Pattern Recognition, Vol. 22, No. 3, pp. 283–297, 1989.
Printed in Great Britain.

0031-3203/89 $3.00 + .00
Pergamon Press plc
Pattern Recognition Society

RECOGNITION OF HANDWRITTEN WORD: FIRST AND SECOND ORDER HIDDEN MARKOV MODEL BASED APPROACH

AMLAN KUNDU, YANG HE and PARAMVIR BAHL
Department of Electrical Engineering, S.U.N.Y. at Buffalo, New York 14260, U.S.A.

(*Received* 4 *March* 1988; *in revised form* 8 *June* 1988; *received for publication* 1 *July* 1988)

Abstract—In this work, handwritten word recognition problem is modeled in the framework of hidden Markov model (HMM). The states of HMM are identified with the letters of the alphabet. The optimum symbols are then generated by experimental study using fifteen different features.

Both the first and second order HMM are used for the recognition task. Using the existing statistical knowledge of English, the calculation scheme for model probabilities are immensely simplified. Once the model is established, Viterbi algorithm is used to recognize the sequence of letters consisting the word. Very high recognition accuracy is obtained with the new scheme.

Handwritten script recognition Hidden Markov model (HMM) Pattern recognition
Feature selection Vector-quantizer Viterbi algorithm Model probabilities
Hypothesis generation

1. INTRODUCTION: OBJECTIVE AND SIGNIFICANCE

Recognition of handwritten script has numerous applications such as writer identification, address and zip code recognition[1–4] etc. The most common approach to this problem is the character based classical pattern recognition scheme. The singular most advantage of classical pattern recognition scheme is its conceptual simplicity and straightforward implementation once the features are known. To this date, many different features have been proposed for patterns by a host of researchers.[5–17]

One important limitation of this scheme is that it does not incorporate any language model, i.e. successive character generation model, in the recognition scheme. In this paper, we have discussed a letter-based word recognition scheme for handwritten script in the framework of hidden Markov model (HMM). The input to such recognizer is the word to be recognized; and the output is either the correctly recognized word or a small set of words which includes the "correct word" as one of its hypotheses. Since HMM is a random process model, the language which generates the successive characters in the word is modeled as a specific random process. At the same time, HMM based approach has all the simplicity of implementation of the classical pattern recognition approach.

Handwritten word recognition system described in the paper is based on: (1) segmenting the individual letters of the word, (2) transforming each segmented letter (unknown) into a symbol (pattern), and (3) recognizing, from the sequence of symbols and the incorporated model, the original word. The great advantage of such a letter-based scheme is that the addition of new words to the vocabulary does not pose any new difficulty as all the letters have their representative patterns computed and stored. However, the task of segmenting words into constituent letters is not addressed in the paper. In this context, we comment that the segmentation of words into letters is a vast and an independent research field; and considerable research efforts are under way[1,19] in this direction. Also, this problem is the focus of our future research.

Is character based recognition of words an optimal approach? One school of researchers[18] contend that human eye processes the whole sequence of letters simultaneously rather than the individual characters sequentially for recognition purpose. The application of syntax, semantics and other AI knowledge are usually applied on "hypothesized words" which are candidates for the word to be recognized.[19–21] In this light, the character based recognition as done with most of the classical pattern recognition scheme is not the optimal approach. But HMM, by incorporating a model, takes into account the particular combination of letters in a given word. In other words, some wholistic analysis of the word-image is in-built in HMM approach. This statement is particularly true for second order hidden Markov model. Equally important is the fact that the HMM based approach can give rise to a number of hypothesized words for the word to be recognized; and a measure of likelihood of these hypothesized words in relation to each other. Such high level knowledge as syntax, semantics etc. could then be used to eliminate the wrong hypotheses. This process is strikingly similar to "human reading

process".[19] Thus, the proposed scheme, in essence, is a bridge between the low-level character based recognition scheme and high-level AI knowledge based recognition scheme. Figure 1 gives an overview of the proposed scheme.

Naturally, the recognition rate achieved by the new scheme is significantly better than the existing comparable schemes. The new algorithm is not limited by the size of the vocabulary. Like the comparable schemes, the new algorithm could be applied to any handwritten script in English with straightforward segmentation clues for the words.

The paper is organized as follows. Section 2 contains the review of hidden Markov model and related research work. Section 3 describes the feature selection scheme for symbol definition. Section 4 describes the methodology and experiments for symbol generation. Section 5 details the methods and procedures used to compute the model probabilities. In Section 6, the word recognition algorithm using the known model probabilities is discussed in detail. Section 7 contains a detailed description of simulation and comparison results.

2. HIDDEN MARKOV MODEL: DESCRIPTION AND REVIEW

Random process models such as Markov model (MM) has been applied with some success to handwritten script recognition.[22] HMM is a more general random process model than MM; and is more appropriate in modeling a language such as English. HMM, in particular, has been applied by Nag *et al.*[23] But their technique needs one hidden Markov model per word in the vocabulary. In this approach,[23] for each word in the vocabulary, a HMM is constructed during the training phase. During the recognition phase, the word to be recognized is separately scored against all the models. The model with the highest score is selected as the recognized word. The proposed scheme, partly described in our earlier works,[24,25] needs only one model for the entire language. As a result, apart from substantial saving in complexity, there is a basic difference between the proposed scheme and the one described in Ref. (23), in viewpoint and in the mode of application of HMM. On the other hand, the scheme described in Ref. (26), though not stated in the language of HMM, is somewhat akin to the new scheme. However, the problem addressed in Ref. (26) is to recover the correct word from the garbled printed word, whereas the new scheme is applied to handwritten word recognition.

HMM is, also, being used in speech processing and recognition, cryptanalysis etc.,[28-30] and has proven to be quite successful at that. These research efforts have greatly contributed to the general understanding of the applicability of HMM, the complexity of parameter computation as well as the limitation of the model.

Definition

A HMM is a doubly stochastic process with an underlying stochastic process that is not observable, i.e. hidden, but can only be observed through another set of stochastic processes that produce the sequence of symbols.[27]

The following notations are now formally defined

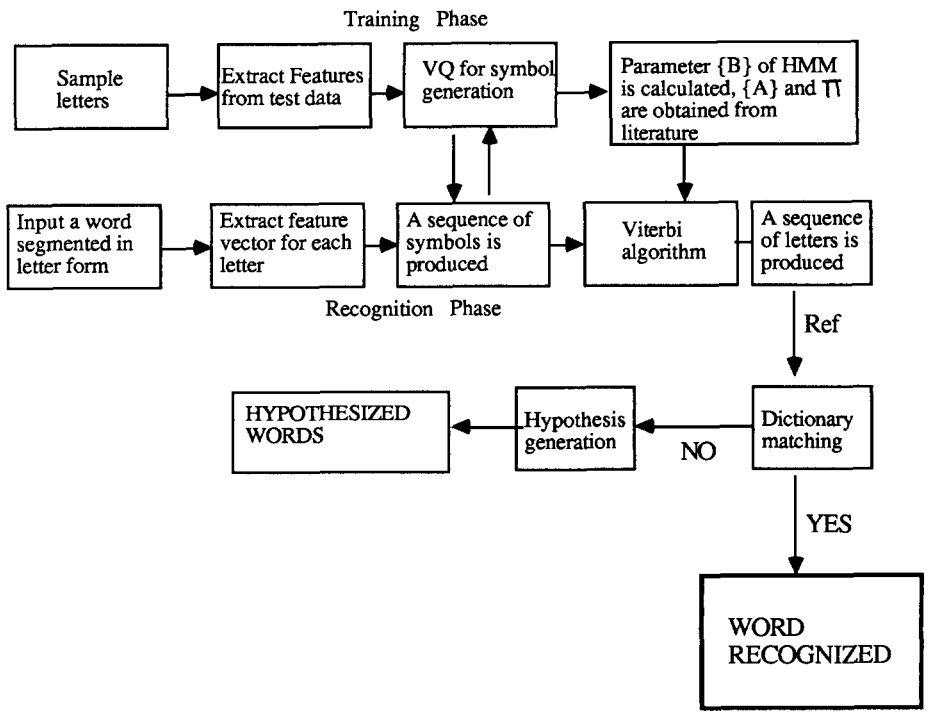


Fig. 1. Block diagram of the word recognition system.

for first order HMM. These notations are very much akin to those introduced by Rabiner and Juang.[27]

T—length of observation sequence.

Q—$\{q_1, q_2, \ldots, q_N\}$—set of states; N—number of states in the model.

V—$\{v_1, v_2, \ldots, v_M\}$—discrete set of possible symbols; M—number of observation symbols.

$A = \{a_{ij}\}$ where $a_{ij} = P(q_j \text{ at } n+1 \mid q_i \text{ at } n)$ where $1 \leq i, j \leq N$; and P stands for probability. This probability is known as the state transition probability distribution.

$B = \{b_j(k)\}$ where $b_j(k) = P(v_k \text{ at } n \mid q_j \text{ at } n)$; $k = 1, \ldots, M$. This probability is known as the symbol probability distribution in state j.

$\Pi = \{\pi_j\}$, $j = 1, \ldots, N$; $\pi_j = P(q_j \text{ at } n = 1)$. This probability gives the initial probability distribution for the states.

In HMM approach, the problem is, given the observation sequence, how to calculate the model parameters A, B, Π; and, then, given the model parameters and observation sequence, how to find the optimal state sequence corresponding to the observation sequence.

