line performance to be achieved.*

## 1. Introduction

Non-obtrusive human computer interfaces demand real-time automatic gesture recognition systems using computer vision techniques. Early research on moving light display experiments suggested that many human gestures could be recognized solely by motion information. Motion profiles and trajectories have been investigated to recognize human gestures. Recently, HMMs have been used extensively in visual gesture recognition [1,2]. HMMs are trained on data that is temporally well aligned. Given a new gesture trajectory, HMMs use dynamic programming to recognize the observation sequence.

Recently, some state-based approaches have been proposed for gesture modeling and recognition. The advantage of a state approach is that it doesn't need a large set of data in order to train the model. Bobick and Wilson [3] proposed an approach that models a gesture as a sequence of states in a configuration space. The training gesture data is first manually segmented and temporally aligned. A prototype curve is used to represent the data, and is parameterized according to a manually chosen arc length. Each segment of the prototype is used to define a fuzzy state representing traversal through that phase of the gesture. Recognition is done by using dynamic programming technique to compute the average combined membership for a gesture.

Davis and Shah [4] used finite state machine to model four qualitatively distinct phases of a generic hand gesture: (1) static start position, (2) smooth motion of the hand and fingers during the gesture, (3) static end position, and (4) smooth motion of the hand back to the start position. They represented gestures as a list of vectors that are then matched to the stored gesture vector models using table lookup based on vector displacements.

McKenna and Gong [5] modeled gestures as sequences of visual events, each represented by a probabilistic model of pre-segmented feature trajectories. Gesture recognition is performed using a probabilistic finite state machine. State transitions depend on both the observed model likelihood and the estimated state duration p.d.f.

Learning and recognizing even 2D gestures is difficult since the 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. The above gesture modeling techniques require labor-intensive data segmentation and alignment work. We desire a technique to help segment and align the data, without involving exhaustive human labor. The representation used by this method can be immediately used to build recognition models.

Toward this goal, we modeled gestures as sequences of states in spatial-temporal space [6]. Each state is modeled as a multivariate Gaussian. The gesture recognition model is represented by an FSM. A threshold is pre-defined to allow a certain degree of spatial variance in each state. The number of the states is then calculated by dynamic k-means clustering on the training data of the gesture *without* temporal information. The spatial and the temporal information of the gesture are learned together iteratively. The recognition is performed online at frame

rate. In this paper, we improve the above method by proposing a semi-automatic model construction method and a modified *Knuth-Morris-Pratt* algorithm for fast gesture recognition.

The rest of the paper is organized as follows. Section 2 presents the model representation and the iterative procedure for model construction and training. Section 3 shows how to utilize the *Knuth-Morris-Pratt* algorithm to speed up the recognition procedure. Experimental results are shown in Section 4, and we conclude in Section 5.

## 2. Modeling Gestures with FSMs

### 2.1. Gesture modeling

The centroids of the 2D positions of the head and hands are acquired from the live input video images by color tracking, normalized, and used as features (as described in [6]). A gesture is defined as an ordered sequence of states in spatial-temporal space and modeled by an FSM (Figure 1). The structure of an FSM is like that of an HMM. Each state can jump to either itself or its next state.

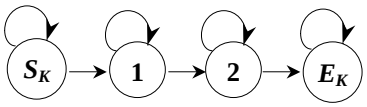


**Figure 1**. The FSM of a gesture with 4 states < $S_K$, 1, 2, $E_K$ >

Each state $s_i$ has parameters $< \vec{\mu}_i, \Sigma_i, d_i, T_i^{\min}, T_i^{\max} >$ to specify the spatial-temporal information captured by it, where $\vec{\mu}_i$ is the 2D spatial centroid of a state, $\Sigma_i$ is the 2x2 spatial covariance matrix, $d_i$ is a spatial threshold, the temporal parameter $[T_i^{\min}, T_i^{\max}]$ 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.

