2–D Shape Classification Using Hidden Markov Model

Yang He and Amlan Kundu

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

Index Terms— **Autoregressive model, hidden Markov model, nonstationary transition, pattern recognition, segmental K-means algorithm, shape classification, shape occlusion, shape orientation, stationarity test.**

I. Introduction

SHAPE classification is a basic problem in computer vision [1]. Its application has been found in various areas such as industrial part identification, target identification, character recognition, and medical diagnosis. Recently, more emphasis is put on 3–D shape classification. In many cases, the 3–D shape classification problem can be decomposed into 2–D shape classification problems. The performance of shape classification largely depends on the shape representation technique.

There currently exist many shape representation techniques. Some of the earlier techniques are summarized by Pavlidis in [2]. The Fourier descriptors proposed by Zahn and Roskies [3] and Granlund [4] have the advantage of simplicity, but they have difficulties in describing local information and discriminating symmetrical shapes [2]. The class of space domain or curve-fitting methods features the chain code approach used by Freeman [5] and McKee and Aggarwal [6] and the polygonal approximation scheme suggested by Pavlidis [7]. The two methods give quite similar results. In the polygonal case, higher order approximation is often desired, but it is also computationally expensive [2]. Another category is the decomposition technique including medial axis transform (MAT) [8], decomposition at concave vertices [9], and decomposition by clustering [10]. These methods posses many desired features, but their implementations are very complicated. A recently developed technique is the scale domain representation used by Mokhtarian and Mackworth [11]. It is suitable for shoreline contour description but has difficulty representing curves with long straight segments or convex curves. The chord length distribution method introduced by Smith and Jain [12] and You and Jain [13] are quite effective in recognizing shapes that are represented by a relatively small number of boundary points. The chord length distribution method is also useful for problems where sufficient number of sample shapes are not available for training. In general, this technique has much better performance than the Fourier descriptor method.

Another very effective technique is the autoregressive (AR) model approach. The AR model for closed-shape analysis was first proposed by Kashyap and Chellappa in 1981 [14]. The use of AR parameters as a feature vector can characterize shapes in a very systematic manner. The experiments conducted by Dubois and Glanz showed some good results [15]. Das *et al.* extended the AR model to the bivariate case and further improved the classification results [16]. In both papers, a 100% correct classification result was reported. The major disadvantage is that these schemes are very sensitive to shape occlusion or even slight occlusion. In this case, only 35 to 70% recognition rates were reported. It can be derived that these schemes are also sensitive to shape contour distortion or perturbation.

The reason for the drawback of the AR model is that it models 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.

The hidden Markov model (HMM) does not model the whole pattern or shape as a single feature vector. It 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 in speech recognition [17]–[20]. Kundu *et al* have developed a handwritten english script recognition scheme using HMM [21]. In these applications, better results are achieved, compared with other methods that treat the patterns as a single set of features.

In this paper, we will present a technique combining an 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

Manuscript received October 16, 1990; revised May 5, 1991. This work was supported by National Science Foundation grant MIP-8908082.

The authors are with the Department of Electrical and Computer Engineering, State University of New York at Buffalo, Amherst, NY 14260.

IEEE Log Number 9102086.

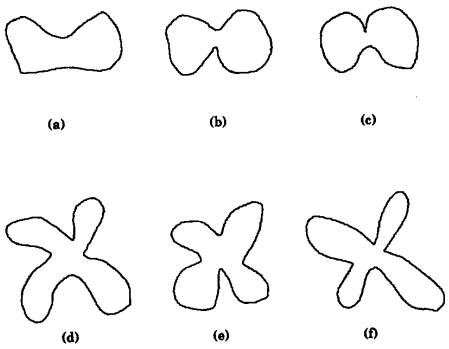


Fig. 1. Six sample shapes for two classes: (a), (b), and (c) form one class; (d), (e), and (f) form another class.

AR parameters and get a vector sequence for each shape, and then apply HMM to classify the vector sequence.

It should be noted, before we describe our work, 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. For example, Fig. 1 shows six sample shapes of two classes. Fig. 1(a), (b), and (c) are of one class, and Fig. 1(d), (e), and (f) are of another. A lot of contour perturbation is seen here, and no matching is possible. We will also consider moderately occluded shapes, which will be discussed with our experiments. In contrast, in [15] and [16], 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 [15] and [16].

Compared with the methods of the same category, say, the AR model method, ours is more general. Our AR parameters characterize only the local property, whereas the relationship among the local segments is included in another stochastic model, i.e., HMM. In contrast, the AR methods characterize the whole shape with only one set of parameters. In addition, the present scheme is more general because it offers more degrees of freedom to deal with more complex shapes, i.e., to deal with more complex shapes, one can keep the complexity of feature extraction constant and increase only the complexity of the HMM.

The remaining sections of this paper are organized as follows: Section II describes shape representation with a sequence of AR parameters. Section III presents the theory of HMM as applied to the shape classification problem. Section IV discusses shape orientation and the starting point determination in tracing the shape boundary. Section V discusses some practical considerations in implementation. Section VI gives the detailed experimental results. Section VII summarizes the conclusions.

II. Shape Representation

Representation of a 2–D shape with one feature vector or a sequence of feature vectors includes two aspects of data manipulation. First, the 2–D shape must be transformed into a 1–D signal sequence. Then, the feature vector is extracted from the 1–D sequence.

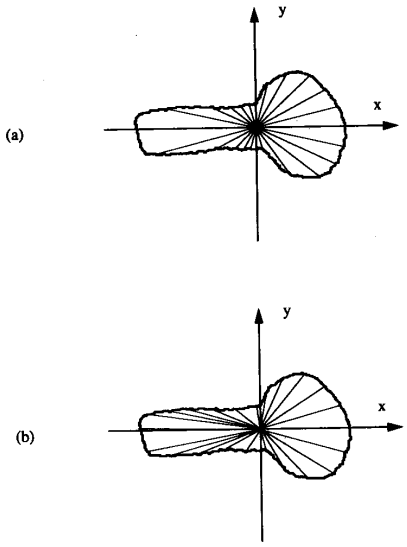


Fig. 2. Two methods of forming a 1–D radii sequence: (a) Equal angle method; (b) equal curve length method.

1–D Representation

The most commonly used method to represent a 2–D shape with a 1–D signal is to use a sequence of radii from the center of gravity to the contour of the shape. We denote this radii sequence by $r(k), k = 1, 2, \ldots, K$, where K is the length of the radii sequence. There are two ways to form the 1–D radii sequence:

1) Choose the radii spaced at an equal angle $\theta = 2\pi/K$.
2) Choose the radii spaced at an equal curve length on the contour.

One example for each method is shown in Fig 2. Fig. 2(a) shows the equal angle method, and Fig. 2(b) shows the equal curve length method. The equal angle method has the advantage that $r(k)$ can be considered to be a circular sequence, i.e., $r(k + nK) = r(k)$ for any integer k and n. Rotating the shape by a $m\theta$ angle is equivalent to shifting the sequence circularly by m samples. This will result in a rotation invariant property of the AR parameters. However, this method has problem representing concave shapes, especially when the center of gravity of the shape lies outside of the closed contour. Modifications are needed in this case, which makes the algorithm more complicated [15].

The equal curve length method can represent concave shapes, even with the center of gravity lying outside of the contour, without any complication. However, it is not, in general, rotation-shift equivalent. Since we will segment the whole shape into several segments and calculate AR parameters from each segment, the rotation-invariant property is not important to us. Therefore, we will use the equal curve length method to simplify 1–D representation. It is noted that this representation is translation invariant.

