Generalized Hidden Markov Models—Part II: Application to Handwritten Word Recognition

Magdi A. Mohamed, *Member, IEEE*, and Paul Gader, *Senior Member, IEEE*

***Abstract*—This is the second paper in a series of two papers describing a novel approach for generalizing classical hidden Markov models using fuzzy measures and fuzzy integrals and their application to the problem of handwritten word recognition. This paper presents an application of the generalized hidden Markov models to handwritten word recognition. The system represents a word image as an ordered list of observation vectors by encoding features computed from each column in the given word image. Word models are formed by concatenating the state chains of the constituent character hidden Markov models. The novel work presented includes the preprocessing, feature extraction, and the application of the generalized hidden Markov models to handwritten word recognition. Methods for training the classical and generalized (fuzzy) models are described. Experiments were performed on a standard data set of handwritten word images obtained from the U. S. Post Office mail stream, which contains real-word samples of different styles and qualities.**

***Index Terms*—Fuzzy integral, fuzzy measures, handwriting recognition, Markov models.**

I. Introduction

AUTOMATED recognition of handwritten words is a central concern in the implementation of a reading machine for many practical applications. The handwritten word recognition task is characterized by high data rates, large amounts of data, possibly error-filled input, and the need for real-time response. Indeed, the magnitude and complexity of the task is such that it must be subdivided into subtasks and yet the applications illustrate the close interconnection of these subtasks and their central importance in accomplishing this extremely valuable task efficiently.

In most cases, there is uncertainty due to the incomplete, imprecise, or ambiguous contents of the handwritten word images. Performing handwritten word recognition becomes, in a sense, a search for a way to capture the organization of the complexity and to deal with a high degree of ambiguity. In view of these considerations, the production of optimally cost-effective handwritten word recognition system design is a challenging task for both practitioners and researchers.

In the following sections, we will discuss some of the issues and considerations that will determine the workings of a handwritten word recognition system and state the specific problem addressed in this study.

Manuscript received June 14, 1996; revised August 10, 1999.

The authors are with the Department of Electrical and Computer Engineering, University of Missouri-Columbia, Columbia, MO 65211 USA.

Publisher Item Identifier S 1063-6706(00)01624-6.

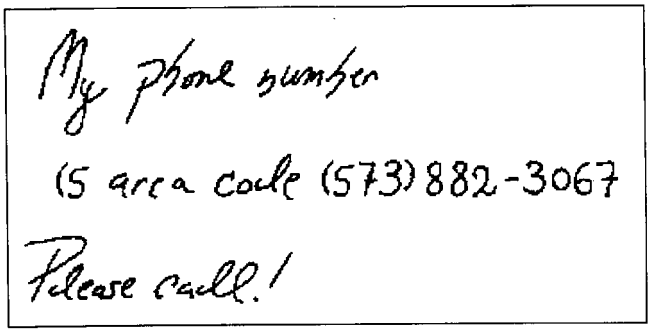


Fig. 1. Ambiguous patterns in handwritten words.

A. The Difficulty of the Problem

Automatic reading of handwritten words is a very rich and interesting problem, involving a broad range of questions. It is a difficult problem, not only because of the great variety in the shape of characters, but also because of the overlapping and the interconnection of the neighboring characters. In handwriting we may observe either isolated letters such as handprinted characters, groups of connected letters, i.e., subwords, or entirely connected words. Furthermore, when observed in isolation, characters are often ambiguous and require context to minimize the classification errors. For example, we may probably read the message shown in Fig. 1 as, "My phone number is area code (573) 882-3067. Please call!" because of the effect of the context of the message. If we look carefully, we will notice that the word "is" and the part of the area code number "(5" are written identically. We will also note that the "h" in the word "phone" and "b" in the word "number" are the same, as well as the "d" in the word "code" and the "l" in the word "please." Thus, characters from different classes can be written using identical shapes. Conversely, characters from the same class can be written very differently. For example, although this message is written by a single writer, many variations in size and shape of the character "a" are observed inside the words "area," "please," and "call." When the number of writers increases (as in handwritten words from address blocks), the degree of variation increases as shown in Fig. 2. Many design efforts for character recognition are available, based nominally on almost all types of classification methods such as neural nets, linear discriminant functions, fuzzy logic, template matching, binary comparisons, genetic algorithms, etc. [1]–[8]. The existing development efforts to solve such problems have involved long evolutions of differing classification algorithms, resulting in a final design that is always an engineering combination of many techniques, not a single technique [1], [9]–[14].