### 2.2. Model Initialization

In the rest of the paper, we assume that the training data sequence contains only a single gesture recurring continuously. Some random data samples are allowed between two consecutive corpora if the start position and the end position of a gesture are not the same. A semi-automatic approach is proposed to learn the spatial-temporal information iteratively. First, we learn the spatial distribution of data without temporal information. A variance of dynamic local K-means [7] is used to learn the spatial parameters of the states (classes) without temporal information. *Mahalanobis* distance between a sample and the spatial center of a state is used as similarity measurement. The spatial variance of the state is

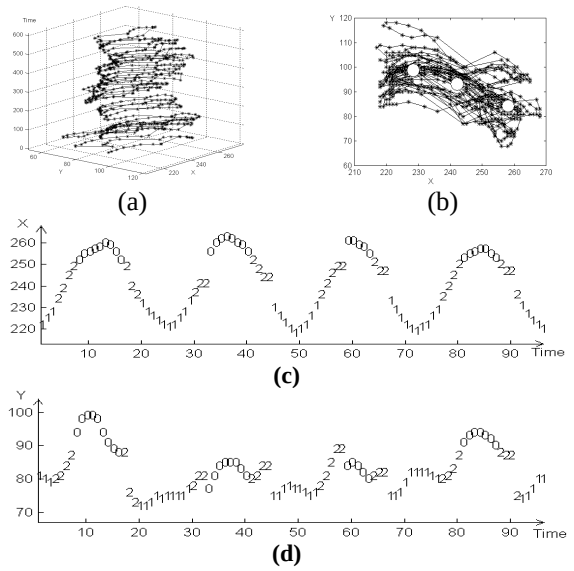


**Figure 2.** The training data of "wave left hand". (a) Shown in temporal-spatial space; (b) Shown in spatial space (vertical axis is time axis); (c) The state sequence plotted by x position along the time; (d) The state sequence plotted by y position along the time.

calculated as the distance variance. A constraint is superimposed: the spatial variance of each state should be less than $\sigma_0$. The parameter $\sigma_0$ depends on the applications. The dynamic k-means algorithm begins with 1 class and stops when the variances of all the classes are smaller than $\sigma_0$.

For example, Figure 2 (a) shows the trajectory of the training data of the "wave left hand" gesture. The dynamic local k-means produces 3 states to cover the spatial data. The centroids of the states are represented by the white circles in Figure 2(b). Each signal sample is assigned a label corresponding to the state to which it belongs. Hence, we get a label sequence corresponding to the signal sequence. The label sequence is shown in Figure 2(c, d). The adjacent signals are grouped together if they have the same label. Each group defines a state. This produces a state sequence for the training data. The structure of this gesture's FSM is manually defined.

Assume a gesture model consist of the state sequence: ($s_1$, $s_2$, …, $s_n$). Two cases are considered for initializing and refining the parameters of the states: (1) The label of $s_1$ is different from that of $s_n$; (2) The label of $s_1$ is same as that of $s_n$. For case 1, we consider the state sequence of the FSM to be a string $S_1$ and the state sequence of the training data as another string $S_2$. We search all the sub-strings of $S_2$ that match with $S_1$. The data segments corresponding to those sub-strings are the corpora of the gesture that are roughly temporally aligned. In this way, the training data are segmented. The parameters $< \vec{\mu}_i, \Sigma_i >$ of a state $s_i$ in the FSM can be initialized by $\vec{\mu}_s = E[\vec{x}]$ and $\Sigma_s = E[(\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T]$, where $\vec{x}$ is the

data sample belonging to state $s_i$ and $d_i$ is set as $3\sigma_0$. The temporal parameters of the states are left open to be decided later (see Section 2.3).

For case 2, only the state sequence $s_2, \ldots, s_{n-1}$ is used for string matching. The central part of the gesture corpora is found and used to initialize the spatial parameter of $s_2, \ldots, s_{n-1}$. We first build a temporary FSM that only contains the states $s_2, \ldots, s_{n-1}$. Later on, states $s_1$ and $s_n$ will be added during modeling refining. The temporal parameters of $s_2, \ldots, s_{n-1}$ and all the parameters of $s_1$ and $s_n$ will be left open to be decided later (see Section 2.3).

### 2.3. Model refinement

A procedure similar to Segmental k-Means [8] is used to refine the model. It performs recognition and retraining iteratively. The recognition is equivalent to segmenting the training data sequence into a state sequence. The segmentation results are used to retrain the parameters of the states.

In the recognition phrase, the training data sequence is fed to the FSM continuously. Recognition can be done without temporal information because we assume each training sequence only contains examples of one gesture. There may occasionally exist some non-gesture data samples between two consecutive corpora. This will be handled by the spatial checking. Given a data sample $\vec{x}$, the *Mahalanobis* distances $D(\vec{x}, s_k)$ and $D(\vec{x}, s_{k+1})$ are calculated. $D(\vec{x}, s_k)$ is the distance between $\vec{x}$ and the current state $s_k$. $D(\vec{x}, s_{k+1})$ is the distance between $\vec{x}$ and the next state $s_{k+1}$. A state transition happens if one of the following conditions is met: (1) if $D(\vec{x}, s_{k+1}) \le D(\vec{x}, s_k)$ and $D(\vec{x}, s_{k+1}) \le d_{k+1}$; (2) $D(\vec{x}, s_{k+1}) \le d_{k+1}$ and $D(\vec{x}, s_k) \ge d_k$. So the transition between states is equal to classifying the data samples along the temporal axis.