States and symbols for handwritten script

For English language, there are 26 letters in the alphabet; and 26 letters correspond to 26 states, i.e. $N = 26$. The states could then be identified as q_1 = a or A, q_2 = b or B and so on. In addition, 10 numbers $(0, 1, \ldots, 9)$ are also used in handwritten script; and will be later incorporated into the model. The next problem is the identification of the number of possible symbols (M) and their representation $\{v_1, v_2, \ldots, v_M\}$. As written style varies among the individuals, there are many possible representation of the same letter, say r. This might suggest a Markov model with increased number of states. However, different letters may look alike. For example 'l' of one person may look like 'e' of another person. That is to say that, a set of features (a symbol) that represents 'l' of one person may represent 'e' of another person. If 'l' is counted as 12th state and 'e' is counted as 5th state, the same symbol can appear in both the states (state 12 and 5). This is the rationale behind using the hidden Markov model rather than the Markov model with increased number of states.

As there are many variations of 'l' itself depending on writer identity, writing equipment variation etc., to represent 'l' under all conditions of variations, a number of symbols are required. So, although there are only 26 letters in English, the optimum number of discrete symbols are likely to be far greater, and no theoretical speculation is possible as to their total number.

Discrete symbols lead to discrete p.d.f. which is preferred here over continuous p.d.f. as discrete p.d.f. is better suited to capture the large variation in the handwritten script. The determination of discrete symbols and their total number can be accomplished by a statistical study in the framework of pattern recognition with an appropriate quantitative definition of symbols. The population for this statistical study should be as diverse as possible in relation to age, sex, ethnic and cultural difference, the difference in writing equipment etc.

3. FEATURE SELECTION

A quantitative definition of symbols, needed to build the HMM, requires the definition of each symbol as a feature vector. For example, in Ref. (30), LPC-coefficient vectors are the observed symbols. For selecting good features, the following criteria are considered useful. (1) Features should be preferably independent of rotation, translation and size. (2) Features should be easily computable. (3) Features should be chosen so that they do not replicate each other. This criterion ensures efficient utilization of information content of the feature vector.

In relation to handwritten script recognition, numerous researchers have proposed many different features. A brief but excellent account can be found in Ref. (8) by Suen *et al.* Using experimental results as our selection criterion, the following fifteen features, among 40 features initially selected, are used for the definition of symbols. Many of these features have been used by other researchers,[8,31-33] sometimes with different names or somewhat different implementation.

The written letter and the background is first segmented into a binary image with the pixels on the letter defined as 'black' (0) and background as 'white' (1). The letter is next thinned using the algorithm proposed by Pavlidis *et al.*[34]

Features based on point distribution

Let the co-ordinates of a generic black pixel be given by (u, v). For the 2D binary image of the letter, the central moments are given by

$$\mu_{pq} = \frac{1}{N} \sum_{i=1}^{N} (u_i - \bar{u})^p (v_i - \bar{v})^q \tag{1}$$

where

$$\bar{u} = \frac{1}{N} \sum u_i; \qquad \bar{v} = \frac{1}{N} \sum v_i; \tag{2}$$

and N is the total number of black points in the image. The following 3 features based on moments are described in Ref. (31) as size, orientation, rotation and translation independent.

$$M_2' = \frac{M_2}{r^4}; \qquad M_3' = \frac{M_3}{r^6}; \qquad M_4' = \frac{M_4}{r^6} \tag{3}$$

where

$$\begin{aligned} r &= (\mu_{20} + \mu_{02})^{1/2}; \\ M_2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2; \\ M_3 &= (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2; \\ M_4 &= (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2. \end{aligned} \tag{4}$$

Geometrical and topological features

Number of loops. Many letters such as 't', 'c' etc. have no loops; and are very rarely written with a distinct loop. On the other hand, letters such as 'a', 'b', 'd' etc. are almost always written with a loop. The proposed loop feature is computed as follows:[32,33]

$$f_l = \text{feature based on loop} = \begin{cases} 0 & \text{no loops} \\ 0.5 & \text{one loop} \\ 1.0 & \text{two or more.} \end{cases} \quad (5)$$

Number of T-joints. T-joints are defined as follows. Thinned image of the letter is scanned by a 3 × 3 sliding window. If at any position of the window, the central pixel is a 'black' pixel and 3 of the 8 neighboring pixels are also 'black' pixels, a T-joint is said to exist at that pixel position. The feature f_t is computed as follows:[33]

$$f_t = \text{feature based on T-joint} = \begin{cases} 0 & \text{no T-joint} \\ 0.5 & \text{one T-joint} \\ 1.0 & \text{two or more.} \end{cases} \quad (6a)$$

Number of X-joints. In the thinned image, if the central pixel of the 3 × 3 window is a 'black' pixel and 4 (or more) out of 8 neighboring pixels are 'black' pixels; an X-joint is said to exist at that pixel position. The feature f_x is defined as follows:[33]

$$f_x = \text{feature based on X-joint} = \begin{cases} 0 & \text{no X-joint} \\ 0.5 & \text{one X-joint} \\ 1.0 & \text{two or more.} \end{cases} \quad (6b)$$

The algorithms to find T or X joints incorporate a number of heuristics to safeguard against false alarms.

Scaled vertical to horizontal (V-to-H) distance. Such letters as 'i' and 'j' have a high V-to-H distance where letters such as 'm' and 'w' have a low V-to-H distance. The vertical distance is calculated as the range of Y-coordinates; and the horizontal distance is calculated as the range of X-coordinates, respectively, of the thinned image. The top and bottom 10% of V-to-H distances are discarded. In the trimmed sample of V-to-H distances, the maximum is scaled to 1.0 and the minimum is scaled to 0.2. Subsequently, for any letter with scaled V-to-H distance more than 1.0, this feature is scaled to 1.0; and for V-to-H distance less than 0.2, the feature is scaled to 0.2.

Isolated dots. Letters such as 'i' and 'j' have isolated dots which could be a good clue to their recognition.

$$f_i = \begin{cases} 0 & \text{no isolated point} \\ 1.0 & \text{one or more.} \end{cases} \quad (7)$$

Zero crossing. Through the c.g. (calculated with the moment features) of thinned letter, an horizontal line is drawn. The number of intersections of this line with the letter gives the number of zero crossings (Fig. 2).

f_{zh} = feature based on horizontal zero-crossings

$$= \begin{cases} 0 & \text{no crossing} \\ 0.25 & \text{one crossing} \\ 0.5 & \text{two crossings} \\ 0.75 & \text{3 crossings} \\ 1.0 & \text{4 or more.} \end{cases} \quad (8)$$

Similarly, one could compute the vertical zero crossings f_{zv}.

End points. End points are calculated in the following fashion.[33] For the sliding 3 × 3 window, if the central pixel is '0' and only one of the neighboring eight pixels is '0', the central pixel is considered an end point (EP).

f_{ep} = feature based on end points

$$= \begin{cases} 0 & \text{no EP} \\ 0.25 & \text{one EP} \\ 0.5 & \text{two EPs} \\ 0.75 & \text{3 EPs} \\ 1.0 & \text{4 or more.} \end{cases} \quad (9)$$

Approximate semi-circle in EWNS direction. If split along vertically through the c.g. 'a' has a semi-circle in the west half; and 'b' has a semi-circle in the east half. If split horizontally, 'm' has two semi-circles in the north half and 'w' often has two in the south half.[32]

f_{se} = feature based on semi-circle in east-direction

$$= \begin{cases} 0 & \text{no semi-circle} \\ 0.5 & \text{one semi-circle} \\ 1.0 & \text{two or more.} \end{cases} \quad (10)$$

The semicircle is determined by detecting continuous unambiguous loop formed by part of the letter and the drawn vertical or horizontal line through the c.g. (see Fig. 2). Similarly for the other features f_{sw}, f_{sn}, f_{ss}.

Observe that all the features are scaled in the range 0.0–1.0. The moment features, by virtue of their definition, are also scaled in the same range. Such scaling is important to ensure that no feature gets more or less weight unless otherwise intended.