After the 1–D sequence is obtained, we divide it into T segments of equal length L. That means we should choose $K = TL$. Then, we form each segment sequence $r_t(l)$, $t = 1, 2, \ldots, T$ by the segment itself, preceeded by M points overlapped with the previous segment, i.e.

$$r_t(l) = r[(t-1)L + l], \quad l = -(M-1), -(M-2), \ldots, L. \tag{2.1}$$

For the first segment sequence, we treat $r(k)$ as a periodic sequence of period K, i.e., the first segment sequence r_1 is overlapped with the last M points of the last segment r_T. The reason for overlapping and the value M takes will be explained later in this section.

Autoregressive Model

In describing the AR model, we will consider each segment sequence separately. To simplify the notation, we will omit the subscript t, i.e., we will use $r(l)$. Each radius in the sequence has some correlation with previous radii and can therefore be predicted through a number of, say, M observations of previous radii. The autoregressive model is a simple prediction of the current radius by a linear combination of M previous radii plus a constant term and an error term:

$$r(l) = \alpha + \sum_{j=1}^{M} \theta_j r(l-j) + \sqrt{\beta} w_l \quad l = 1, \ldots, L, \tag{2.2}$$

where $r(l)$ is the current radius value, $r(l-j)$ is the previous radius values, θ_j is the autoregressive coefficients to be estimated, M is the model order, α is a constant to be estimated, $\sqrt{\beta}$ is the variance of prediction noise (and reflects the accuracy of the prediction), and w_l is a random number with zero-mean and unit variance. $\theta_1, \ldots, \theta_M, \alpha, \beta$ are called the model parameters.

It is noted from (2.2) that to predict $r(1)$, we need M initial values of $r(l)$, i.e., $r(-M+1), r(-M+2), \ldots, r(0)$. This is the reason for overlapping M points with the previous segment when forming each segment sequence. It is obvious that the number of overlapped points should at least be equal to the order number. It is easy to derive that

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_M \\ \alpha \end{bmatrix} = \begin{bmatrix} R_{11} & \ldots & R_{1M} & S_1 \\ \vdots & \ddots & \vdots & \vdots \\ R_{M1} & \ldots & R_{MM} & S_M \\ S_1 & \ldots & S_M & N \end{bmatrix}^{-1} \begin{bmatrix} R_{01} \\ \vdots \\ R_{0M} \\ S_0 \end{bmatrix} \tag{2.3}$$

where

$$R_{ij} = R_{ji} = \sum_{l=1}^{L} r(l-i) r(l-j), \qquad i, j = 1, \ldots, M \tag{2.4}$$

$$S_i = \sum_{l=1}^{L} r(l-i), \qquad i = 0, 1, \ldots, M \tag{2.5}$$

and

$$\beta = \frac{1}{L} \sum_{l=1}^{L} [r(l) - (\alpha + \sum_{j=1}^{M} \theta_j r(l-j))]^2. \tag{2.6}$$

It should be noted that the choice of optimal model order M is a very complicated problem and is usually decided by experiment.

Segment Mean

We define the segment mean m_r by

$$m_r = \frac{1}{L} \sum_{l=1}^{L} r(l). \tag{2.7}$$

This parameter carries a lot of information. For example, if we segment a circle into any number of segments, all segment means are equal to one another, but if we segment an ellipse into eight segments, m_r will change with a period of 4 as t changes from 1 to 8. In this case, the segment mean itself can discriminate the two shapes. Therefore, we add m_r as an additional feature.

Translation and Scale Invariance

In practical application, the shapes to be recognized may have different location, size, and orientation. Therefore, it is required that the feature vector be translation, scale, and orientation invariant. We leave the discussion on orientation problem to Section IV and discuss only the translation and scale invariance here.

Since the radii are translation invariant, all parameters derived from radius sequence are also translation invariant. It is easy to show that $\theta_1, \theta_2, \ldots, \theta_M$ and α are also scale invariant [14]. Although β is not scale invariant, it can also be shown that $\alpha/\sqrt{\beta}$ is scale invariant [15]. Thus, instead of using α and β as two independent features, we use $\alpha/\sqrt{\beta}$ as a single feature. It is both translation and scale invariant.

To make m_r scale invariant as well, we scale it, for each shape sample, into the range $[0, 1]$. To do so, we subtract the minimum value of m_r from each m_r and divide it by the difference of maximum and minimum values of m_r. During the training procedure, we will decide a proper weight for this component such that the role of this feature and other features are balanced. This will be discussed again in Sections V and VI. In the sequel, we will always refer to the scaled and weighted segment mean when we mention this component, but we will use the same notation m_r. Now, the final feature vector is $X = [\theta_1, \ldots, \theta_M, \alpha/\sqrt{\beta}, m_r]^t$, with dimension of M_2, where $M_2 = M + 2$.

III. Continuous Density HMM

A first-order N-state Markov chain is defined by an $N \times N$ state transition probability matrix A and an $N \times 1$ initial probability vector Π, where

$A = \{a_{ij}\}$, $a_{ij} = \Pr(q_{t+1} = j | q_t = i)$, $i, j = 1, 2, \ldots, N$,
$\Pi = \{\pi_i\}$, $\pi_i = \Pr(q_1 = i)$, $i = 1, 2, \ldots, N$,
$Q = \{q_t\}$ — state sequence. $q_t \in \{1, 2, \ldots, N\}$, $t = 1, 2, \ldots, T$,
N — number of states,
T — length of state sequence.

By definition, $\sum_{j=1}^{N} a_{ij} = 1$ for $i = 1, 2, \ldots, N$ and $\sum_{i=1}^{N} \pi_i = 1$. A state sequence Q is a realization of the

Markov chain with probability

$$\Pr(Q|A,\Pi) = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1}q_t}. \tag{3.1}$$

A hidden Markov model (HMM) is a Markov chain whose states cannot be observed directly but can be observed through a sequence of observation vectors. Each observation vector, which is also called a symbol, manifests itself as states through certain probability distributions. In other words, each observed vector is generated by an underlying state with an associated probability distribution. To solve our problem, we will consider only the observations with continuous density. A continuous density HMM is characterized by the state transition probability A, the initial state probability Π, and an $N \times 1$ observation density or symbol probability density vector B, where

$B = \{b_j(o_t)\}$, $b_j(o_t)$ = *a posteriori* density of observation o_t given $q_t = j$.
$O = \{o_t\}$ – observation sequence. $t = 1, 2, \ldots, T$

In our shape recognition problem, each observation o_t is a vector of M_2 dimension, where M_2 is the dimension of the feature vector described in Section II.

It is reasonable to assume that the observation density is Gaussian. In this case, the density is completely specified by the mean and covariance of o_t, i.e., $b_j(o_t) = N(\mu_j, V_j)$, where μ_j and V_j are the conditional mean vector and the conditional covariance matrix, respectively, of o_t given state $q_t = j$. It should be noted that the states of an HMM may not have specific physical meaning. They may just reflect some clustering properties of the observation vectors in the feature space.

We can more compactly denote the parameter set by $\lambda = (A, \Pi, B)$. Then, an HMM is completely specified by λ. Three problems associated with HMM are of our concern:

1) Based on what optimization criterion should our model be built?
2) Given the model and an observation sequence $O = \{o_1, o_2, \ldots, o_T\}$, how can we classify the observation efficiently? This is the classification problem.
3) Given a number of observation sequences of a known class, how can we obtain the optimal model estimate $\hat{\lambda}$? This is the training problem.