A gesture is recognized when the last state of the FSM is activated. Its corresponding data segment begins with the first data sample accepted by the FSM and ends with the last data sample accepted by the FSM. Every time a sub-sequence is recognized as a complete gesture, it is collected. The sub-sequence is also segmented into even small units according to the state sequence during the recognition phrase. The collected sub-sequence is then used to update the spatial parameters of the states. The $d_i$ parameters of the states are set as $3\sigma_0$ again. Then the recognition and retraining procedure repeats until the change of the spatial parameter is very small.

If the labels of the begin and end states are different, the constructed FSM covers the whole gesture. If the end state and the beginning state have the same label, the temporary FSM only contains states $s_2, \ldots, s_{n-1}$. In other words, the above iterative procedure only trains the center part of the FSM (states $s_2, \ldots, s_{n-1}$). The sub-sequence set, called $\Omega$, corresponding to $s_2, \ldots, s_{n-1}$ in the training sequence is located. The spatial parameters of states $s_1$ and $s_n$ can be calculated by gradually growing the sub-sequences in $\Omega$. The growing procedure collects the data samples belonging to state $s_1$ and $s_n$ as below. Take $s_1$ as an example:

(1) Set $n = 0$, the sample set of $s_1$: $X^{(n)} = \{\}$, and $\Omega^{(n)} = \Omega$.
(2) The data samples right before those sub-sequences in $\Omega^{(n)}$ form a sample set $Y$. Stop if $Y = \{\}$.
(3) Use $X^{(n)} \cup Y$ to calculate the mean $\vec{\mu}$ and covariance matrix $\Sigma$. Set $d$ as three times of the standard derivation of the distances of the samples from the center of $s_1$.
(4) If $d < \sigma_0$, (a) set $\Omega^{(n+1)} = \Omega^{(n)} + Y$ by attaching the signals in $Y$ to the sub-sequences in $\Omega^{(n)}$ according to their physical positions. (b) set $X^{(n+1)} = X^{(n)} \cup Y$. Go to (2).
(5) Otherwise, set the spatial parameters of $s_1$ as $\vec{\mu}$, $\Sigma$, and $d$.

This algorithm is applied to all the states. While in step (2), the data samples to be collected are those right behind the sub-sequences.

After estimating the spatial parameters, the training data sequence is segmented into pieces such that each of them is an example of the gesture. Those pieces are temporally aligned according to their state sequence. We can calculate the number of samples belonging to each state for each gesture corpus. The temporal parameters of a state $s_i$ can be simply set as: (1) $T_i^{\min}$ = the smallest number of the samples belongs to $s_i$ among all the corpora of $s_i$; and (2) $T_i^{\max}$ = the smallest number of the samples belongs to $s_i$ among all the corpora of $s_i$.

## 3. Speeding up the recognition procedure

Using an FSM as the representation, gesture recognition can be thought as string matching between a data sequence and the state sequence of an FSM. This inspires us to use the *Knuth-Morri-Pratt* (KMP) algorithm [7], a fast string-matching algorithm, to speed up the recognition procedure. It runs in linear time by using a *prefix function*. The prefix function encapsulates the information about how a pattern matches against shifts of itself. This information can be used to avoid testing useless match trials. In the rest of this paper, we assume readers are familiar with KMP algorithm.

### 3.1. Compute Prefix-Function

Given a pattern string $P[1\ldots m] = (p_1, p_2, \ldots, p_m)$, the prefix function for the pattern $P$ is the function $\pi$: $\{1, 2, \ldots, m\} \to \{0, 1, \ldots, m\text{-}1\}$ such that $\pi(q) = \max\{k : k < q \text{ and } P_k \supset P_q\}$, where $P_k = P[1\ldots k]$, $P_q = P[1\ldots q]$, and $P_k \supset P_q$ denotes a string $P_k$ is a prefix of a string $P_q$. See [7] for the details of how to calculate the *prefix function*.

The prefix function calculation is modified for the FSM to fit into our task by changing the similarity measurement. Assume we have two states in an FSM, $s_i$ and $s_k$. The state $s_i$ is listed in front of $s_k$ in the FSM.