4. SYMBOL GENERATION

Vector-quantizer for symbol generation

Let there be a sample of I letters contributing to I training vectors $\underline{s}_i$, $i = 1, \ldots, I$. This is the training set. The main idea behind the vector quantization algorithm is to determine the optimum set of M feature vectors, called codebook, such that the average distortion in replacing each of the training set vectors $\underline{s}_i$, $i = 1, \ldots, I$, by the closest entry in the codebook, $\hat{s}_m$, is minimum. Formally stated, if we define $d(\underline{s}_R, \underline{s}_T)$ as

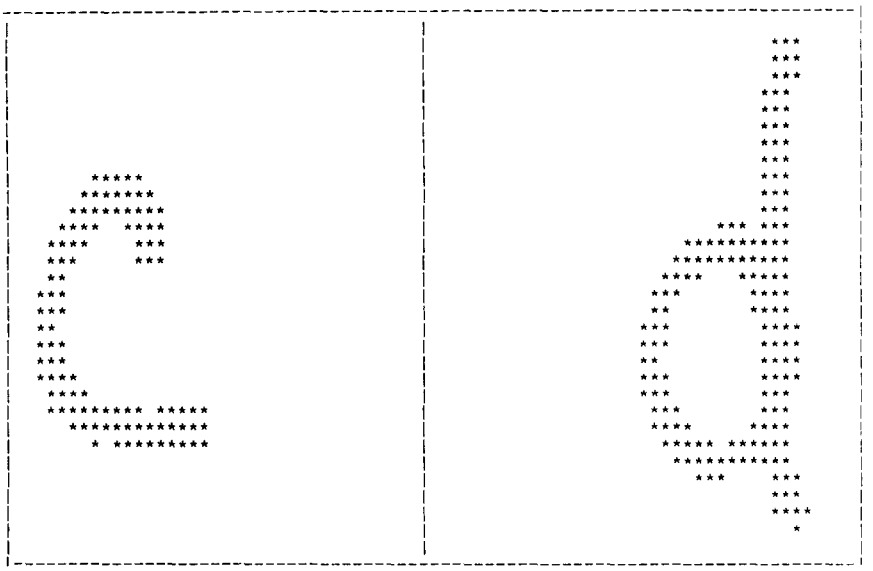


Figure 2 (a)

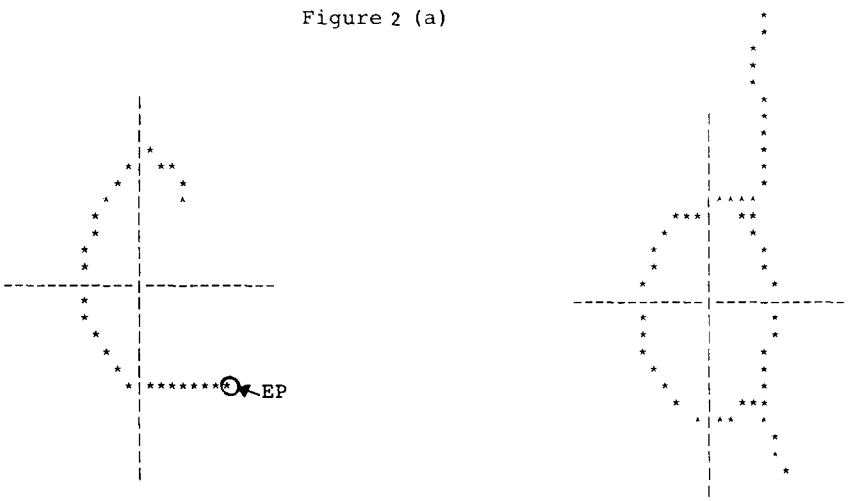


Figure 2 (b)

Fig. 2. (a) Original handwritten letter. (b) Thinned letter with horizontal and vertical axis through the C.G. (equation 2) (no. of vertical zero-crossings 2, horizontal zero-crossings 1 for the letter 'c' etc.).

the distance between two vectors $\underline{s}_R$ and $\underline{s}_T$, then the goal of vector quantizer is to find the set $\hat{s}_m$, for a given M, such that

$$[D_M] = \min_{\hat{s}_m} \frac{1}{I} \sum_{i=1}^{I} \min_{1 \le m \le M} [d(\hat{s}_m, \underline{s}_i)] \qquad (11)$$

is satisfied. The quantity $[D_M]$ is the average distortion of the vector quantizer. In the present case, the feature vectors are as described before in Section 3. The set of symbols $\{v_j, j = 1, \ldots, M\}$, for a given M, constitutes the codebook. The distance measure to be used in the proposed scheme is the unweighted Euclidean distance measure.[35] The number M, the optimum number of symbols, is generally determined experimentally. For large sample, usually a point is reached when increasing M does not decrease $[D_M]$ in any significant manner. The value of M around this vicinity is a good experimental choice. For practical implementation of *V-Q*, a number of implementation schemes are available and details could be found in Ref. (35).

With nearly 2500 sample of letters, we have generated 90 optimum symbols using the vector quantizer algorithm.[35] Once the optimum symbols are determined, the training set vectors $\underline{s}_i$, $i = 1, \ldots, I$, are classified, according to minimum distance criterion, as symbols $v_j, j = 1, \ldots, M$, which are then used to calculate the symbol probabilities as given by equation (12).

5. CALCULATION OF MODEL PARAMETERS

A: *Model probabilities for first order model*

State transition probability. In cryptography, one major task is to calculate the probability of generation of a given plaintex.[36] In order to do so, the English language is modeled as a Markov source with 26 states. The 1-state transition probabilities and initial probabilities are then determined by statistical studies. Konheim[36] reports one such study based on 67,320 1-state transition between two successive letters in the English language, and the result is shown in Fig. 3. Since the states in our proposed HMM are the letters of English language, a_{ij} probabilities (transition probability from state i to state j) of the proposed HMM are given by the probabilities of Fig. 3. Equilibrium distribution of the Markov process as reported by Konheim, however, cannot be taken as initial probabilities since the underlined Markov process cannot be assumed stationary.

Transition Probabilities P = (p(j/i))

	A	B	C	D	E	F	G	H	I	J	K	L	M
A	0.0011	0.0193	0.0388	0.0469	0.0020	0.0100	0.0233	0.0020	0.0480	0.0020	0.0103	0.1052	0.0281
B	0.0931	0.0057	0.0016	0.0008	0.3219	0.0000	0.0000	0.0000	0.0605	0.0057	0.0000	0.1242	0.0049
C	0.1202	0.0000	0.0196	0.0004	0.1707	0.0000	0.0000	0.1277	0.0761	0.0000	0.0324	0.0369	0.0015
D	0.1044	0.0020	0.0026	0.0218	0.3778	0.0007	0.0132	0.0007	0.1803	0.0033	0.0000	0.0125	0.0178
E	0.0660	0.0036	0.0433	0.1194	0.0438	0.0142	0.0125	0.0021	0.0158	0.0005	0.0036	0.0456	0.0340
F	0.0838	0.0000	0.0000	0.0000	0.1283	0.0924	0.0000	0.0000	0.1608	0.0000	0.0000	0.0299	0.0009
G	0.1078	0.0000	0.0000	0.0018	0.2394	0.0000	0.0177	0.1281	0.0839	0.0000	0.0000	0.0203	0.0027
H	0.1769	0.0005	0.0014	0.0008	0.5623	0.0000	0.0000	0.0005	0.1167	0.0000	0.0000	0.0016	0.0016
I	0.0380	0.0082	0.0767	0.0459	0.0437	0.0129	0.0280	0.0002	0.0016	0.0000	0.0050	0.0567	0.0297
J	0.1259	0.0000	0.0000	0.0000	0.1818	0.0000	0.0000	0.0000	0.0350	0.0000	0.0000	0.0000	0.0000
K	0.0395	0.0028	0.0000	0.0028	0.5282	0.0028	0.0000	0.0198	0.1582	0.0000	0.0113	0.0198	0.0028
L	0.1342	0.0019	0.0022	0.0736	0.1918	0.0105	0.0108	0.0000	0.1521	0.0000	0.0079	0.1413	0.0082
M	0.1822	0.0337	0.0026	0.0000	0.2975	0.0010	0.0000	0.0000	0.1345	0.0000	0.0000	0.0010	0.0654
N	0.0550	0.0004	0.0621	0.1681	0.1212	0.0102	0.1391	0.0013	0.0665	0.0009	0.0066	0.0073	0.0104
O	0.0085	0.0101	0.0162	0.0231	0.0037	0.1299	0.0082	0.0025	0.0092	0.0014	0.0078	0.0416	0.0706
P	0.1359	0.0000	0.0006	0.0000	0.1747	0.0000	0.0000	0.0237	0.0423	0.0000	0.0000	0.0812	0.0073
Q	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
R	0.1026	0.0033	0.0172	0.0282	0.2795	0.0031	0.0175	0.0017	0.1181	0.0000	0.0205	0.0164	0.0303
S	0.0604	0.0012	0.0284	0.0027	0.1795	0.0024	0.0000	0.0561	0.1177	0.0000	0.0091	0.0145	0.0112
T	0.0619	0.0003	0.0036	0.0002	0.1417	0.0007	0.0002	0.3512	0.1406	0.0000	0.0000	0.0101	0.0044
U	0.0344	0.0415	0.0491	0.0243	0.0434	0.0052	0.0382	0.0010	0.0258	0.0000	0.0014	0.1097	0.0329
V	0.0749	0.0000	0.0000	0.0023	0.6014	0.0000	0.0000	0.0000	0.2569	0.0000	0.0000	0.0000	0.0012
W	0.2291	0.0008	0.0000	0.0032	0.1942	0.0000	0.0000	0.1422	0.2104	0.0000	0.0000	0.0041	0.0000
X	0.0672	0.0000	0.1119	0.0000	0.1269	0.0000	0.0000	0.0075	0.1119	0.0000	0.0000	0.0000	0.0075
Y	0.0586	0.0034	0.0103	0.0069	0.2897	0.0000	0.0000	0.0000	0.0690	0.0000	0.0034	0.0172	0.0379
Z	0.2278	0.0000	0.0000	0.0000	0.4557	0.0000	0.0000	0.0000	0.2152	0.0000	0.0000	0.0127	0.0000