Problem 1: Optimization Criterion

Suppose we are given a model λ and an observation sequence $O = \{o_1, o_2, \ldots, o_T\}$. Then, the density function of O is given by

$$p(O|\lambda) = \sum_{Q} \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t}(o_t). \tag{3.2}$$

A direct choice of optimization criterion is the maximum likelihood criterion that maximizes (3.2). The estimation of the parameters by this criterion can be solved using the Baum-Welch reestimation algorithm. The algorithm is an iterative procedure that guarantees monotonic increase of the likelihood function for a given set of training samples.

Another optimization criterion is to maximize the state-optimized likelihood function defined by

$$p(O, Q^*|\lambda) = \max_{Q} p(O, Q|\lambda)$$
$$= \max_{Q} \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t}(o_t) \tag{3.3}$$

where $Q^* = \{q_1^*, q_2^*, \ldots, q_T^*\}$ is the optimal state sequence associated with the state-optimized likelihood function. Equation (3.3) is the density of the optimal or the most likely state sequence path among all possible paths. The estimation of the parameters using this criterion is given by the segmental K-means algorithm. This algorithm is an iterative procedure that guarantees, under some conditions described later, the monotonic increase of the state-optimized likelihood function for a given set of training samples.

Comparing (3.2) and (3.3), we find out that (3.2) involves computation along all possible state paths, whereas (3.3) tracks only the most likely path. Therefore, the computation required by (3.3) is much less than that of (3.2). In addition, since $b_{q_t}(o_t)$ often has a large dynamic range, overflow or underflow is more likely to happen in an evaluation of (3.2) than in an evaluation of (3.3). Furthermore, in a particular application, if the data fit the model very well, all the observation samples of one class are likely to have a single or a few underlying state sequences. It means that the optimal state sequence in (3.3) carries a lot of information that may discriminate one class from another. For these reasons, we have chosen maximization of (3.3) as our criterion.

Problem 2: 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 λ_p, $p = 1, 2, \ldots, P$. When a shape O of unknown class is given, we calculate

$$p^* = \arg\max_{p} p(O, Q^*|\lambda_p) \tag{3.4}$$

and classify the shape as class p^*.

Now, we can immediately see one of the advantages of HMM. The model for one class is independent of the model for any other class, i.e., the training for one class is not related to the training for any other class. It follows that when a new class is added to the classifier, we need only to train for this new class but do not have to retrain for any other class.

For a given λ, an efficient method to find $p(O, Q^*|\lambda)$ is the well-known Viterbi algorithm [22] as described below.

Viterbi Algorithm
Step 1. Initialization
For $1 \leq i \leq N$,

$$\delta_1(i) = \pi_i b_i(o_1) \tag{3.5}$$

$$\psi_1(i) = 0. \tag{3.6}$$

Step 2. Recursive computation
For $2 \leq t \leq T$ for $1 \leq j \leq N$

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i)a_{ij}]b_j(o_t) \tag{3.7}$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i)a_{ij}]. \tag{3.8}$$

Step 3. Termination

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{3.9}$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]. \tag{3.10}$$

Step 4. Tracing back the optimal state sequence
For $t = T-1, T-2, \ldots, 1$

$$q_t^* = \psi_{t+1}(q_{t+1}^*). \tag{3.11}$$

P^* is the state-optimized likelihood function, and $Q^* = \{q_1^*, q_2^*, \ldots, q_T^*\}$ is the optimal state sequence.

In practice, as t increases, the value of $\delta_t(j)$ could be very large or very small; therefore, an overflow or underflow may occur during computation on a computer. To avoid this problem, we take the logarithm of all the probabilities and the densities, and replace all multiplications with additions. Obviously, the result of tracing the optimal state sequence is not affected by this modification. If any particular value is zero, we set it to a very small number so that it does not affect the result.

Problem 3: Training

In creating the model for each class, we should guarantee that the parameters we obtain are the optimum for a given set of training samples. Since our decision rule is the state-optimized likelihood function, it requires that the estimated parameter $\hat{\lambda}$ be such that $p(O, Q^*|\hat{\lambda})$ is maximized for the training set. It is shown in [19] that the segmental K-means algorithm [20] converges to the state-optimized likelihood function for a wide range of observation density functions, including the Gaussian density we have assumed. It can be compactly written as follows: Starting from an initial model $\hat{\lambda}^0$, calculate

$$\hat{\lambda}^{k+1} = \arg \max_{\hat{\lambda}^k} \{\max_Q p(O, Q|\hat{\lambda}^k)\} \tag{3.12}$$

iteratively until $\hat{\lambda}^{k+1} = \hat{\lambda}^k$. The superscripts here indicate the iteration number.

A more elaborate procedure of the algorithm, which is suitable for direct computation, is described below.

1. Cluster all training vectors into N clusters using the 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, which is denoted as $o_t \in i$, if its distance to state i is smaller than its distance to any other state j, $j \neq i$. The distance measure we have used is the unweighted Euclidean distance. This step is to get a good initialization for the complete training procedure.

2. Calculate the mean vector and the covariance matrix for each state: For $1 \leq i \leq N$:

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{o_t \in i} o_t, \tag{3.13}$$

$$\hat{V}_i = \frac{1}{N_i} \sum_{o_t \in i} (o_t - \hat{\mu}_i)^t (o_t - \hat{\mu}_i) \tag{3.14}$$

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

3. Calculate the transition probabilities and the initial probabilities: For $1 \leq i \leq N$

$$\hat{\pi}_i = \frac{\text{Number of occurrences of } \{o_1 \in i\}}{\text{Number of training sequences}}. \tag{3.15}$$

For $1 \leq i \leq N$ and $1 \leq j \leq N$

$$\hat{a}_{ij} = \frac{\text{Number of occurrences of } \{o_t \in i \text{ and } o_{t+1} \in j\} \text{ for all } t}{\text{Number of occurrences of } \{o_t \in i\} \text{ for all } t}. \tag{3.16}$$

4. Calculate density functions of each training vector for each state. For $1 \leq j \leq N$

$$\hat{b}_j(o_t) = \frac{1}{(2\pi)^{M_2/2} |\hat{V}_j|^{1/2}} \exp\left[-\frac{1}{2}(o_t - \hat{\mu}_j)\hat{V}_j^{-1}(o_t - \hat{\mu}_j)^t\right]. \tag{3.17}$$

5. Use the Viterbi algorithm and the new probabilities 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 $q_t^* = i$.

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

IV. 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. However, the orientation invariance requirement can be overcome if and only if we rotate all the shapes of each class to the same orientation. This rotation should make the radii sequences of the shapes of each class most similar. It is not an easy job in general. We will use two characteristic quantities (elongation axis and minimum radius point) to determine the starting point.