Instead of using exact matching, we use inexact matching. A state $s_i$ is said to match with $s_k$ if $D(\vec{\mu}_{s_i}, \vec{\mu}_{s_k}) < d_{s_k}$ and $D(\vec{\mu}_{s_k}, \vec{\mu}_{s_i}) < d_{s_i}$.

### 3.2. FSM-KMP Gesture Recognizer

Using the new *prefix function*, we incorporate the KMP algorithm into the FSM gesture recognition procedure, called FSM-KMP recognizer. During the online gesture recognition, a global buffer *B* is used to store the data sequence currently being recognized. The first element of the global buffer is the first sample accepted by one of the FSM gesture recognizers. The last of the element of *B* is the data sample observed at the current time point. Each state of the FSM maintains a pointer pointing to the position of the global buffer where it begins to accept the data. The global buffer is initialized as empty. The pointers of the states are set to the beginning of the buffer. When a gesture is recognized as completed or it is decided that no gesture could exist in *B*, the buffer are cleaned up and the pointers are reset.

Different from the original KMP, the FSM-KMP algorithm tries to match an FSM with a data sequence. The state sequence of the FSM is equal to the pattern in the KMP algorithm. We replace the pattern shift step in KMP with a state shift. Similar to that in KMP, the shift is performed based on the prefix function. At the same time, the buffer pointer of the state is updated accordingly so that the states know where to look for the data.

The complexity of the FSM-KMP algorithm is $O(KN)$, where $N$ is the length of the data sequence to be recognized and $K$ is the number of FSMs. This is much lower than DTW, HMM and other state approaches.

## 4. Experiments

We tested the proposed approach on two kinds of data. In the first case, the data is the 2D hand position in the image captured by a skin color tracker. The hand gestures include waving left hand, drawing a circle, and drawing a figure eight. In the second case, mouse trajectories were sampled when the user used a mouse to draw some patterns continuously. The results are shown in Tables 1 and 2. For hand gestures, recognition rates are 90% or better. For mouse gestures, rates are as low as 70% for complex gestures and 90-100% for simpler gestures.

## 5. Conclusions

We have developed an efficient approach for gesture modeling and recognition. The gesture is modeled as an FSM. In contrast to much work on gesture modeling, our method allows a semi-automatic way for constructing the gesture models. One of the advantages of an FSM approach is that it does not need large data sets in order to train a good model. The algorithm first learns spatial information by decoupling the spatial and temporal information of the data. The result provides support for data segmentation and alignment. The temporal information is then learned from the aligned data segments. The KMP algorithm is incorporated into the FSM recognition procedure to achieve fast recognition speed. The computation efficiency will potentially allow this approach to be applied to a very large vocabulary. The proposed approach has been successfully tested on a set of hand gestures captured from live video and mouse gestures.

| Gesture | Wave left hand | Draw a circle | Draw '8' |
|---|---|---|---|
| Size of training set | 30 | 26 | 24 |
| Size of testing set | 10 | 9 | 10 |
| Number of states in FSM | 3 | 9 | 13 |
| Recognition rate on training set | 100% | 96.1% | 100% |
| Recognition rate on test set | 100% | 100% | 90% |

**Table 1**. Results of hand gestures

| Gesture | Z | ∇ | & | ζ |
|---|---|---|---|---|
| Size of training set | 30 | 30 | 30 | 30 |
| Size of testing set | 10 | 10 | 10 | 10 |
| Number of states in FSM | 7 | 6 | 15 | 6 |
| Recognition rate on training set | 100% | 100% | 86.7% | 93.3% |
| Recognition rate on test set | 90% | 100% | 70% | 90% |

**Table 2**. Results of mouse gestures

### References


[1] 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.

[2] 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.

[3] 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.

[4] James Davis and Mubarak Shah. "Visual gesture recognition," *Vision, Image and Signal Processing*, 141(2), 1994, pp. 101-106.

[5] Stephen J. McKenna and Shaogang Gong. "Gesture recognition for visually mediated interaction using probabilistic event trajectories," *Ninth British Machine Vision Conference*, Sep. 1999.

[6] P. Hong, M. Turk, and T.S. Huang, "Gesture modeling and recognition using finite state machines," *Proc. Fourth IEEE International Conference and Gesture Recognition*, March 2000, Grenoble, France.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*, The MIT Press and McGraw-Hill, 1990.

[8] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-286, 1989.