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0.1878	0.0008	0.0222	0.0000	0.1180	0.1001	0.1574	0.0137	0.0212	0.0057	0.0026	0.0312	0.0023
B	0.0000	0.0964	0.0000	0.0000	0.0662	0.0229	0.0049	0.0727	0.0016	0.0000	0.0000	0.1168	0.0000
C	0.0011	0.2283	0.0000	0.0004	0.0426	0.0087	0.0893	0.0347	0.0000	0.0000	0.0000	0.0094	0.0000
D	0.0053	0.0733	0.0000	0.0007	0.0324	0.0495	0.0013	0.0601	0.0099	0.0040	0.0000	0.0264	0.0000
E	0.1381	0.0040	0.0192	0.0034	0.1927	0.1231	0.0404	0.0048	0.0215	0.0205	0.0152	0.0121	0.0004
F	0.0009	0.2789	0.0000	0.0000	0.1215	0.0026	0.0496	0.0462	0.0000	0.0000	0.0000	0.0043	0.0000
G	0.0451	0.1140	0.0000	0.0000	0.1325	0.0256	0.0247	0.0512	0.0000	0.0000	0.0000	0.0053	0.0000
H	0.0038	0.0786	0.0000	0.0000	0.0153	0.0027	0.0233	0.0085	0.0000	0.0011	0.0000	0.0041	0.0000
I	0.2498	0.0893	0.0100	0.0008	0.0342	0.1194	0.1135	0.0011	0.0250	0.0000	0.0023	0.0002	0.0079
J	0.0000	0.3147	0.0000	0.0000	0.0070	0.0000	0.0000	0.3357	0.0000	0.0000	0.0000	0.0000	0.0000
K	0.0565	0.0198	0.0000	0.0000	0.0085	0.1102	0.0028	0.0028	0.0000	0.0000	0.0000	0.0113	0.0000
L	0.0004	0.0778	0.0041	0.0000	0.0034	0.0389	0.0254	0.0269	0.0056	0.0011	0.0000	0.0819	0.0000
M	0.0042	0.1246	0.0722	0.0000	0.0026	0.0244	0.0005	0.0337	0.0005	0.0000	0.0000	0.0192	0.0000
N	0.0194	0.0528	0.0004	0.0007	0.0011	0.0751	0.1641	0.0124	0.0068	0.0018	0.0002	0.0157	0.0004
O	0.2190	0.0222	0.0292	0.0000	0.1530	0.0357	0.0396	0.0947	0.0334	0.0345	0.0012	0.0041	0.0004
P	0.0006	0.1511	0.0581	0.0000	0.2306	0.0180	0.0287	0.0457	0.0000	0.0000	0.0000	0.0017	0.0000
Q	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000
R	0.0325	0.1114	0.0055	0.0000	0.0212	0.0655	0.0596	0.0192	0.0142	0.0017	0.0002	0.0306	0.0000
S	0.0021	0.0706	0.0386	0.0009	0.0027	0.0836	0.2483	0.0579	0.0000	0.0039	0.0000	0.0081	0.0000
T	0.0015	0.1229	0.0003	0.0000	0.0479	0.0418	0.0213	0.0195	0.0005	0.0088	0.0000	0.0203	0.0005
U	0.1517	0.0019	0.0386	0.0000	0.1460	0.1221	0.1255	0.0029	0.0014	0.0000	0.0010	0.0014	0.0005
V	0.0000	0.0530	0.0000	0.0000	0.0000	0.0023	0.0000	0.0012	0.0012	0.0000	0.0000	0.0058	0.0000
W	0.0357	0.1292	0.0000	0.0000	0.0106	0.0366	0.0016	0.0000	0.0000	0.0000	0.0000	0.0024	0.0000
X	0.0000	0.0075	0.3507	0.0000	0.0000	0.0000	0.1716	0.0000	0.0000	0.0000	0.0373	0.0000	0.0000
Y	0.0172	0.2207	0.0310	0.0000	0.0310	0.1517	0.0172	0.0138	0.0000	0.0103	0.0000	0.0069	0.0034
Z	0.0000	0.0506	0.0000	0.0000	0.0000	0.0000	0.0000	0.0127	0.0000	0.0000	0.0000	0.0000	0.0253

Fig. 3. First order state transition probability.

Letter	Initial Prob.	Letter	Initial Prob.
A :	0.0569462	N :	0.023779
B :	0.0544431	O :	0.021902
C :	0.0963705	P :	0.082603
D :	0.0550688	Q :	0.005006
E :	0.0350438	R :	0.052565
F :	0.0475594	S :	0.102628
G :	0.0312891	T :	0.056946
H :	0.0350438	U :	0.025031
I :	0.0431790	V :	0.023153
J :	0.0087610	W :	0.041301
K :	0.0068836	X :	0.000625
L :	0.0294118	Y :	0.004380
M :	0.0575720	Z :	0.002503

Fig. 4. Initial state probabilities.

Initial state probability. The initial probabilities can be calculated in the following way.

The meaning of the initial probability π_j is the probability that a word could start with a letter depicting the state 'j'. In other words, given the dictionary of all the words that the recognizer is supposed to recognize, π_j, to a good approximation, is that fraction of the words in the dictionary that starts with the letter depicting the state 'j'. A commonly used dictionary such as the Random House dictionary or the Merriam-Webster dictionary would do fine in this regard. For instance, using the Merriam-Webster dictionary, the initial probability for letter 'z' is $\pi_{26} = 0.002$ where z is the 26th state, and for 's', $\pi_{19} = 0.1026$ where s is the 19th state. Figure 4 gives the table of initial probability.

Symbol probability. Once the training data set of handwritten letters are classified into symbols by *V-Q* algorithm, $b_j(k)$ probabilities are found by the following equation.

$$b_j(k) = \frac{\gamma_j(k)}{\gamma_j} \tag{12}$$

γ_j = total number of times the letter depicting the state j appear in the sample

$\gamma_j(k)$ = total number of times the symbol k is observed while the state is 'j'.

Needless to say that the symbol probability as given by equation (12) could approach the actual probability only when the sample size is very big indeed. Some other considerations are important too. It is possible that some symbols may not appear in a particular state 'j' at all, and the corresponding probability is zero according to equation (12). To all such zero probability, a small probability is assigned; and other probabilities are normalized accordingly (Fig. 5). If these probabilities are assigned zero value, some genuine alternate or second alternate choice would be lost as the optimum probabilities, in the recognition phase, are calculated using the model probabilities in a multiplicative fashion. The particular heuristic will be made clear in the next section.

B: *Model probabilities for second order HMM*

State transition probability. A typical transition probability for the second order model is $P(a \mid bc)$ or $P(1 \mid 23)$. In English language, the transition from two consecutive consonants to a succeeding consonant is infrequent. Similarly, the transition from two consecutive vowels to a succeeding vowel is also infrequent. Additionally, such transition as given by $P(a \mid aa)$ is non-existent. It is then conceivable that the second order model can represent the pecularities of the language more efficiently than the first order model. The first order model assumes that such probabilities as $P(a \mid ae)$, $P(b \mid ae)$, $P(c \mid ae)$ etc. are all independent of e. Figure 6 shows a small segment of all second order transition probabilities. Comparing Fig. 6 with Fig. 3 reveals that $P(a \mid ae)$ is not equal to $P(a \mid a)$. Thus, at least theoretically, the second order model has the potential to outperform the first order model. The biggest advantage with the first order model, on the other hand, is the simplicity in implementation. The second order state transition probabilities are not available in literature. We calculate these transition probabilities in the following manner.

In our VMS-based VAX-785 machine, a 92,000 strong word dictionary is available. Every word in the dictionary is searched; and all the second order transitions are recorded. For instance, the word 'five' has two second order transitions—(1) to 'v' from 'fi'; and (2) to 'e' from 'iv'. After all the transitions are recorded, the probabilities are calculated using the following equation.

	A	B	C	D	E	F	G	H	I
1 :	0.0010	0.0010	0.8049	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
2 :	0.0010	0.0010	0.0010	0.0010	0.1093	0.0010	0.0010	0.0010	0.0010
3 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0137	0.0010	0.0010
4 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.1578
5 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
6 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0684	0.0010
7 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
8 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
9 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
10 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
11 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
12 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
13 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
14 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
15 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
16 :	0.0010	0.0957	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
17 :	0.0010	0.0010	0.0010	0.0010	0.1366	0.0010	0.0010	0.0010	0.0010
18 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
19 :	0.0010	0.0010	0.0546	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
20 :	0.0010	0.0684	0.0010	0.0137	0.0010	0.0010	0.1097	0.0010	0.0010
21 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
22 :	0.0010	0.0010	0.0010	0.0010	0.6419	0.0010	0.0010	0.0010	0.0010
23 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.1777	0.0010
24 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
25 :	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.3691	0.0010

Fig. 5. Symbol probability distribution (for 25 symbols over states A–I).

$$P(i \mid jk) = \frac{\text{No. of transitions from state } j \text{ at } (n-1) \text{ and } k \text{ at } (n-2) \text{ to state } i \text{ at } n}{\text{No. of transitions from state } j \text{ at } (n-1) \text{ and } k \text{ at } (n-2) \text{ to any state at } n}. \tag{13}$$

Figure 6 show some of these transition probabilities.

Initial probability. For the second order model, the initial probabilities are probabilities as given by Fig. 4 as well as the first order transition probabilities as given by Fig. 3.

Symbol probability. Since, the letters of the alphabet are identified with the states, the difference in writing style etc., which leads to symbol distribution, does not affect the state transition probabilities, first order or second order; or vice-versa. Thus, the symbol probabilities for the second order model is also given by Fig. 5, the symbol probabilities for the first order model.