Elongation Axis

Definition: The elongation axis of a shape 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 the elongation axis and x axis θ, where $-\pi/2 \leq \theta \leq \pi/2$ is defined as the elongation angle. Consider the shape given by Fig. 3. The center of gravity of the shape

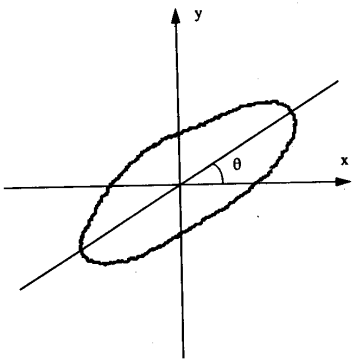


Fig. 3. Elongation axis and elongation angle.

is at the origin of the (x, y) coordinates. It is shown in [1] that the elongation angle satisfies

$$\sin 2\theta = \frac{b}{\sqrt{b^2 + (a-c)^2}} \; ; \quad \cos 2\theta = \frac{a-c}{\sqrt{b^2 + (a-c)^2}} \tag{4.1-4.2}$$

where

$$a = \sum x^2 \; ; \quad b = 2\sum xy \; ; \quad c = \sum y^2. \tag{4.3-4.5}$$

The summation is carried out over all the points on the shape. The corresponding minimum and maximum second moments E_{min} and E_{max} are given by

$$E_{min} = \frac{1}{2}[(a+c) - \sqrt{b^2 + (a-c)^2}] \tag{4.6}$$

and

$$E_{max} = \frac{1}{2}[(a+c) + \sqrt{b^2 + (a-c)^2}] \tag{4.7}$$

respectively. We also define the elongation ratio R_E as

$$R_E = E_{min}/E_{max}. \tag{4.8}$$

R_E is in the range $[0, 1]$. It reflects how well the shape is elongated. If the value is smaller, the shape is more elongated. If $E_{min}/E_{max} = 0$, the shape shrinks to a straight line. If $E_{min}/E_{max} = 1$, the shape is too symmetric to have a unique elongation axis. If every shape is well elongated, we may simply calculate θ for each shape and rotate it by $-\theta$ so that the elongation axis is aligned to the x axis, and the resulting elongation angle is zero. Figs. 4 (a) and (b) show two well elongated shapes of the same class at random orientation. Figs. 4(c) and (d) are the results of rotation of Figs. 4(a) and (b), respectively. After this rotation, we always start from the right-most point on the x axis to trace the shape contour counterclockwise.

Unfortunately, the starting point determined by the elongation axis method does not always yield the most similar radii sequences for the shapes of the same class. An example is shown in Fig. 5. Figs. 5(a) and (b) are two sample shapes of the same class. They are rotated by the elongation axis method. Obviously, they cannot be considered to have the same orientation. If we consider the orientation of Fig. 5(a)

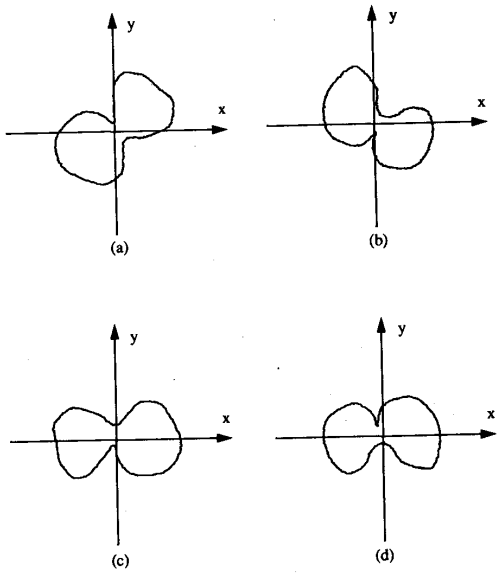


Fig. 4. Orientation results of well-elongated shapes: (a), (b) Random orientation; (c), (d) after rotation by elongation axis method.

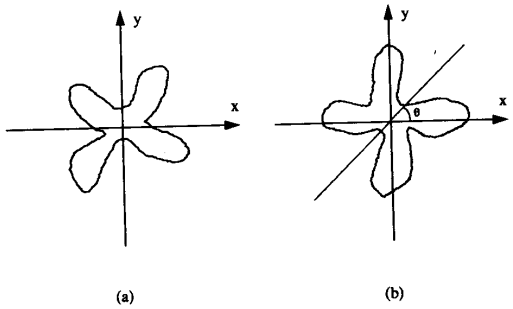


Fig. 5. Orientation results of shapes that are not well elongated: (a) Before rotation; (b) after rotation.

to be standard, we would like to rotate Fig. 5(b) again by the angle θ, as shown in Fig. 5(b).

The reason for the bad performance of the elongation axis method in this example is that the two shapes are not well elongated. The elongation ratio R_E for the shapes in Figs. 5(a) and (b) are 0.646 and 0.933, respectively. From (4.3)–(4.8), we can see that the large R_E values are related to the symmetry property of the shapes. The elongation axis of such shapes is very sensitive to shape contour perturbation. Consider another example. A shape like a + sign with four identical branches will not change if we rotate it by 90°. We may call this property the four-way symmetry property. In this case, we are not able to define an elongation axis since $a = c$ and $b = 0$. If we increase the length of one branch a little bit, the elongation axis will be along this branch. However, if we increase the length of two neighbor branches, the elongation axis can be anywhere in between these two branches. In both cases, R_E will be close to 1. A similar situation is true for the shapes with large R_E values and a three-, five-, or more-way symmetry

property. Therefore, if R_E crosses a threshold T_r, we use the alternative method described below.

Minimum Radius Point

Definition: The minimum radius point is the point on the shape whose distance to the center of gravity is the minimum compared with all the other points on the shape.

If the R_E values of all the shapes in a particular class are bigger than the threshold T_r, we may use the minimum radius point as the starting point. However, the existence of such a T_r value is not guaranteed for all classes. If, for a particular class, the R_E values of some shapes are greater than T_r but the R_E values of other shapes are smaller than T_r, then both the methods will be applied to this class.

It can be easily shown that the point corresponding to the minimum radius is very likely to be on or near the y axis due to the symmetry property. Therefore, when $R_E > T_r$, we rotate the minimum radius point to the positive y axis.

The large R_E value represents a rather symmetric shape. We may consider that the shape is the result of perturbation from a strictly symmetric shape. For the strictly symmetric shape, there are more than one minimum radius points appearing periodically along the contour. Any one of them can be used as the minimum radius point. For the perturbed shape, one of these points, or a point close to one of these points, may be chosen as the minimum radius point. It is not relevant which one of the minimum radius points of the symmetric shape becomes this point as long as the perturbed shape remains "close to symmetric". This condition is guaranteed by a large R_E value. Therefore, uniqueness is not a major concern for the minimum radius point.

It is noted that the elongation axis method is based on statistical measures and, therefore, is substantially robust for contour perturbation. The minimum radius point method, however, is sensitive to a particular characteristic point and, therefore, is sensitive to contour perturbation. This is the reason for using the minimum radius point method only when $R_E > T_r$, i.e., when the elongation axis method fails to work properly.

It is possible that the combination of the two methods still cannot rotate all the shapes of the same class to an exact orientation, but as long as the resulting orientations of the shapes in each class are in a reasonable range, there will be no serious problem. We do not use the radii sequence directly to recognize the shape. The training algorithm for getting HMM parameters, as discussed in Section III, will take the orientation bias into consideration. The effect of orientation bias is to increase the covariance of the symbol probability distribution in each state. In this respect, although our feature vector is not rotation invariant, the overall classification scheme is orientation insensitive.

Two-Way Ambiguity

After above rotation, there remains a two-way ambiguity in orientation along x axis since the second moments cannot determine the direction of elongation axis. For example, Fig. 6 is derived from a single shape. Depending on its original orientation, the results have a 180° difference in orientation. Obviously, we have to solve this two-way ambiguity and determine a unique direction.

