PLANAR SHAPE CLASSIFICATION USING
HIDDEN MARKOV MODEL

Yang He and Amlan Kundu
Department of Electrical and Computer Engineering
State University of New York at Buffalo
Buffalo, NY 14260, U.S.A.

ABSTRACT

In this paper, we present a planar shape recognition approach based on hidden Markov model and autoregressive parameters. This approach segments closed shapes into segments and explores the characteristic relations between consecutive segments to make classification at finer level. The algorithm can tolerate a lot of shape contour perturbation and moderate amount of occlusion. Also, the overall classification scheme is independent of shape orientation. Excellent recognition result has been reported. A distinct advantage of the approach is that the classifier does not have to be trained all over again when a new class of shapes is added.

I. INTRODUCTION

Two dimensional shape classification is an important problem in computer vision and image processing. There currently exist many shape representation techniques. Some of the early techniques are summarized in [2]. They include Fourier descriptors [3], the class of space domain or curve fitting methods featuring the chain code approach [4,5], the polygonal approximation scheme [6], the category of decomposition technique including the medial axis transform (MAT) [7], decomposition at concave vertices, decomposition by clustering, and the scale domain representation [8]. All of these methods have their advantages and limitations.

Smith and Jain [9] and You and Jain [10] have developed a series of chord length distribution methods which are data independent. Some performance improvement over the Fourier descriptor method has been shown. They have also considered occluded shapes.

Another very effective technique is the Autoregressive (AR) model approach [11–12]. It has been further extended to bivariate case with better results [13]. In [12] and [13], 100% correct classification results are reported. The major disadvantage is that the schemes are very sensitive to shape occlusion, even slight occlusion. In this case, only 35% to 70% recognition rates were reported. The schemes are also sensitive to shape contour distortion or perturbation.

The reason for this is that these schemes model the whole shape with only one set of predictive parameters. If the shape contains a large number of sample points, and the contour varies radically, the shape may seem rather unpredictable. In this case, it is reasonable to think that an AR model with finite number of parameters is not adequate for the whole shape. This problem inspires the consideration of another important technique in pattern recognition – the hidden Markov model.

Hidden Markov Model (HMM) explores the relationship between consecutive segments of a pattern to be classified. Each segment is relatively smaller, and therefore is easier to be characterized. HMM has been used successfully in many applications. For example, Rabiner et al have applied HMM to speech recognition [14–16]. Kundu et al have developed a HMM based handwritten word recognition scheme [17]. In these applications, better results are achieved compared to other methods that treat the patterns as a single set of features.

In this paper, we will present a technique combining HMM and AR model to recognize closed 2-D shapes. Our approach is to segment the 1-D representation sequence of a closed shape into several pieces, characterize each piece with AR parameters and get a vector sequence for each shape, and then apply HMM to classify the vector sequence.

It should be noted that our goal is to recognize shapes with contour perturbation, i.e., the shapes of the same class may not come from different silhouettes of the same 2-D object, and may not be matched by just changing the size and orientation. We will also
consider moderately occluded shapes, which will be discussed with our experiments. In contrast, in [12] and [13], all shapes of the same class, except for those in the occluded data set, are taken from different silhouettes of a single cut paper with only changes in size and orientation. Obviously, our goal is much more difficult to achieve than that of [12] and [13].

II. SHAPE REPRESENTATION

First, the 2-D shape is transformed into 1-D signal sequence. The feature vector is then extracted from the 1-D sequence.

1-D Representation — We use the radii spaced at equal curve length on the contour as the 1-D representation. We calculate the center of gravity of the shape and the length of shape contour, divide the length by an integer K, and then form the 1-D sequence by taking the radii from the center of gravity to each dividing point on the contour. This method can represent concave shapes without any modification. After the 1-D sequence is obtained, we divide it into T segments of equal length L such that K = TL. Each segment is denoted by r(l), 1 = 1, 2, ..., L.

AR Parameters — Each segment sequence is modeled as an AR process:

\[ r(l) = \alpha + \sum_{j=1}^{M} \theta_j r(l-j) + \sqrt{\beta} w_l \quad l = 1, \ldots, L \]

where \( \theta_1, \ldots, \theta_M, \alpha, \beta \) are the model parameters and M is the order of the model. To get M initial values of \( r(l-j) \) for \( l-j < 1 \), we let each segment overlap with the previous segment by M points. Our feature vector is chosen as \( [m_0, \theta_1, \ldots, \theta_M, \alpha/\sqrt{\beta}] \), where \( m_0 \) is the normalized average radius value of the segment, called the segment mean. The feature vector is invariant to shape size and translation.