Transition	Probability	Transition	Probability
EA ---> A	0.0000	EA ---> N	0.0583
EA ---> B	0.0465	EA ---> O	0.0000
EA ---> C	0.0476	EA ---> P	0.0179
EA ---> D	0.1358	EA ---> Q	0.0005
EA ---> E	0. 0008	EA ---> R	0.1757
EA ---> F	0.0140	EA ---> S	0.1078
EA ---> G	0.0129	EA ---> T	0.1465
EA ---> H	0.0022	EA ---> U	0.0187
EA ---> I	0.0000	EA ---> V	0.0231
EA ---> J	0.0000	EA ---> W	0.0052
EA ---> K	0.0487	EA ---> X	0.0008
EA ---> L	0.0896	EA ---> Y	0.0000
EA ---> M	0.0450	EA ---> Z	0.0022

Fig. 6. A small segment of (26 × 26 × 26) two-state transition probabilities.

6. RECOGNITION OF UNKNOWN WORDS

For a given test word, each unknown letter in the word is first transformed into its representative symbol by comparing it with all the symbols in the codebook according to minimum distance criterion. Assuming that the test word is T time unit long, this procedure yields an observation sequence of T unit given by $O_1, O_2, \ldots, O_T$. For example, if $T = 4$, the observation sequence could be symbol No. 50, 23, 11, 72 from the set of 90 symbols. The recognition problem is that, given these observed symbols, what is the best sequence of states, i.e. letters. If $P(\underline{X} \mid \underline{O})$ gives the probability of choosing the state sequence $\underline{X}$ given the observed symbol sequence $\underline{O}$, the optimal state sequence corresponds to the maximum value of this probability. This optimality criterion is known as maximum *a posteriori* probability (MAP) criterion. Using MAP criterion, the solution to the problem is 'Viterbi algorithm'.

Viterbi algorithm for first order model

Step 1: initialization. Define

$$\delta_1(i) = \pi_i b_i(O_1); \qquad 1 \le i \le N; \tag{14a}$$

$$\varphi_1(i) = 0. \tag{14b}$$

Here, $\delta_1(i)$ is the probability that symbol O_1 occurs at time $n = 1$ and at state i. The variable φ stores the

optimal states; and are initialized by equation (14b).

Step 2: recursive computation. For $2 \leq n \leq T$ and $1 \leq j \leq N$

$$\delta_n(j) = \max_{1 \leq i \leq N} (\delta_{n-1}(i)a_{ij})b_j(O_n). \tag{15a}$$

The right-hand side of equation (15a) is the maximum value for the probability that the partial sequence $O_1, O_2, \ldots, O_n$ occurs at state j at time n where $n \leq T$; and j could be any state depicting A to Z.

$$\varphi_n(j) = \operatorname*{argmax}_{1 \leq i \leq N} (\delta_{n-1}(i)a_{ij}) \tag{15b}$$

$\varphi_n(j)$ stores the value of the state 'i' at $n-1$ that makes the probability $\delta_n(j)$ the highest, i.e. the most probable state is 'i'. For example, at $n = 2$ and $j = 1$ (state a or A), if $\delta_1(3)a_{31}$ is the highest, then $\varphi_2(1) = 3$.

Step 3: terminal states.

$$p^* = \max_{1 \leq i \leq N} [\delta_T(i)]; \tag{16a}$$

$$i_T^* = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_T(i)]. \tag{16b}$$

At the final time unit $n = T$, there are N probabilities $\delta_T(i)$, $i = 1, \ldots, N$. The highest probability of these probabilities is the candidate for the optimal state sequence. Equation (16b) stores the corresponding state, the terminal state. The final task, now, is to backtrack from the state given by equation (16b) to the initial state following the variable φ_n.

Step 4: backtracking. For $n = T-1, T-2, \ldots, 1$, compute

$$i_n^* = \varphi_{n+1}(i_{n+1}^*). \tag{16c}$$

Hypothesis generation and recognition

From the recognition algorithm discussed above, it is clear that a very badly written letter in the word may lead to wrong identification of the word. A simple heuristic may immensely improve the situation. Step 3 of Viterbi algorithm chooses the maximum probability and the corresponding terminal state. This terminal state is then used to backtrack the states which yield the optimal best state sequence. It is also possible to choose the second maximum (one below the maximum) probability and the corresponding terminal state as an alternate. This information could be used to generate the second hypothesized word for the input word to be recognized. Figure 7 depicts this situation clearly. This idea can be extended to symbol identification. Each segmented character, instead of being mapped to the closest neighbor, could be mapped to two nearest neighbors. For any segmented character, however, if the ratio of the second closest distance to the closest distance exceeds some threshold, say 1.5, only the closest symbol is considered. In this manner, a word of length T could give rise to 2^T symbol strings, each of length T. As each symbol string can lead to two hypothesized words, the maximum number of hypothesized words could be $2 \cdot 2^T$. Figure 8(a) shows all the hypothesized words with the corresponding probabilities as calculated by 'Viterbi algorithm' for the word 'sample'. If the probability is low or the hypothesized word is not linguistically correct, it is discarded from the set of hypotheses as shown in Fig. 8(a).

Viterbi algorithm for second order model

The essence here is to modify the second order model into a first order model by increasing the number of states. Consider the second order transition probability $P(X_3 \mid X_2X_1)$. Let us define $Y_3 = X_3X_2$ and $Y_2 = X_2X_1$. Then, $P(Y_3 \mid Y_2) = P(X_3 \mid X_2X_1)$. In other words, instead of computing $N \times (N \times N)$ second order transition, we have to compute

		SECOND ORDER HMM					FIRST ORDER HMM					
No.	Word	1st Choice	Probability	2nd Choice	Probability	Selection	1st Choice	Probability	2nd Choice	Probability	Selection	Overall
1	CAT	CAT	2.03×10^{-5}	CAL	3.22×10^{-7}	CAT	CAT	1.23×10^{-5}	CAX	3.90×10^{-07}	CAT	CAT
2	HELP	HELP	4.57×10^{-8}	HELE	3.73×10^{-9}	HELP	HELP	4.19×10^{-8}	HELE	6.83×10^{-9}	HELP	HELP
3	FOOD	FOOD	3.51×10^{-7}	FOOC	2.73×10^{-9}	FOOD	FOOD	1.94×10^{-8}	FOOC	2.09×10^{-9}	FOOD	FOOD
4	BYE	PRE	9.18×10^{-7}	BLE	6.72×10^{-7}	PRE	BYE	2.61×10^{-5}	BYO	7.25×10^{-8}	BYE	BYE
5	COLD	COLD	1.76×10^{-9}	COAD	2.58×10^{-11}	COLD	COLD	6.16×10^{-10}	COLE	3.76×10^{-12}	COLD	COLD
6	QUICK	QUICK	4.45×10^{-10}	QUICA	8.55×10^{-12}	QUICK	QUICK	2.07×10^{-11}	QUICO	4.99×10^{-13}	QUICK	QUICK
7	HOPE	HOPE	7.54×10^{-7}	HOBE	3.40×10^{-7}	HOPE	HOBE	1.78×10^{-7}	HOPR	3.34×10^{-10}	HOBE	HOPE
8	WORK	WORT	3.69×10^{-7}	WORK	1.82×10^{-7}	WORK	WORT	7.88×10^{-8}	WORK	4.22×10^{-8}	WORK	WORK
9	EXAM	EXAM	8.95×10^{-8}	EXAC	5.01×10^{-10}	EXAM	ETAM	3.92×10^{-9}	ETAN	1.62×10^{-10}	ETAM	EXAM
10	IMAGE	IMAGE	9.65×10^{-11}	IMACE	7.75×10^{-12}	IMAGE	IMADE	1.20×10^{-10}	IMACE	2.08×10^{-11}	IMADE	IMAGE
11	MONEY	MONEY	1.36×10^{-9}	MONES	7.21×10^{-10}	MONEY	MONEP	1.54×10^{-9}	MONEY	9.40×10^{-10}	MONEY	MONEY
12	LONG	LONG	8.19×10^{-8}	LOND	1.73×10^{-8}	LONG	LONG	3.50×10^{-7}	LOND	8.44×10^{-8}	LONG	LONG
13	APPLE	APPLE	5.62×10^{-8}	APPIE	2.25×10^{-10}	APPLE	APPLE	2.15×10^{-9}	APPLI	2.77×10^{-12}	APPLE	APPLE
14	SHORT	SHORT	2.30×10^{-8}	SHOLT	7.77×10^{-10}	SHORT	SHORT	4.22×10^{-9}	SHORE	1.14×10^{-10}	SHORT	SHORT
15	JUMP	JUMP	3.03×10^{-7}	JUMB	7.50×10^{-8}	JUMP	JUMP	1.55×10^{-7}	JUMB	1.38×10^{-8}	JUMP	JUMP
16	WALK	VALK	7.53×10^{-10}	VALI	4.04×10^{-11}	VALK	WALK	3.98×10^{-10}	WALE	3.30×10^{-11}	WALK	WALK
17	PAPER	PAPER	3.03×10^{-8}	PAPAR	2.53×10^{-11}	PAPER	PAPER	1.69×10^{-8}	PAPEX	3.57×10^{-11}	PAPER	PAPER
18	SAMPLE	KAMPLE	9.96×10^{-12}	KAMPLA	2.22×10^{-14}	KAMPLE	KAMPLE	2.08×10^{-12}	KAMPLI	2.68×10^{-13}	KAMPLE	KAMPLE *
19	KEYS	KERS	7.99×10^{-10}	KEIS	5.70×10^{-10}	KERS	KEIS	9.82×10^{-9}	KEIN	8.40×10^{-11}	KEIS	KEIS *
20	CHAIR	CHANT	2.77×10^{-11}	CHAIR	1.62×10^{-11}	CHANT	CHANT	7.69×10^{-11}	CHANL	1.70×10^{-11}	CHANT	CHANT
						CHAIR						CHAIR

Fig. 7. Recognition results using first and second order hidden Markov model based approach. Probability stands for maximum probability as calculated by Viterbi algorithm. (See Fig. 8(b) for further results on unrecognized words.)