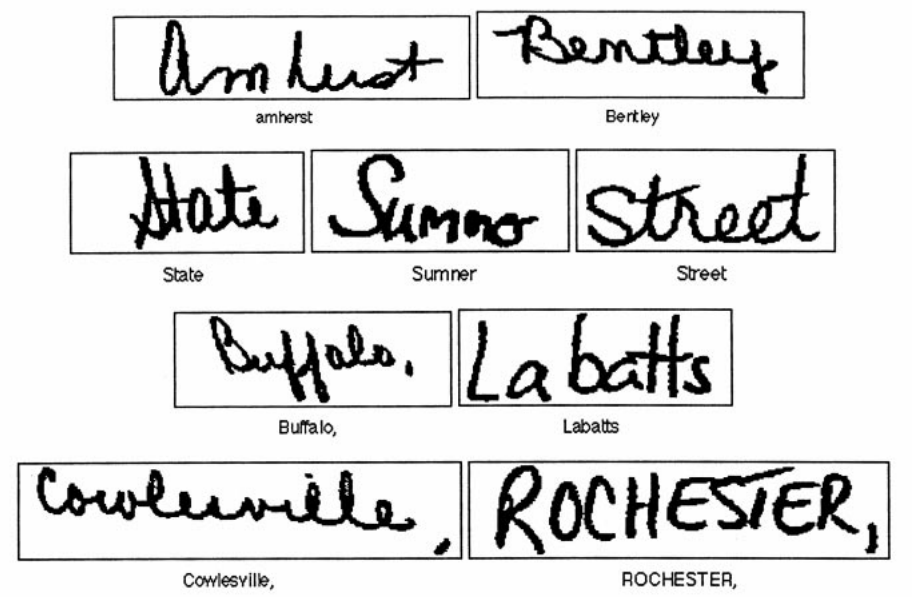


Fig. 2. Handwritten words from address blocks.

B. HMM in Handwritten Word Recognition

Hidden Markov models (HMM's) have been applied to handwritten word recognition by many researchers during the last decade. Chen *et al.* [15] used discrete HMM to recognize words using lexicons. The input word image is first segmented into a sequence of segments in which an individual segment may be a complete character, a partial character, or joint characters. The HMM parameters are estimated from the lexicon and the training image segments. A modified Viterbi algorithm is used to find the best L state sequences. One HMM is constructed for the whole language and the optimal L paths are utilized to perform classification. When tested on a set of 98 words, this system is able to achieve 72.3% success for a lexicon of size 271 using 35 features. Another application of HMM in a character segmentation-based word recognition was presented by Chen *et al.* [16]. Chen used continuous density, variable duration hidden Markov model (CDVDHMM) to build character models. The character models are left-to-right HMM's in which it is possible to skip any number of states. A major disadvantage of this technique is that it is slow to train and in operation because of introducing more parameters and computation for the state duration statistics.

Another interesting technique was developed by Gillies [9] for cursive word recognition using left-to-right discrete hidden Markov models. In this technique, each column of the word binary image is represented as a feature vector. A series of morphological operations are used to label each pixel in the image according to its location in strokes, holes, and concavities located above, within and below the core region. A separate model is trained for each individual character using word images where the character boundaries have been identified. The word matching process uses models for each word in a supplied lexicon. The advantage of this technique is that the word need not be segmented into characters for the matching process. When tested on a set of 269 cursive words this technique is able to achieve 72.6% success for a lexicon of size 100.

Performing word recognition without segmenting into characters is an attractive feature of a word recognition system since segmentation is ambiguous and prone to failure for a significant portion of the handwritten words coming from the mail stream. In an attempt to avoid some limitations of the existing segmentation-based handwritten word recognition techniques we designed a segmentation-free model-based system. In this technique, the training process does not need representative images of each word in a lexicon. Our proposed segmentation-free system computes features from each column in a word image and uses a continuous density HMM for each character class.

The continuous density HMM has a significant advantage over the discrete density HMM. Using a continuous density HMM is attractive in the sense that the observations are encoded as continuous signals. Although it is possible to quantize such continuous signals there might be a serious degradation associated with such quantization. In the case of the discrete HMM, a codebook must be constructed prior to training the models for any class. The codebook is constructed using vector quantization or clustering applied to the set of all observations from all classes. By contrast, in the continuous case, the clusters of observations are created for each model separately (e.g., by estimating Gaussian mixture parameters for each model). Thus, continuous density HMM's provide more flexibility for representing the different levels of complexity and feature attributes of the individual character classes. Furthermore, in the discrete case, training a new model may require creating a new codebook since the features required by the new model may be quite different from those represented in the old codebook. Since the continuous density models perform clustering independently for each class, there is no such requirement.