III. ORIENTATION AND STARTING POINT

Since the planar shape is cut into segments and the feature vectors are calculated from each segment, by no means can we make the feature vectors invariant to shape orientation. We have to rotate all the shapes of each class to the same orientation before we form the 1-D sequence.

Elongation Axis is defined as the line that passes through the center of gravity of the shape and about which the second moment of the shape is minimum. The angle between elongation axis and x axis, \( \theta \), where \(-\pi/2 \leq \theta \leq \pi/2\), is defined as elongation angle. The ratio of minimum and maximum second moments, \( E_{\text{min}} \) and \( E_{\text{max}} \), is defined as elongation ratio. If the ratio is smaller than a threshold \( T_e \), the shape is well elongated, and will be rotated by the elongation angle. If the ratio is larger than \( T_e \), the shape is not well elongated, and the elongation method does not work well. In this case we use the method described below.

Minimum Radius Point is the point on the shape contour that has the minimum distance to the center of gravity. If the value of elongation ratio is larger than \( T_e \), we simply rotate the shapes so that this point is on the positive y axis.

Two Way Ambiguity is the 180 degree difference in orientation that may remain after above rotation. To remove two way ambiguity along x axis, we cut the shape along y axis into two parts and calculate the energy of each part. If the energy of the positive y part is smaller than that of the negative y part, we rotate the shape by 180 degree. Similar procedure is used to remove the two way ambiguity along y axis.

IV. CONTINUOUS DENSITY HMM

A hidden Markov model (HMM) is a Markov chain whose states cannot be observed directly, but can only be observed through a sequence of observation vectors. Given a model and its parameters, each observed vector could have been generated from any state with an associated density function. A continuous density HMM is characterized by the initial state probabilities, \( \pi_i \), \( i = 1, \ldots, N \), the state transition probabilities, \( a_{ij} \), \( i, j = 1, \ldots, N \), and the observation densities, \( b_j(o_t) \), \( j = 1, \ldots, N ; t = 1, \ldots, T \). Each observation \( o_t \) is a feature vector. We assume that the density of the feature vector is a multivariate Gaussian distribution. Two problems associated with HMM are of our concern:

Problem 1: Classification

To solve our shape classification problem, we create one HMM for each class. For a classifier of P classes, we denote the P models by \( \lambda_p \), \( p = 1, 2, \ldots, P \). When a shape \( O \) of unknown class is given, we calculate

\[ p^* = \arg \max_p p(O, Q^*|\lambda_p) \]  

and classify the shape as class \( p^* \), where \( Q^* \) is the optimum state sequence, and \( p(O, Q^*|\lambda_p) \) is called state-optimized likelihood function. For a given \( \lambda \), an efficient method to find \( p(O, Q^*|\lambda) \) is the well known Viterbi algorithm.

Problem 2: Training

Since our decision rule is based on the state-optimized likelihood function, the estimated parameter \( \lambda \) should be such that \( p(O, Q^*|\lambda) \) is maximized over the training set. It is shown in [16] that the segmental K-means algorithm converges to the maximized state-optimized likelihood function for the Gaussian density we have assumed. The segmental K-means algorithm for our training problem is summarized below.
1. Cluster all training vectors into \( N \) clusters using minimum distance rule with random initial clustering centroids. Each cluster is chosen as a state and numbered from 1 to \( N \). The \( t \)-th vector, \( o_t \), of a training sequence \( O \) is assigned to state \( i \), denoted as \( o_t \in i \), if its distance to state \( i \) is smaller than its distance to any other state \( j, j \neq i \). This is the initialization step of the complete procedure.

2. Calculate the mean vector \( \mu_i \) and covariance matrix \( V_i \) for each state \( i \):

\[
\mu_i = \frac{1}{N_i} \sum_{o_t \in i} o_t, \quad (3)
\]

\[
V_i = \frac{1}{N_i} \sum_{o_t \in i} (o_t - \mu_i)(o_t - \mu_i)^T. \quad (4)
\]

where \( N_i \) is the number of vectors assigned to state \( i \).