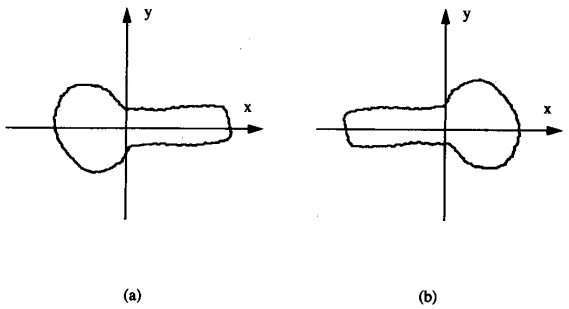


Fig. 6. Example of (a) two–way ambiguity; (b) same shape with 180° difference in orientation.

Since the center of gravity of the shape is at the coordinate origin, we have

$$\sum_{x>0} x = -\sum_{x<0} x. \tag{4.9}$$

If we cut the shape along the y axis into two pieces and consider both parts to be a 1–D signal with the same duration, then (4.9) says that the two signals have the same dc value, but they may have different energy. The part that has a flatter signal will always have less energy, whereas the part whose signal changes a lot will have more energy. We denote the energy for the $x > 0$ part and for the $x < 0$ part by E_{x+} and E_{x-}, respectively. Then

$$E_{x+} = \sum_{x>0} x^2 \; ; \quad E_{x-} = \sum_{x<0} x^2 \tag{4.10-4.11}$$

We further assume that the contour perturbation of the each class is not severe enough to change the relative value of E_{x+} and E_{x-}. Then, the difference of the energy D_{Ex} defined as

$$D_{Ex} = E_{x+} - E_{x-} \tag{4.12}$$

specifies a unique direction. If a shape has a positive D_{Ex}, we leave it as it is; otherwise, we rotate it by 180° by changing the signs of all x and y coordinates.

After the direction along the x axis is determined, there could still be two-way ambiguity along the y axis if the shape is not symmetric about the x axis. We similarly define the energy difference in the y direction as

$$D_{Ey} = E_{y+} - E_{y-} \tag{4.13}$$

where E_{y+} and E_{y-} are defined in the same way as E_{x+} and E_{x-}, respectively. If $D_{Ey} > 0$, we do not change the y direction; otherwise, we change the$y$ direction by changing the sign of all y coordinates.

Now, the complete orientation algorithm can be summarized as follows:

1) Calculate the elongation ratio R_E using (4.3)–(4.8).
2) If $R_E \leq T_r$, calculate the elongation angle θ using (4.1) and (4.2), and rotate the shape by $-\theta$. If $R_E > T_r$, find

the minimum radius point, and rotate the shape so that this point is on the positive y axis.

3) Calculate D_{Ex} and D_{Ey} using (4.10)–(4.13).
4) If $D_{Ex} < 0$ and $D_{Ey} < 0$, change the sign of both the x and y coordinates. If $D_{Ex} < 0$ and $D_{Ey} > 0$, change the sign of the x coordinates. If $D_{Ex} > 0$ and $D_{Ey} < 0$, change the sign of the y coordinates.

It is noted that Step (4) combines the two procedures for solving the two-way ambiguity in both the x and y directions into a single step.

The experimental results of this orientation algorithm will be described in Section VI with the recognition results.

V. Implementation Considerations

HMM Order and Nonstationary Transition

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. Given a decision criterion, an optimum order may be obtained, but in shape recognition, the large number of training samples required for determining the optimum order are often not available. In addition, the higher order model increases the computational complexity.

An alternative method to improve HMM performance in this respect is to consider the state transition as nonstationary, i.e., the state transition probability a_{ij} depends on t. This kind of model can be used only when multiple realization of the Markov process, i.e., multiple training samples, are available. To estimate the transition probabilities, we simply replace (3.16) in Section II by

$$\hat{a}_{ij}(t) = \frac{n_{ij}(t+1)}{n_i(t)} \tag{5.1}$$

for $1 \le i \le N$, $1 \le j \le N$, and $1 \le t \le T-1$, 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 time t. Comparing (3.16) and (5.1), it is seen that only the storage for the transition probabilities is increased but not the computation. The same situation is true for the classification procedure. In this case, we simply change $\hat{a}_{ij}$ in the Viterbi algorithm described in Section III to $\hat{a}_{ij}(t)$. The computation does not increase at all.

Test for First-Order Stationarity

The following test, as outlined by Anderson and Goodman [23], can be used to determine whether the underlying Markov chain is first-order stationary or not. Here, we test the null hypothesis

$$H : a_{ij}(t) = a_{ij} \quad (t = 1, 2, \ldots, T-1) \tag{5.2}$$

versus the alternative hypothesis. Under the alternate hypothesis, the estimates of the transition probabilities for time t are given by (5.1). The likelihood function maximized under the null hypothesis is

$$\prod_{t=1}^{T-1} \prod_{i,j} \hat{a}_{ij}^{n_{ij}(t)}. \tag{5.3}$$

The likelihood function maximized under the alternate hypothesis is

$$\prod_{t=1}^{T-1} \prod_{i,j} \hat{a}_{ij}(t)^{n_{ij}(t)}. \tag{5.4}$$

The likelihood ratio is

$$\lambda_a = \prod_{t=1}^{T-1} \prod_{i,j} \frac{\hat{a}_{ij}^{n_{ij}(t)}}{\hat{a}_{ij}(t)^{n_{ij}(t)}}. \tag{5.5}$$

We now determine the confidence region of the test. Following [21], $-2\log\lambda_a$ is distributed as a Chi-square distribution with $(T-1)N(N-1)$ degrees of freedom when the null hypothesis is true. The N here is the number of states of the underlying Markov chain. Thus, $S = -2\log\lambda_a$ is our test statistic. Since its distribution is χ^2, an α point can be computed as the threshold S_α, where α could be 95, 99.9, or 99.95%. Then, if

$$S < S_\alpha \tag{5.6}$$

the null hypothesis is accepted, i.e., the Markov chain is first-order stationary. Otherwise, the null hypothesis is rejected at a $100\% - \alpha$ level of significance, i.e., the chain is not first-order stationary, and we opt for the nonstationary model.

Normalization of the State-Optimized Likelihood Function

The criterion we use to classify the shapes into different classes is to match each shape to the models of all classes and classify the shape into the class whose state-optimized likelihood P^* is the largest among all models compared, but when the training vectors have some unfavorable distributions in the feature space, using P^* directly may cause some problems, as described below.

The value of P^* largely depends on the degree of dispersion of the feature vectors of the training samples. The degree of dispersion is reflected by the value of $|\hat{V}_j|, j = 1, \ldots, N$. If all the feature vectors of the training samples of a particular class are well clustered around its N cluster centers, all $|\hat{V}_j|$ values will be very small, and therefore, $\hat{b}_j(o)$ will be very large for an o very close to the j-th cluster center and will drop off rapidly as o goes away from the center. The consequence is that P^* will be very large if a shape is very "typical" of that class and P^* decreases very quickly as the shape goes away from the typical characteristics. On the other hand, if another class has relatively large values for all $|\hat{V}_j|$, P^* for a typical shape of this class may be smaller but decreases more slowly than that of the first class. In this case, it is possible that a shape of unknown class may have a larger P^* for the first class than for the second, even though the shape actually belongs to the second class.