One computational difficulty associated with using continuous density HMM's is the inversion of the covariance matrices. We overcame this difficulty by reducing the dimensionality

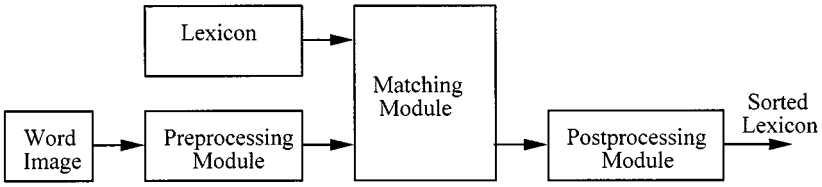


Fig. 3. Overview of a word recognition system.

using principal component analysis and approximating the densities as a mixture of Gaussian densities with diagonal covariance matrices. The results provided in this paper demonstrate that the approach performs reasonably well when confronted with various styles of handwritten words.

C. Off-Line Word Recognition Systems

The specific problem addressed in this study is the recognition of handwritten words extracted from postal addresses. Our concern is the recognition of isolated handwritten words using the contextual information provided through the associated lexicon that defines the vocabulary range.

A generic off-line word-recognition system has two inputs: a digital image, assumed to be an image of a word and a list of strings called a lexicon, representing possible identities for the word image. In general, before looking for features, some preprocessing techniques are applied to the word image to avoid recognition mistakes due to the processing of irrelevant data (see Fig. 3). The goal of the word recognition system is to assign a match score to each candidate in the lexicon. The match score assigned to a string in the lexicon represents the degree to which the image "looks like" the string. The output from this matching process is usually followed by a postprocessing step to check for highly unlikely decisions. Finally, a sorted lexicon is the output from the word recognition system.

The approach considered for our research is based on the use of the classical and generalized hidden Markov models. The first task accomplished in this research is the development and demonstration of a classical HMM handwritten word recognition system using novel image processing and feature extraction techniques. We used a segmentation-free continuous density hidden Markov modeling approach to improve the performance of the existing techniques in the literature. Our approach is the first to use continuous density hidden Markov models for a segmentation-free handwritten word recognition. The second task is the development and demonstration of a generalized (fuzzy) HMM system that uses fuzzy sets, fuzzy measures, and fuzzy integrals. In this paper, we describe our implementation for the classical and and generalized models in detail.

The remainder of this paper is organized to describe our research approach in the following manner. In Section II, a complete description of the implemented classical and fuzzy HMM off-line handwritten word recognition systems is provided. This section describes the feature extraction, dimensionality reduction, training and matching strategies. In Section III a complete description of the database is provided. We describe the database and the preprocessing steps we applied to the raw images contained in the standard training and testing sets. Section IV shows the experimental results and analysis of the performance of the implemented systems. Finally, Section V is dedicated to the summary of this study and the suggestions for future research.

II. Design and Implementation of Off-Line HMM Handwritten Word-Recognition Systems

In our handwritten word-recognition systems, an HMM is constructed for each character class using the training data. We used left-to-right HMM's with continuous probability densities to model the character classes. The approach is a segmentation-free technique. An HMM is constructed for each string in the lexicon by concatenating the state chains of the corresponding character models. Classification is performed according to the matching scores computed from the optimal state sequence using the Viterbi algorithm in the classical HMM and the fuzzy Viterbi algorithm in the generalized hidden Markov model (GHMM).

The inputs to the word recognition algorithm are a binary word image and a lexicon (see Fig. 4). After preprocessing the input binary word image, the resultant image is subjected to a feature extraction process. The output from this process is a sequence of observation vectors, each corresponds to an image column. Using the character models, a word model is constructed for each string inside the lexicon. The string matching process computes a matching score between the sequence of observation vectors and each word model using the Viterbi algorithm. After postprocessing, a lexicon sorted by the matching score is the final output of the word recognition system.

The following sections describe the system components and related implementation issues.

A. Feature Extraction

The performance of any classification or recognition algorithm depends, to a large extent, on the representation chosen, i.e., the features or primitives that are extracted from the inputs. These characteristics must, as far as possible, summarize the information, which is pertinent and useful for clustering and at the same time eliminate useless or irrelevant information, i.e., the randomness due to variability and nondiscriminant information.