3. Calculate transition probabilities and initial probabilities from the training sequences:

\[
\pi_i = \frac{\text{No. of occurrence of } \{o_t \in i\}}{\text{No. of training sequences}}, \quad (5)
\]

\[
a_{ij} = \frac{\text{No. of occurrence of } \{o_t \in i \text{ and } o_{t+1} \in j\}}{\text{No. of occurrence of } \{o_t \in i\} \text{ for all } t}, \quad (6)
\]

4. Calculate density functions of each training vector for each state \( i \):

\[
b_i(o_t) = \frac{1}{(2\pi)^{d/2}|V_i|^{1/2}} \exp\left[-\frac{1}{2}(o_t - \mu_i)(o_t - \mu_i)^T V_i^{-1}\right]. \quad (7)
\]

5. Use Viterbi algorithm to trace the optimal state sequence \( Q^* \) for each training sequence. A vector is reassigned a state if its original state assignment is different from the tracing result, i.e., assign \( o_t \in i \) if \( q_t^* \neq i \).

6. If any vector is reassigned a new state in Step 6, use the new state assignment and repeat Step 2 through Step 5, otherwise stop.

V. NON-STATIONARY MODELS

The HMM described in the previous section is the first order approximation of the observed random process. In the first order model, it is assumed that the occurrence of each state at time \( t \) depends only on the state the process takes at time \( t - 1 \). A practical process, however, may not be that simple. The occurrence of each state at time \( t \) may depend not only on the state at time \( t - 1 \), but also on the states at times prior to \( t - 1 \). In this case, to further improve the performance, a higher order HMM should be built. But the higher order model requires more training samples and increases the computational complexity.

An alternative method to improve HMM performance in this respect is to consider the state transition as non-stationary, i.e., the state transition probability \( a_{ij} \) depends on \( t \). To estimate the transition probabilities, we simply replace Eqn. (6) in Section IV by

\[
\hat{a}_{ij}(t) = \frac{n_{ij}(t+1)}{n_i(t)} \quad (8)
\]

for \( 1 \leq i \leq N, 1 \leq j \leq N \), and \( 1 \leq t \leq T - 1 \), where \( n_{ij}(t) \) is the number of transitions from state \( i \) at time \( t \) to state \( j \) at time \( t + 1 \), and \( n_i(t) \) is the number of times the state \( i \) is visited at times \( t \). It is seen from Eqns. (6) and (8) that only the storage for the transition probabilities is increased, but not the computation. The same contention is true for the classification procedure. In this case we simply change \( a_{ij} \) in the Viterbi algorithm to \( \hat{a}_{ij}(t) \). The computation does not increase at all.

VI. EXPERIMENTAL RESULTS AND CONCLUSIONS

We have used 8 classes of shapes to test our algorithm. 8 typical shapes, one from each class, are shown in Fig. 1. We created 30 samples for each class, of which 20 are used as training sample and 10 as test sample. The shapes are drawn by hand on a SUN workstation. The only restriction on generating the shapes is that it should not be difficult for human beings to classify the shapes. A lot of contour perturbation is allowed as long as this restriction is satisfied. Of the 30 samples of each class, only 24 are individually drawn. The rest 6 are occluded shapes. They are obtained by cutting off a portion of one of the generated shapes with a straight line. Three occluded shapes are put in the training set and the other three in the test sample set. In this way, a moderate amount of occlusion is accommodated in our experiments. The restriction on occlusion is that all the major portions, or branches, should remain, so that no major geometric property is changed. Some examples of occluded shapes are shown in Fig. 2. The left most shape of each class is the original one. The rest 3 shapes in the same row are occluded from the left most one. It can be seen that occlusion may be considered as a special case of perturbation.