To avoid this problem, we want the state-optimized likelihood function to be normalized. In the training procedure, we

find the mean value of the state-optimized likelihood functions of all training samples in each class. Then, in the classification procedure, we divide all state-optimized likelihood functions by this mean value. This is similar to the situation that $P*$'s for each class are random variables distributed around its mean value, and we classify each sample to the class having the nearest mean. From now on, we will refer to this normalized state-optimized likelihood function without changing the notation P^*. After this normalization, the P^* values are around 1 if they are calculated for the class to which the shapes actually belong.

Initial Clustering Center and Local Maxima

In Section II, we have assumed that the feature vectors have normal distribution within each state. The global convergence property of the segmental K-mean algorithm is based on this assumption. Although this is a practical and reasonable assumption, when the number of training samples is not sufficiently large, the data may not conform to this assumption very well. In this case, the clustering algorithm may not converge or converge to a local maximum when a certain initial cluster center is used. The full solution to this problem is replacing the Gaussian density by a mixture of Gaussian densities [18], but it will greatly increase the complexity of the model and, therefore, will be computationally very costly. If we do not change our model but carefully choose the initial cluster centers, we may still reach the global maximum. Thus, in the training procedures, we will try different initial cluster centers and select the set of parameters that result in the largest average P^* over all training samples of that class.

Computational Requirement

The amount of computation required for the training procedure depends on the convergence property of the training data. For each class, we need this computation only once. In classification, the Viterbi algorithm requires $N(N+1)(T-1)$ multiplications. Therefore, the total computation for P classes is about $PN(N+1)(T-1)$ multiplications plus the computation for feature extraction and rotation of the shape. Actually, we can transform the multiplications in the Viterbi algorithm to additions by taking the logarithm of all the probabilities and densities in advance.

Feature extraction requires the solution of $(M+1)$ simultaneous linear equations (see (2.3)). The number of multiplications needed for this solution is of the order of $(M+1)^3/3$. To compute the elements of the matrix in (2.3), $\frac{M(M+1)}{2}L$ multiplications are required. The computation of β requires another $(ML+1)$ multiplications. Disregarding some other minor calculations, the computational complexity, in terms of multiplications, for all the T segments of the shape to be recognized is approximately equal to $T[\frac{(M+1)^3}{3}+\frac{M(M+1)}{2}L+ML)]$. The complexity of orientation algorithm is dominated by (4.3–4.5), and $3TL$ multiplications are needed.

In our typical experiments, $T=16$, $L=16$, and $M=4$. The number of states for the HMM is typically 4 for 100% correct recognition (see Table II). Then, for eight classes of shapes, the total number of multiplications needed is about 7400, including the multiplications needed for the Viterbi algorithm. It is reasonable to state that the use of a single global AR model for shape classification requires a higher value of M, i.e, more AR coefficients. Thus, if we assume that we need six or more AR coefficients, i.e., $M=6$, to match the performance of the HMM method, the complexity for feature extraction is about $[\frac{(M+1)^3}{3}+\frac{M(M+1)}{2}TL+MTL)]=7000$ multiplications. Additional multiplications and divisions are required for classification. As a result, we conclude that the HMM and local AR model-based approach has the same level of complexity as the other comparable methods.

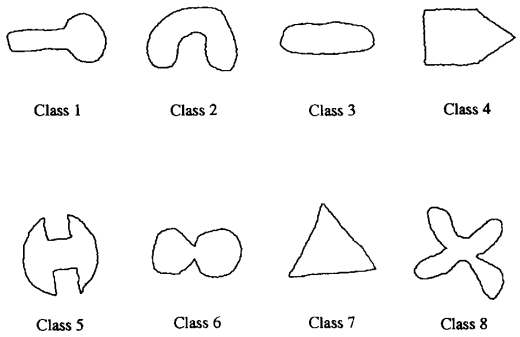


Fig. 7. Eight classes of shapes.

VI. Experimental Results

We have used two sets of data to test the algorithm. The first set includes eight classes of shapes. Some of them are similar to those used in [13]. Eight typical shapes (one from each class) are shown in Fig. 7. We created 30 samples for each class, 20 of which are used as a training sample and 10 as a 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. Three more samples each for class 2 and 8 were already seen in Fig. 1. Of the 30 samples of each class, only 24 are individually drawn. The other six 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 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; therefore, no major geometric property is changed. Some examples of occluded shapes are shown in Fig. 8. The left-most shape of each class is the original one. The other three shapes in the same row are occluded from the left-most one. It can be seen that occlusion may be considered to be a special case of perturbation.

All shapes are rotated by a random angle in between 0 and 2π. Then, the orientation algorithm described in Section IV is used to orient 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 that both methods work well and give consistent orientation

TABLE I
RECOGNITION RESULTS OF DATA SET 1: NUMBER OF ERRORS OUT OF 80 TEST SAMPLES VERSUS NUMBER OF STATES, TYPE OF TRANSITION, AND ORDER OF AR MODEL

No. of HMM States	Type of HMM Transition	Order of AR Model				
		2	3	4	5	6
2	Stationary	11	7	5	4	5
2	Nonstationary	4	2	2	2	2
3	Stationary	6	3	4	5	4
3	Nonstationary	3	2	0	0	1
4	Stationary	3	2	1	2	2
4	Nonstationary	2	0	0	0	1
5	Stationary	4	2	2	2	3
5	Nonstationary	2	1	1	2	1
6	Stationary	4	5	2	1	2
6	Nonstationary	6	4	3	2	3

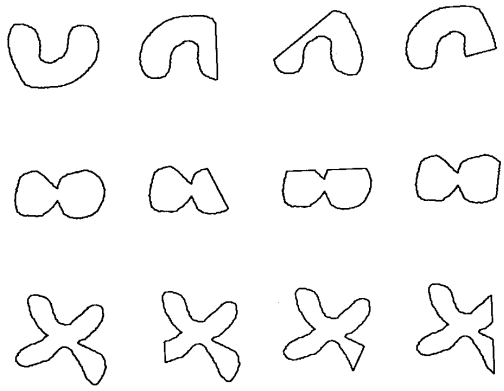


Fig. 8. Samples of moderately occluded shapes.

for classes 1, 3, 4, and 5. For classes 2 and 6, the elongation axis method gives good orientation, but the minimum radii point method does not. For classes 7 and 8, the results are the opposite of classes 2 and 6. The values of the elongation ratio R_E are all smaller than 0.55 for classes 1, 2, 3, and 6 and are all greater than 0.55 for classes 5, 7, and 8. For class 4, four shapes have an R_E value larger than 0.55, and all others are smaller than 0.55. Therefore, it turns out that 0.55 is a good choice for the value of T_r for our data set. After this threshold is set, the complete orientation algorithm is applied. For classes 1, 2, 3, 4, and 6, the shapes in each class are oriented exactly as would be done by inspection. For the rest of the classes, more than 80% of the shapes are well oriented. The other 20% of the shapes are somewhat biased after orientation, but the bias angles are within the range of -20 to $+20°$. 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 using (2.3–2.6). The segment mean is used as the $(M+2)$-th feature. The mean is scaled to the range 0–2. This range is comparable with 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 a 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 numbers of states, from $N = 2$ to $N = 6$, and both stationary and nonstationary transition for our HMM. We have also compared the results of AR models of different orders, from $M = 2$ to $M = 6$. Table I shows that the number of errors occurred in classifying the total 80 test shapes in all the above cases. 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 nonstationary transition.