The description of a binary word image as an ordered list of observation vectors $\{O_1\ O_2\ \cdots\ O_T\}$ is accomplished by encoding a combination of features computed from each column in the given preprocessed image. This representation uses what

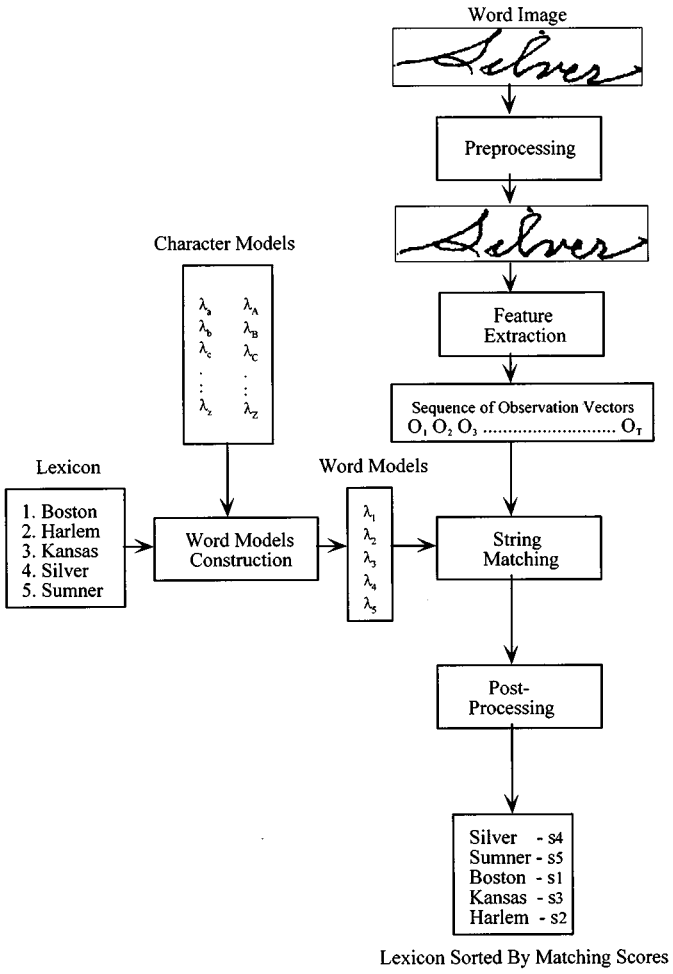


Fig. 4. Word recognition systems.

Fig. 5. Transition feature computations.

we refer to as the transition features. The idea behind the transition features is to compute the location and number of transitions from background to foreground pixels along vertical lines (image columns). This transition calculation is performed from top to bottom and from bottom to top (Fig. 5).

The first step of the feature computation is to estimate the center line of the word using the horizontal projection. The centroid of the horizontal projection of the word image is taken as an estimate for the word center-line location. The center line of the word is used to account for differences in the positions of characters within a word depending on whether a descender and/or ascender is present or not. For example, a lower case "a" may be at the bottom of a word image if there is no character of type descender in the word, but may be in the middle of the word if there is a descender and an ascender. The transition features are computed on a virtual bounding box of the word image. The core of the word image is considered to be at the middle of the virtual bounding box.

The location of each transition is represented as a fraction of the distance across the word image inside the virtual bounding box in the direction under consideration [7]. For example, when calculating the location of transitions from top to bottom, a transition close to the top edge would have a high value, whereas a transition far from the top edge would have a low value. A maximum number of transitions $MXNT$ are counted along each column. If there are more than $MXNT$ transitions in a given column, then only the first $MXNT$ transitions are considered and the rest are ignored. Currently, the parameter $MXNT$ is set to four. If there are less than $MXNT$ transitions on a column, then the "nonexistent" transitions are assigned a value of zero. Two more features representing the gradient of the north and south contours are included resulting in a total of ten features per each image column.

We used principal component analysis to reduce the dimensionality from ten to five when using both transition and gradient features. Fig. 6 shows the singular values for the covariance matrix of all character samples used for training the system.

When computing features for training the character modules the observation sequences are resampled to a fixed length $T = 24$. After the transition feature vectors are calculated for each column in the character image, they are resampled using linear interpolation. The rationale behind this resampling process is to guard againest biasing the training in favor of large characters. This is because each character image will be represented by a sequence of only 24 feature vectors in the training process regardless of the actual number of columns in the character image.