All shapes are rotated by a random angle in between 0 and \( 2\pi \). Then the orientation algorithm described in Section IV is used to orientate the shapes. Before we determine the value for threshold \( T_r \), both the elongation axis and minimum radius point methods are tried to orientate all the shapes. It is found out that both methods work well and give consistent orientation for class 1, 3, 4, and 5. For class 2 and 6, elongation axis method gives good orientation, but minimum radii point method doesn't. For class 7 and 8, the results are the opposite of class 2 and 6. The
values of elongation ratio $R_E$ are all smaller than 0.55 for class 1, 2, 3, and 6 and are all greater than 0.55 for class 5, 7, and 8. For class 4, four shapes have an $R_E$ value larger than 0.55, and all others smaller than 0.55. Therefore, it turns out that 0.55 is a good choice for the value of $T_E$ for our data set. After this threshold is set, the complete orientation algorithm is applied. For class 1, 2, 3, 4, and 6, the shapes in each class are orientated exactly as we would do by inspection. For the rest of the classes, more than 80% of the shapes are well orientated. The rest 20% of the shapes are somewhat biased after orientation, but the bias angles are within the range of $-20$ to $+20$ degrees. We consider this result quite acceptable for the training procedure.

After the initial points are determined, a boundary tracing algorithm is applied to calculate the boundary curve length. The length is divided by $K = 256$ and 256 radii are taken at equal length on the boundary to form the 1-D sequence as described in Section II. Each sequence is divided into $T = 16$ segments. Each segment has $M$ points overlapped with previous segments, where $M$ is the AR model order number. The $M + 1$ parameters are calculated as described in Section II. The segment mean is used as the $(M+2)$-th feature. The mean is scaled to the range 0–2. This range is comparable to the ranges of other parameters.

In the training process, we take the average of all training sequences of each class to get an average sequence for each class. Then we successively choose every possible $N$ vector combination as a set of initial cluster centers to get a model. Most initial clustering centers gave the same or very close results. Only a few initial clustering centers gave results with significant difference. In the latter case, the average values of $P^*$ are all very small. We simply ignore these results and choose the one with the largest average $P^*$ as our model.

We have tried different number of states, from $N = 2$ to $N = 6$, and both stationary and non-stationary transition for our HMM. We have also compared the results of AR models of different orders, from $M = 2$ to $M = 6$. Table 1 shows the number of errors occurred in classifying the total 80 test shapes. It is seen that 100% correct classification is achieved in five cases. The five cases are: $N = 3$ and $M = 4$, $N = 3$ and $M = 5$, $N = 4$ and $M = 3$, $N = 4$ and $M = 4$, and $N = 4$ and $M = 5$, all with non-stationary transition.

It is seen from Table 1 that the HMM of a certain number of states gives better results. Three of the five 100% cases are obtained with $N = 4$. For $N = 2$ to $N = 4$, the correct recognition rate tends to increase with the increase of $N$. This is expected, since more states can distinguish more different characteristics of the shape segments. However, for $N = 5$ and $N = 6$, the results are the opposite. It is due to the fact that the models with more states require more training samples for an accurate estimation of the model parameters. Our results show that $N = 4$ is quite adequate.

The recognition rate is also affected by the order of AR model. For $M = 2$ to $M = 4$, the correct recognition rate tends to be higher for the higher order model. But for $M = 4$ to $M = 6$, the increase of model order does not increase the recognition rate in most cases. This is a strong indication that the shape segments are well represented by an AR model of order no higher than 4. Collectively, the fourth and fifth order AR models give the best result, each with two cases of 100% correct recognition, as seen from Table 1.

It is also seen from Table 1 that all the five 100% recognition results are obtained with non-stationary transition. This result substantiates the assumption that the underlying Markov chain is more likely to be non-stationary.

It is interesting to compare our data and results to those of [12] and [13]. Since all our samples are drawn by hand and some are occluded, but not generated from some "parent" shapes, as mentioned earlier in Section I, our data are random. In contrast, the data in [12] and [13] may be considered as less random. Therefore, although their work and ours have all achieved 100% recognition rate, our scheme has more tolerance to contour perturbation and moderate amount of occlusion. A final comment on this shape recognition scheme is that its training procedure is independent for each class. That means if more classes are added to the classifier, we need only to train for the new classes.

**REFERENCE**


Class 1 Class 2 Class 3 Class 4
Class 5 Class 6 Class 7 Class 8

Figure 1. 8 classes of shapes
Figure 2. Occluded shapes

Table 1. Recognition result

<table>
<thead>
<tr>
<th>No. of HMM States</th>
<th>Type of HMM</th>
<th>Order of AR Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Stationary</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>Non-Stationary</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Stationary</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Non-Stationary</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Stationary</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Non-Stationary</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Stationary</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Non-Stationary</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Stationary</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Non-Stationary</td>
<td>6</td>
</tr>
</tbody>
</table>