It is seen from Table I that the HMM of a certain number of states gives better results. Three of the five 100% cases are with $N = 4$. With $N = 2$ 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. This 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 the AR model. For $M = 2$ to $M = 4$, the correct recognition rate tends to be higher for the model of higher order, but for $M = 4$ to $M = 6$, the increase in model order does not increase the recognition rate in most cases. This is a strong indication that

TABLE II
RECOGNITION RESULTS OF DATA SET 1: P^* OF TWO TEST SAMPLES FROM EACH CLASS WITH $N = 4, M = 4$, AND NONSTATIONARY TRANSITION. (The maximum value in each row identifies the recognized class.)

Class No.	Sample No.	Model No. 1	2	3	4	5	6	7	8
1	1	6.4	−174.5	−85.3	−15.9	−30.4	−4.1	−93.4	−621.2
1	2	3.8	−169.8	−125.7	−41.1	−50.9	−13.6	−127.6	−812.5
2	1	−1243.3	3.8	−1304.2	−144.1	−271.9	−489.1	−106.3	−499.6
2	2	−1036.4	−4.4	−1155.2	−232.4	−373.4	−443.6	−118.5	−1142.5
3	1	−28.1	−161.3	12.3	−9.9	−72.2	−41.6	−22.5	−724.0
3	2	−27.8	−177.1	5.6	−38.5	−97.9	−20.1	−132.9	−791.5
4	1	−335.5	−135.7	−395.4	−4.7	−141.0	−141.6	−120.1	−1120.5
4	2	−587.6	−106.0	−762.9	−9.4	−245.5	−254.5	−157.0	−965.5
5	1	−115.3	−218.4	−1288.3	−122.1	−24.4	−173.3	−285.6	−1035.5
5	2	−198.0	−138.0	−363.5	−65.9	5.8	−74.3	−140.7	−288.4
6	1	−48.4	−201.3	−195.3	−89.6	−45.8	2.5	−212.2	−1317.9
6	2	−40.9	−191.2	−75.2	−43.6	−88.7	−6.9	−116.0	−594.4
7	1	−385.1	−112.0	−502.6	−102.3	−180.3	−171.1	−1.8	−471.7
7	2	−553.1	−95.6	−732.9	−103.1	−236.3	−205.8	−11.6	−626.4
8	1	−621.5	−109.8	−853.8	−62.3	−137.1	−112.9	−91.6	−7.7
8	2	−464.6	−107.3	−683.7	−71.0	−151.7	−199.9	−201.8	−19.7

the shape segments are well represented by an AR model of an order no higher than 4. Collectively, the fourth- and fifth-order AR models give the best result, where each has two cases of 100% correct recognition, as seen in Table I.

Table II shows the P^* of two test samples from each class calculated against the models of all classes for $N = 4$, $M = 4$ and nonstationary transition. All values of P^* here involve a constant scaling factor applied to the computation of $\hat{b}_j(o)$ to avoid algorithmic overflow and are after taking the logarithm. Therefore, the values themselves are meaningless. Each value is compared with all other values in the same row. The maximum value in a row locates the class into which the test sample of this line is classified.

It is also seen from Table I that all five 100% recognition results have nonstationary transition. This result substantiates the assumption that the underlying Markov chain is more likely to be nonstationary. It is further supported by the experiments on the stationarity test described in Section V. Table III shows the results of the test with $N = 4$ and $M = 4$ for all eight classes. In this case, the degree of freedom is $(T - 1)N(N - 1) = 180$, and the threshold S_α with $\alpha = 95$, 99.9, and 99.95%, is 212.31, 244.87 and 249.05, respectively [24]. The data used for the test are the same as that used for training. For each class, 20 state sequences are obtained from the training procedure for the stationary model. Then, $n_{ij}(t)$, $\hat{a}_{ij}$, and $\hat{a}_{ij}(t)$ in (5.5) are calculated from the 20 state sequences for each class. It is seen from Table III that all the values of S, except for Class 8, are greater than S_α with $\alpha = 95\%$. Therefore, for Classes 1 through 7, the stationary model is rejected at a 5% level of significance. Similarly, the stationary model is rejected for Classes 1 through 6 at a 0.1% level and for Classes 2 through 6 at a 0.05% level.

TABLE III
STATIONARY TEST. ($N = 4, M = 4$, degree of freedom= 180, $S_{\alpha=95\%} = 212.31$, $S_{\alpha=99.9\%} = 244.87$, $S_{\alpha=99.5\%} = 249.05$. "Yes" indicates that the stationary model is rejected for the class at the associated level of significance.)

Class	S	Significant at? 5%	0.1%	0.05%
1	248.469	Yes	Yes	No
2	263.932	Yes	Yes	Yes
3	439.604	Yes	Yes	Yes
4	254.014	Yes	Yes	Yes
5	302.952	Yes	Yes	Yes
6	251.756	Yes	Yes	Yes
7	226.906	Yes	No	No
8	157.783	No	No	No

This is a strong evidence that the nonstationary model is more applicable.

The second set of data includes four country maps as used in [13]. These maps are shown in Fig. 9. The original country maps are scanned into the computer from the pages of [13]. Thirty samples of each country are created by roughly following the map contour on the SUN screen by manually moving the mouse. Again, six samples for each country are occluded. Then, the same experiment as used for the first data set is repeated. Table IV shows the number of errors out of the total 40 test samples used in our experiment. We observe that the same general conclusions as those drawn from the first set of data are valid. Three cases of zero recognition error occur with nonstationary models. The country maps

TABLE IV
RECOGNITION RESULTS OF COUNTRY MAP DATA SET: NUMBER OF ERRORS OUT OF 40 TEST SAMPLES VERSUS NUMBER OF STATE, TYPE OF TRANSITION, AND ORDER OF AR MODEL

No. of HMM States	Type of HMM Transition	Order of AR Model				
		2	3	4	5	6
2	Stationary	6	7	3	4	3
2	Nonstationary	5	4	2	3	2
3	Stationary	4	3	4	2	2
3	Nonstationary	3	3	1	1	1
4	Stationary	3	2	2	3	2
4	Nonstationary	2	1	0	1	1
5	Stationary	3	2	2	2	2
5	Nonstationary	2	1	0	0	1
6	Stationary	4	3	2	2	2
6	Nonstationary	4	4	1	2	3

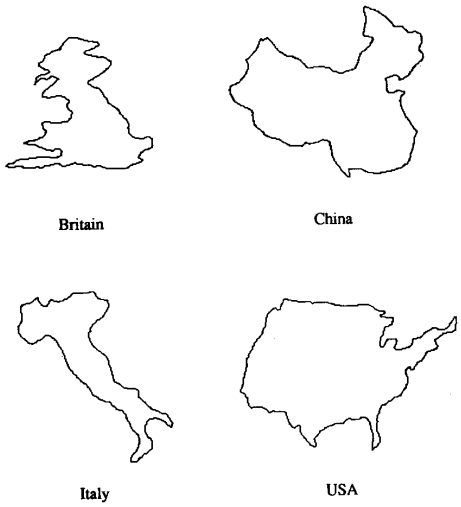


Fig. 9. Four country maps.

have more complexity than those used for the first data set. As a result, four or five state HMM's are needed for zero recognition error compared with three or four state HMM's needed for comparable results with first data set. The hidden Markov model-based scheme can handle 2–D shapes of higher complexity by making the model more complex, i.e., increasing the number of states in the model and replacing the symbol probability density by a Gaussian sum.