$(N \times N) \times (N \times N)$ first order transition. So, Viterbi algorithm as outlined before could now be used. But the number of state transition probability is N^4; not N^2 as with the first order model. Of these N^4, only N^3 transitions are possible at each step. This fact leads to considerable reduction in algorithmic complexity.

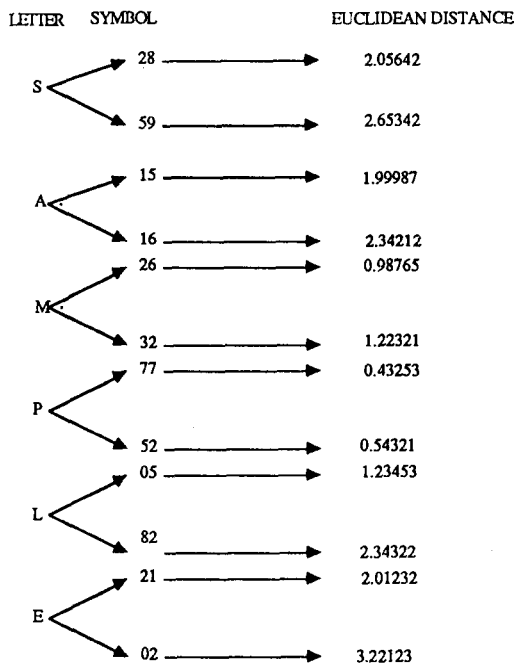
Define

$P(\underline{X})$ = Probability of State Sequence
$\underline{X} = X_1, X_2, \ldots, X_T$

$P(\underline{X}, \underline{O})$ = Joint Probability of State Sequence
$\underline{X} = X_1, X_2, \ldots, X_T$

RESULTS FROM ADDITIONAL TESTS CARRIED OUT ON UNRECOGNIZED WORD:

WORD BEING CONSIDERED: SAMPLE
1st SET OF SYMBOLS : 28 15 26 77 05 21
2nd SET OF SYMBOLS : 59 16 32 52 82 02



LETTER	SYMBOLS	SELECT
S	28, 59	28, 59
A	15, 16	15, 16
M	26, 32	26, 32
P	77, 52	77, 52
L	05, 82	05
E	21, 02	21

No.	Combinations	1st Choice	Probability	2nd Choice	Probability	Select
1	28 15 26 77 5 21	KAMPLE	9.96×10^{-12}	KAMPLA	2.22×10^{-14}	
2	28 15 26 52 5 21	KAMBLE	7.02×10^{-12}	KAMBLA	5.79×10^{-15}	
3	28 15 32 77 5 21	KABBLE	1.38×10^{-11}	KABBLA	6.84×10^{-15}	
4	28 15 32 52 5 21	KABBLE	2.41×10^{-11}	KABBLA	4.96×10^{-15}	
5	28 16 26 77 5 21	KAMPLE	1.66×10^{-12}	KAMPLA	3.71×10^{-15}	
6	28 16 26 52 5 21	KAMBLE	1.17×10^{-12}	KAMBLA	9.67×10^{-16}	
7	28 16 32 77 5 21	KABBLE	2.30×10^{-12}	KABBLA	1.14×10^{-15}	
8	28 16 32 52 5 21	KABBLE	4.02×10^{-12}	KABBLA	8.27×10^{-16}	
9	59 15 26 77 5 21	SAMPLE *	1.20×10^{-10} *	SAMPLA	2.67×10^{-13}	SAMPLE
10	59 15 26 52 5 21	SAMBLE	8.45×10^{-11}	SAMBLA	6.97×10^{-14}	
11	59 15 32 77 5 21	SAMPLE	3.69×10^{-11}	SAMPLA	8.23×10^{-14}	
12	59 15 32 52 5 21	SABBLE	5.06×10^{-11}	SABBLA	2.15×10^{-14}	
13	59 16 26 77 5 21	SAMPLE	2.00×10^{-11}	SAMPLA	4.46×10^{-14}	
14	59 16 26 52 5 21	SAMBLE	1.41×10^{-11}	SAMBLA	1.16×10^{-14}	
15	59 16 32 77 5 21	SAMPLE	6.16×10^{-12}	SAMPLA	1.37×10^{-14}	
16	59 16 32 52 5 21	SABBLE	8.44×10^{-12}	SABBLA	3.58×10^{-15}	

Fig. 8(a). Hypothesis generation and recognition scheme (HGRS) as applied to the input handwritten word "sample".

No.	Word	Probability	Selection	Liklihood
1	SAMPLE	1.20×10^{-10}	SAMPLE	1
2	CHAIR	$1.11 \times 10^{-9}/3.25 \times 10^{-11}$	CHANT/CHAIR	0.73/0.27
3	KEYS	2.59×10^{-7}	KEIS	1

Fig. 8(b). Results of running HGRS on unrecognized words of Fig. 7. Likelihood for ith hypothesized word $= 0.5/N + 0.5(P_i)/\sum_{i=1}^{N} P_i$; P_i = probability calculated by VA for ith hypothesized word; N = total no. of hypothesized words.

and Observation Sequence $\underline{O} = O_1, O_2, \ldots, O_T$

$P(\underline{O} \mid \underline{X})$ = Probability of Observation Sequence $\underline{O} = O_1, \ldots, O_T$ given State Sequence $\underline{X} = X_1, \ldots, X_T$.

$$\text{Then, } P(\underline{X}, \underline{O}) = P(\underline{X})P(\underline{O} \mid \underline{X}) \tag{17}$$

$$= P(X_1)P(O_1 \mid X_1)P(X_2 \mid X_1)P(O_2 \mid X_2) \prod_{i=3}^{i=T} P(X_i \mid X_{i-1}X_{i-2})P(O_i \mid X_i)$$

where T is the length of the observation sequence. The new sequence Y_i is defined as

$$Y_1 = X_1; \qquad Y_i = X_{i-1}X_i \tag{18}$$

For example, for the word "seems" $\underline{X} = (X_1, X_2, X_3, X_4, X_5) = (s, e, e, m, s)$; and the corresponding Y states are given by $\underline{Y} = (Y_1, Y_2, Y_3, Y_4, Y_5) = (s, se, ee, em, ms)$. With the definition as given by equation (18), we could write

$$P(Y_1) = P(X_1) \tag{19a}$$

$$P(Y_2 \mid Y_1) = P(X_2 \mid X_1). \tag{19b}$$

$$P(Y_i \mid Y_{i-1}) = P(X_i \mid X_{i-1}X_{i-2}); \qquad 3 \le i \le T. \tag{19c}$$

With these definitions, equation (17) could be rewritten as

$$P(\underline{Y}, \underline{O}) = P(Y_1)P(O_1 \mid Y_1)P(Y_2 \mid Y_1)P(O_2 \mid Y_2) \sum_{i=3}^{i=T} P(Y_i \mid Y_{i-1})P(O_i \mid Y_i). \tag{20}$$

Now, the state sequence $\underline{Y}$ that maximizes the probability on the left-hand side of equation (20) is given by the 'Viterbi algorithm' for first order model as described by equations (14–16). In the following, the 'Viterbi algorithm' for the second order model is described in detail. Define

$$a0(l) = P(X_1 = l); \tag{21a}$$

$$a1(l, m) = P(X_1 = l, X_2 = m \mid X_1 = l); \tag{21b}$$

$$a2(l, m, n) = P(X_i = n, X_{i-1} = m \mid X_{i-1} = m, X_{i-2} = l) \tag{21c}$$

$$b(O_i \mid n) = P(O_i \mid X_i = n); \tag{21d}$$

$$d_1(l) = P(X_1 = l, O_1); \tag{21e}$$

$d_i(m, n)$ = Maximum for

$$P(X_1, \ldots, X_{i-1} = m, X_i = n, O_1, \ldots, O_i); \; i \ge 3 \tag{21f}$$

$c_i(m, n)$ = state of X_{i-2} that makes

$$P(X_1, \ldots, X_{i-1} = m, X_i = n, O_1, \ldots, O_i) \text{ maximum} \tag{21g}$$

$X(i)$ = State at time i in the optimal sequence of $\underline{X}$. (21h)

Step 1: intialization. For $1 \le l \le N$

$$d_1(l) = a0(l)b(O_1 \mid l). \tag{22a}$$

For $1 \le l \le N$ and $1 \le m \le N$

$$d_2(l, m) = d_1(l)a1(l, m)b(O_2 \mid m). \tag{22b}$$

Step 2: recursive computation. For $3 \le i \le T$ and $1 \le n \le N$ and $1 \le m \le N$

$$d_i(m, n) = \operatorname*{Max}_{1 \le l \le N} (d_{i-1}(l, m)a2(l, m, n))b(O_i \mid n) \tag{23a}$$

$$c_i(m, n) = \operatorname{Arg} \operatorname*{Max}_{1 \le l \le N} (d_{i-1}(l, m)a2(l, m, n)). \tag{23b}$$

Step 3: terminal states.

$$X(T) = \operatorname*{Arg}_{n} \operatorname*{Max}_{1 \le m, n \le N} d_T(m, n) \tag{24a}$$

$$X(T-1) = \operatorname*{Arg}_{m} \operatorname*{Max}_{1 \le m, n \le N} d_T(m, n). \tag{24b}$$

Step 4: backtracking. For $T - 2 \ge i \ge 1$

$$X(i) = c_{i+2}(X(i+1), X(i+2)). \tag{25}$$

Optimal state sequence is now stored in $X(i)$, $1 \le i \le T$.