It is interesting to compare our data and results with those of Dubois *et al.*[15] and Das *et al.*[16]. Since all of our samples were 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, Dubois's and Das's data may be considered to be less random. Therefore, although their work and ours have all achieved a 100% recognition rate, ours has more tolerance to contour perturbation and a moderate amount of occlusion. It is important to note here that all the occluded shapes are recognized correctly with HMM of an adequate number of states and the appropriate AR model. In [15] and [16], the result is not particularly satisfactory for the occluded shapes. This improved performance of our method is due to the very nature of HMM. The feature vectors based on small segments of the shape take into account the local variation of the shape, whereas the model itself "codes" the global arrangements of these small segments. Thus, as long as the occlusion does not change any major geometric property of the shape, HMM can correctly recognize the shape. In this context, it is relevant to note here that if one object occludes the other, the occluded shape may be dominated by the geometric property of one of the shapes. In that case, HMM will be able to recognize the shape by the dominant geometric features. However, if none of the shapes dominates the other in such cases of occlusion, the HMM-based technique is not likely to recognize this scenario. The solution to this problem is to build separate HMM's to account for such cases of occlusions.

VII. CONCLUDING REMARKS

In this paper, we have presented a scheme for shape recognition, using the combination of an autoregressive model and a hidden Markov model, which makes use of the relationship between characteristics of consecutive segments of the shape. We have also developed a shape orientation algorithm. The 100% correct recognition rate we have obtained shows the effectiveness of the approach. One of the important advantages of this approach over other shape classification schemes is that when a new class of shapes is added, we do not need to train the system from the beginning since each HMM is for a single class of shapes. The experimental results also show that the approach can adequately handle the shapes with a lot of contour perturbation and a moderate amount of occlusion as well as the shapes of different size and orientation; they can still achieve a close-to-zero percent recognition error. Since it uses both global and local information, the HMM-based approach can handle highly complex planar shapes. If the

complexity of the shape increases, the model complexity may be increased accordingly, i.e., more states may be needed for the HMM. It is relevant to note here that the present paper has dealt with closed planar shapes. However, the HMM-based method can be applied to any type of shapes as long as there is a unique way to transform the shapes into a 1–D signal.

REFERENCES

[1] B. K. P. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.

[2] T. Pavlidis, "Algorithms for shape analysis of contour and waveforms," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-2, no. 4, pp. 301–312, July 1980.

[3] C. T. Zahn and R. Z. Roskies, "Fourier descriptions for plane closed curves, *IEEE Trans. Comput.*, vol. C-21, no. 3, pp. 269–281, Mar. 1972.

[4] G. H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE Trans. Comput.*, vol. C-21, , pp. 195–2–1, 1972.

[5] H. Freeman, "Computer processing in line drawing images," *Comput. Survey*, vol. 6, no. 1, pp. 57–98, Mar. 1974

[6] J. W. McKee and J. K Aggarwal, "Computer recognition of partial views of curved objects," *IEEE Trans. Comput.*, vol. C-26, no. 8, pp. 790–800, 1977.

[7] T. P. Pavlidis, "Polygonal approximation by Newton's method," *IEEE Trans. Comput.*, vol. C-26, no. 8, pp. 800–807, Aug. 1977.

[8] H. Blum, "A transformation for extracting new descriptions of shape," in *Proc. Symp. Models Perception Speech Visual Form*. Cambridge, MA: MIT Press, 1964.

[9] H. Y. Feng and T. Pavlidis, "Decomposition of polygons into simpler components: Feature extraction for syntactic pattern recognition," *IEEE Trans. Comput.*, vol. C-24, pp. 636–650, 1975.

[10] L. G. Shapiro and R. M. Haralick, "Decomposition of two-dimensional shapes by graph-theoretic clustering," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-1, pp. 10–20, 1979.

[11] F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curve and two-dimensional shapes," *IEEE Trans. Patt Anal Machine Intell.*, vol. PAMI-8, no. 1, pp. 34–44, Jan. 1986.

[12] S. P. Smith and A. K. Jain, "Chord distribution for shape matching," *Comput. Graphics Image Processing*, vol. 20, pp. 259–271, 1982.

[13] Z. You and A. K. Jain, "Performance evaluation of shape matching via chord length distribution," *Comput. Vision Graphics Image Processing*, vol. 28, pp. 185–198, 1984.

[14] R. L. Kashyap and R. Chellappa, "Stochastic models for closed boundary analysis: Representation and reconstruction," *IEEE Trans. Inform. Theory*, vol. IT-27, no. 5, pp. 627–637, Sept. 1981.

[15] S. R. Dubois and F. H. Glanz, "An autoregressive model approach to two-dimensional shape classification," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-8, no. 1, pp. 55–66, Jan. 1986.

[16] M. Das, M.J Paulik, and N. K. Loh, "A bivariate autoregressive modeling technique for analysis and classification of planar shapes," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-12, no. 1, pp. 97–103, Jan. 1990.

[17] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, pp. 4–16, Jun. 1986.

[18] L. R. Rabiner, B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Recognition of isolated digits using hidden Markov models with continuous mixture densities," *AT& T Tech. J.*, vol. 64, no. 6, pp. 1211–1234, July-Aug., 1985.

[19] B-H Juang and L. R. Rabiner, "The segmental K-means algorithm for estimating parameters of hidden Markov models," *IEEE Trans. Accoust. Speech Signal Processing*, vol. ASSP-38, no. 9, pp. 1639–1641, Sept. 1990.

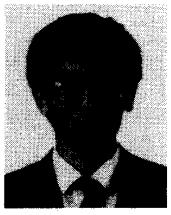
[20] L. R. Rabiner, J. G. Wilpon, and B.-J. Juang, "A segmental k-means training procedure for connected word recognition," *AT& T Tech. J.*, vol. 65, no. 3, pp. 21–31, May/June 1986.

[21] A. Kundu, Y. He, and P. Bahl, "Recognition of handwritten word: First and second order hidden Markov model based approach," *Patt. Recognition*, vol. 22, no. 3, pp. 283–297, 1989.

[22] G. D. Forney, Jr., "The Viterbi algorithm," Proc. IEEE, vol. 61, no. 3, pp. 263–278, Mar. 1973.

[23] T. W. Anderson and L. A. Goodman, "Statistical inference about Markov chains," *Annals Math. Stat.*, vol. 28, pp. 89–110, 1957.

[24] W. H. Beyer (Ed.), *Handbook of Tables for Probability and Statistics (2nd ed.)*. Cleveland, OH: Chemical Rubber, 1968.



Yang He received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 1982, the M.S. degree from Tsinghua University, Beijing, China, in 1984, and the Ph. D. degree from the State University of New York at Buffalo in 1991, all in electrical engineering.

He is currently a research associate at the Medical School, State University of New York at Buffalo. His current research interests are in pattern recognition, image processing, signal processing, and computer vision.

Amlan Kundu received the B.Tech. degree in electronics from Calcutta University, Calcutta, India, in 1979, and the M.S. and Ph.D. degrees form the University of California, Santa Barbara, in 1982 and 1985, respectively, all in electrical engineering.

Curently, he is an Assistant Professor of Electrical Engineering at the State University of New York, Buffalo. His research interests are in image processing, robust image smoothing, pattern recognition, and VLSI implementation of signal processing algorithms.