Hypothesis generation scheme with the second order model is same as with the first order model.

7. EXPERIMENTAL RESULTS AND COMPARISON

In our experiment, during the learning phase, 2500 letters are used. One hundred different writers of different sexes and ages with very diverse ethnic background were given to write all 26 different letters of the English alphabet. As a result, we started with 2600 different letters. Out of these, approximately 100 letters were later rejected as they were even difficult for humans to recognize. This does not mean that our data

is too constrained. On the contrary, our data is, to a large extent, free of constraints. The writers were only instructed not to write too big or too small; or to put too much style into it. Figure 9 shows some typical data used in our experiment. Even a casual glimpse at these data clearly brings out the large variation in writing style embedded in our data.

Using 2500 letters, a 90-symbol strong VQ codebook is generated; and the symbol probability distribution is calculated. Other model probabilities are calculated using the procedure discussed in Section 5. During the recognition phase, tens of words by different writers are used to test the recognition system. In the following, we describe one experiment which is typical of our experimental repertoire. We took 20 different writers, half of them not included during the phase of symbol generation (learning phase). Each writer wrote one word different from others. Figure 7 shows the recognition results. Observe that some of the words not recognized by the first order model are recognized by the second order model. Also, the probabilities calculated by 'Viterbi algorithm' is generally higher for the second order model indicating greater selectivity. However, such words as 'bye', though correctly recognized by the first order model, are not recognized correctly by the second order model. The reason is that for small word length (typically 3 or less), the second order model is not very meaningful. The combined result is 17 correct recognitions out of 20—a good 85% accuracy. The unrecognized words (3 of them) are then further analyzed using hypothesis generation and recognition scheme (HGRS). Figure 8(a) graphically depicts the scheme for the word sample. Each segmented letter of 'sample' is mapped to two nearest neighbor symbol, and then the symbols are combined as depicted in Fig. 8(a). Among all the hypothesized words, those with high probability, in relative sense, are selected; and then matched with the dictionary. This leads to correct recognition of the word 'sample'. Figure 8(b) shows the result of HGRS. Very interestingly, the input word chair leads to two highly probable and linguistically correct words chant and chair. The subsequent use of context, semantics etc. could resolve the uncertainty. Finally, the word keys is not recognized at all. But the most likely hypothesis keis is very very close.

In brief, the new scheme has an accuracy of 92.5%. On the basis of experimental results using tens of words, this figure of accuracy seems to be very realistic. Next some comparison results are given indicating the comparative success of the new scheme.

COMPARISONS AND CONCLUSIONS

The results described above are based on somewhat limited simulation experiments given the nature of the problem. Nevertheless, it is reasonable to claim that more data will lead to better symbol definition which is the biggest source of error. Better symbol definition, in turn, will give better percentage accuracy for recognition. Thus, a 92.5% claim of recognition accuracy at this stage of the experiment, is eminently justified. This result is even better than the best result obtained with highly constrained character recognition scheme extrapolated to word recognition.

An algorithm for handwritten word recognition should not be evaluated only on its recognition accuracy performance alone. Because, the unrecognized words are important too. A good recognition algorithm should be able to provide a lot of information for the unrecognized words such that any further information could help the recognition process. In

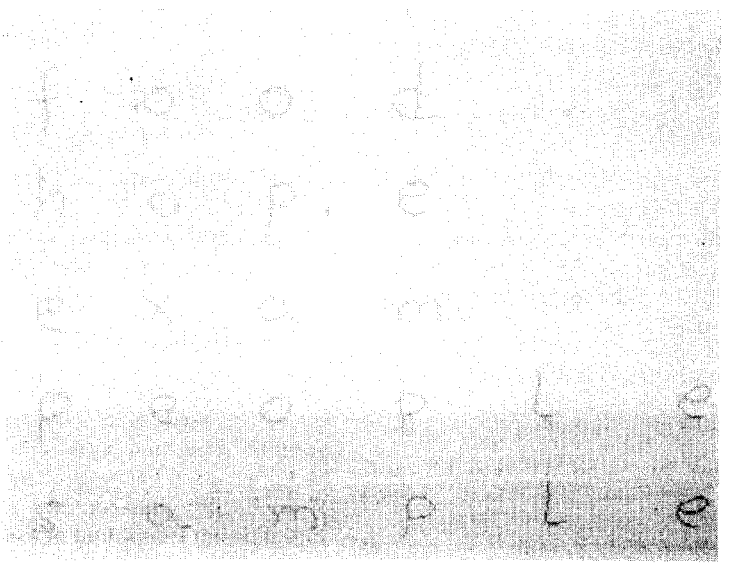


Fig. 9. Typical data used in our experiment.

this area, our algorithm scores really high. Consider Fig. 8(b) which lists all the plausible hypotheses for the handwritten input word chair with their relative likelihood function (sum of the likelihood functions is normalized to unity). For the input handwritten word sample, the hypothesis 'sample' stands out with respect to other hypothesis 'kample' which is also not linguistically correct (Fig. 8(a)). Finally, for the input word 'keys' the most probable hypothesis 'keis' is very very close. Any knowledge of syntax, semantics and context could recover the correct word rather easily.

How do we compare our results with that of other researchers? Surprising it may seem, there are not many word recognition scheme for handwritten script while there are many on handwritten character recognition. Since the word is made of segmented characters, this viewpoint assumes, correctly, that high accuracy in character recognition would lead to high accuracy in word recognition. This is rightly so. But the words are made up of many characters; and error in recognizing a single character leads to erroneous recognition of the entire word. On the basis of a random experiment, using 100 randomly chosen words from a daily newspaper, we have computed that the average length of commonly used English words is 4.5. The best possible character recognition results using rather unconstrained data like ours is 95% as reported in literature.[8,12,14] For an average wordlength of 4.5, this translates to a word recognition accuracy of 77.5%, compared to ours 92.5%. For very constrained data, the character recognition accuracy is 98% as reported in literature.[8,16] However high it may seem, this leads to an accuracy of 91% for an average wordlength of 4.5—a good 1.5% point less than our scheme. This result is not surprising given the fact that the HMM based approach incorporates a statistical model of successive character generation in a word; and consequently uses more information during the recognition process. Figure 10 gives the comparison results in a succinct tabular form.

For the unrecognized words, HMM based approach posits an even bigger advantage. It can come up with a number of 'hypothesized words' with a quantitative likelihood function to aid the future selection process. It is possible to generate word hypothesis with the methods based on individual character recognition, but not to the level of sophistication of HMM based approach.

HMM based approach, understandably, has more complexity than character based pattern recognition approach. During the training phase, the complexity for HMM based approach is dominated by feature selection and symbol generation. For character based approach, it would be feature selection and representative pattern generation. Thus, the complexities of training phase are approximately the same for both the scheme. During recognition phase, HMM based approach has the additional complexity of using 'Viterbi algorithm'. This complexity is rather negli-

	SCHEME	TYPE OF DATA	WORD RECOGNITION ACCURACY	TREATMENT OF UNRECOGNIZED WORDS	COMPLEXITY
1.	Ours Kundu et al. [24 , 25]	Unconstrained	92.5 %	Very sophisticated. A no. of hypothesized words with numerical likelihood function for unrecognized word.	Training phase complexity is approximately equal to that of (2) & (3). Recognition phase complexity is slightly higher.
2.	As reported in [8,12,14,32]	Unconstrained	79 % *	Some hypotheses possible for unrecognized words but not to the level of sophistication of (1)	Complexity of a typical pattern recognition algorithm. Complexity dominated by feature selection and extraction, pattern generation and matching.
3.	As reported in [8,33,16]	Constrained	91 % *	.....not to the level of sophistication as in (1)	same as in (2)

* Based on an average word length of 4.5 letters/word .

Fig. 10. Comparison results between the proposed scheme and some well-known techniques with high recognition rate. Proposed scheme clearly comes out on top.

gible—to generate 16 hypotheses for the input word 'sample' approximately 0.3 s of CPU time is needed.

In conclusion, we comment that for handwritten word recognition, the HMM based approach is clearly superior to any character recognition based approach. It is pertinent to mention here that there are some other word recognition schemes[4,23] reported in literature for handwritten script. But these schemes are very limited in scope as they deal with very limited vocabulary. As a result, these schemes are not directly comparable to our scheme.

Although the performance of the algorithm is excellent, it could be improved further at least in three respects (1) The symbols are defined using fifteen features. Use of few more features or better features is likely to improve the clustering of symbols. Hence, the recognition results. (2) More training data will lead to statistically superior symbol definition. (3) Numerical digits could be added to the alphabet.

Extension to numerics

Once the numbers are included in the handwritten script, there are, theoretically speaking, 36 states instead of 26 states. But most importantly, the state transition probabilities from number to number, or from number to letter are very different. By probabilistic argument, one could infer that the state transition probability from one number to another number is equally likely (1/10, when only the numbers are considered). In written English, the numbers often appear as one word though such words as 3rd, 12th etc. are not uncommon. Consequently, we need to build a separate HMM for the numbers. Building one HMM for both numbers and alphabets will be very complex in terms of increased number of states and the inapplicability of the state transition probabilities form one letter to another as given in Figs 3 and 6. Using two HMM, each word to be 'recognized' will be separately scored[30] against both the models. The bigger score determines the nature of the word as well as the word itself. For the recognition of such words as 12th, 3rd etc., a third model has to be designed.

Investigations are underway in these directions; and will be reported in due time.

SUMMARY

A hidden Markov model (HMM) is a doubly stochastic process with an underlying stochastic process that is not observable, i.e. states are hidden, but can only be observed through another set of stochastic processes that produce the observable sequence of symbols. In this work, the handwritten script recognition problem is modeled in the framework of HMM. For English text, which is the focus of the present research, the states could be identified with the letters of the alphabet, and the optimum symbols could be generated by means of experimental study. In order to do so, a quantitative definition of symbols, in terms of features, is required. Fifteen features (some old, some new) are used for this task.

Both the first and second order hidden Markov models are proposed for the recognition task. Using the existing statistical knowledge of the English language as used in cryptography etc., the calculation scheme of the model parameters could be immensely simplified for the first order model. Extending these results and through an exhaustive dictionary search, probabilities of the second order model could be calculated. Once the model is established, the recognition algorithm, known as Viterbi algorithm, is used to recognize the single best optimal state sequence, i.e. sequence of letters consisting the word. The modification of the recognition algorithm to accommodate context information has also been discussed. Finally, some experimental results for the first and second order models are provided indicating the tremendous success potential of the proposed scheme.

REFERENCE

1. J. R. Ullmann, Advances in character recognition, *Applications of Pattern Recognition*, K. S. Fu, Ed., pp. 197–236. CRC Press, Boca Raton, FL (1982).
2. H. Genchi, K. I. Mori, S. Watanabe and S. Katsuragi, Recognition of handwritten numerals for automatic letter sorting, *Proc. IEEE* **56,** 1292–1301 (1968).
3. J. Schurmann, A multifont word recognition system for postal address reading, *IEEE Trans. Comput.* **C-27,** 721–732 (1978).
4. L. D. Harmon, Automatic recognition of print and script, *Proc. IEEE*, pp. 1165–1176 (1972).
5. C. Y. Suen and R. J. Shillman, Low error-rate optical character recognition of unconstrained handprinted letters based on a model of human perception, *IEEE Trans. SMC*, 491–495 (1977).
6. T. Pavlidis and F. Ali, Computer recognition of handwritten numerals by polygonal approximations, *IEEE Trans. SMC* **5,** 601–614 (1975).
7. M. J. Minneman, Handwritten character recognition employing topology, cross correlation and decision theory, *IEEE Trans. Syst. Sci. Cybernetics* **2,** 86–96 (1966).
8. C. Y. Suen, M. Berthod and S. Mori, Automatic recognition of handprinted character: the state of the art, *Proc. IEEE* **68,** 469–487 (1980).
9. E. Persoon and K. S. Fu, Shape discrimination using Fourier descriptors, *IEEE Trans. SMC* **7,** 170–179 (1977).
10. S. Wendling and G. Stamon, Hadamard and Harr transform and their power spectra in character recognition, *Proc. Joint Workshop Pattern Recognition AI*, pp. 103–112 (1976).
11. N. D. Tucker and F. C. Evans, A 2-step strategy for character recognition using geometrical moments, *Proc. 2nd Int. Joint Conf. Pattern Recognition*, pp. 223–225 (1974).
12. M. Kushnir, K. Abe and K. Matsumoto, Recognition of handprinted Hebrew characters using features selected in Hough transform space, *Pattern Recognition* **18,** 103–114 (1985).
13. A. L. Knoll, Experiments with characteristic loci for recognition of handprinted characters, *IEEE Trans. Comput.* **C-18,** 366–372 (1969).
14. C. C. Tappert, Adaptive on-line handwriting recognition, *Proc. Int. Conf. Pattern Recognition*, pp. 1004–1007 (1984).

15. J. T. Tou and R. C. Gonzalez, Recognition of handwritten characters by topological feature extraction and multilevel categorization, *IEEE Trans. Comput.* **C-21,** 776–785 (1972).
16. K. Yamamoto and S. Mori, Recognition of handprinted characters by outermost point method, *Pattern Recognition* **12,** 229–236 (1980).
17. S. Mori, K. Yamamoto and M. Yasuda, Research on machine recognition of handprinted characters, *IEEE Trans PAMI* **PAMI-6,** 386–403 (1984).
18. J. K. Cattell, The time it takes to see and name objects, *Mind* **11,** 63–65 (1886).
19. J. J. Hull and S. N. Srihari, A computational approach to visual word recognition: hypothesis generation and testing, *Proc. IEEE Comput. Soc. Conf. Comput Vision Pattern Recognition*, Miami Beach, FL (1986).
20. R. Shingal and G. T. Toussaint, A bottom-up and top-down approach to using context in text recognition, *Int. J. Man Mach. Stud.* 201–212 (1979).
21. J. Schuermann and W. Doster, A decision theoretic approach to hierarchical classifier design, *Pattern Recognition* **17,** 359–369 (1984).
22. Y. L. Ma, The pattern recognition of Chinese characters by Markov chain procedure, *IEEE Trans SMC* 223–228 (1974).
23. R. Nag, K. H. Wong and F. Fallside, Script recognition using hidden Markov models, *Proc. ICASSP*, pp. 2071–2074 (1986).
24. A. Kundu and P. Bahl, Recognition of handwritten script: a hidden Markov model based approach, *Proc. ICASSP*, New York City, pp. 928–931 (1988).
25. A. Kundu, Yang He and P. Bahl, Word recognition and word hypothesis generation for handwritten script: a hidden Markov model based approach, *Proc. CVPR*, Ann Arbor, Michigan, pp. 457–462 (1988).
26. S. N. Srihari, J. J. Hull and R. Chowdhury, Integrating diverse knowledge sources in text recognition, *ACM Trans. Off. Automn Syst*, 68–87 (1983).
27. L. R. Rabiner and B. H. Juang, An introduction to hidden Markov model, *ASSP* Mag. **3,** 4–16 (1986).
28. A. B. Poritz, Linear predictive hidden Markov model and the speech signal, *Proc. ICASSP* '82, Paris, France, pp. 1291–1294 (1982).
29. A. Ljolje and F. Fallside, Synthesis of natural sounding pitch contours in isolated utterances using hidden Markov models, *IEEE Trans ASSP*, **34,** 1074–1080 (1986).
30. L. R. Rabiner, S. E. Levinson and M. M. Sondhi, On the application of vector quantization and hidden Markov models to speaker independent isolated word recognition, *BSTJ*, 1075–1105 (1983).
31. S. A. Dudani, K. J. Breeding and R. B. Mcghee, Aircraft identification by moment invariants, *IEEE Trans. Comput.* **C-26,** 39–45 (1977).
32. P. Siy and C. S. Chen, Fuzzy logic for handwritten numeral character recognition, *IEEE Trans. SMC*, 570–574 (1974).
33. J.S. Huang and K. Chuang, Heuristic approach to handwritten numeral recognition, *Pattern Recognition* **19,** 15–19 (1986).
34. T. Pavlidis, *Algorithm for Graphics and Image Processing*. Computer Science, Rockville, MD (1982).
35. R. M. Gray, Vector quantization, *IEEE ASSP Mag.* **1,** 4–29 (1984).
36. A. G. Konheim, *Cryptography: A Primer*. John Wiley, New York (1982).

About the Author—AMLAN KUNDU was born in Calcutta on 8 April 1955. He received his B.Tech. in Electronics from Calcutta University, India in 1979; M.S. and Ph.D. in Electrical Engineering from UC Santa Barbara in 1982 and 1985 respectively. Since fall 1984, he is an Assistant Professor of Electrical Engineering at the State University of New York at Buffalo. His research interest spans various aspects of Image Processing, Robust Image Smoothing, Pattern Recognition and VLSI implementation of Signal Processing Algorithms.

About the Author—YANG HE was born in Beijing, China on 6 August 1955. He received B.S. degree from Huazhong University of Science and Technology, Wuhan, China in 1982 and M.S. Degree from Quinghua University, Beijing, China in 1984.

From 1984 to 1978, he was with the Department of Electronics Engineering, Shenzhuen University, China, where he served as a lecturer. He is currently a visiting scholar in the Department of Electrical and Computer Engineering at the State University of New York at Buffalo. His research areas include Image Processing, Signal Processing, Pattern Recognition and Computer Vision.

About the Author—PARAMVIR BAHL was born in New Delhi, India on 23 September 1964. He received his B.S. and M.S. in 1986 and 1988, respectively, in Electrical and Computer Engineering from the State University of New York at Buffalo. Currently, he is working in the Research and Advanced Development, Image Systems Group in Digital Equipment Corp.

Bahl's current research interests include Image Processing, Signal Processing, Computer Vision and Software Engineering. He is the President of Eta Kappa Nu, Zeta Pi chapter and is a member of Tau Beta Pi.