B. Character Models

For the purpose of isolated handwritten word recognition, it is useful to consider left-to-right models. In a left-to-right model transition from state i to state j is only allowed if $j \geq i$, resulting in a smaller number of transition probabilities to be learned. The left-to-right HMM has the desired property that it can readily model signals whose properties change over time. It is useful for building word models from character models because fewer between-character transition probabilities are required for constructing the word models from the corresponding character models.

The first problem one faces is deciding what the states in each character class correspond to, and then deciding how many states should be in each model. We optimized the structure for the character hidden Markov models by searching for the best number of states to be fixed among all character classes. We allow skipping only one state such that if a character class needs fewer states some of the skipping probabilities could be relatively high. Fig. 7 illustrates the structure of a character model assuming five states.

C. Training The Classical Character HMM's

A continuous density HMM is trained for each upper and lower case character class using the transition and gradient

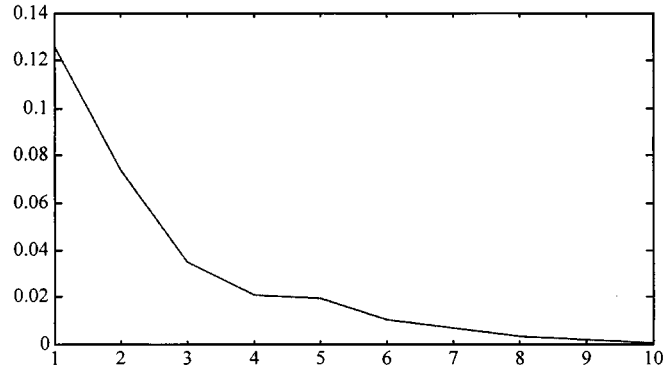


Fig. 6. Singular values for the training covariance matrix of transition and gradient feature vectors.

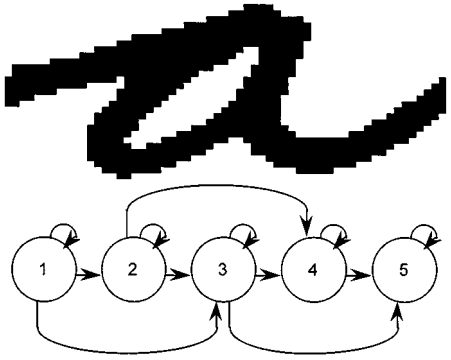


Fig. 7. A character model assuming five states.

features computed inside word images. We varied the number of states (from $N = 5$ to $N = 12$) for each model and the number of Gaussian mixtures per state (from $M = 1$ to $M = 3$). Initially, the mean and covariance matrix for each state mixture were estimated after dividing the resampled training transition vectors equally between the states.

The standard reestimation formulae were used inside the segmental K-mean algorithm assuming diagonal covariance matrices for the Gaussian mixtures. The Viterbi algorithm was used to find the optimal state sequence and assign each of the observation vectors to one of the states. The probability of the observation vectors given the model was used to check whether the new model is better than the old model inside the training procedure. The training algorithm proceeds in the following manner:

```
Algorithm: Clasical Segmental K-Means
  Choose initial values for model parameters;
  Refine initial values using standard reestimation formulae;
  Set Improving=True;
  WHILE (Improving=True)
  DO
    Segment training sequences into N states (Usig Viterbi);
    Cluster vectors in each state into M clusters;
    Update model parameters from the resulting states and clusters;
    Refine model parameters using standard reestimation formulas;
    IF the new model is not better than the old model set Improving=False;
  END WHILE.
```

The character models were trained on the training word images included in the State University of New York (SUNY) database described in Section III. For each character class we used all the available training samples. For example, 3474 samples are used to model the "a" class while only three samples are used to model the "Q" class. The actual distribution of all character training sets that have been extracted from all word images in the training sets is shown in Fig. 8.

D. Training The Fuzzy Character HMM's

Initially, the mean and covariance matrix for each state mixture are estimated after dividing the resampled training feature vectors equally among the states. The fuzzy reestimation formulas are used inside a fuzzy version of the segmental K-means algorithm assuming diagonal covariance matrices in the following manner:

```
Algorithm: Fuzzy Segmental K-Means
  Choose initial values for model parameters;
  Refine initial values using the fuzzy reestimation formulae;
  Set Improving=True;
  WHILE (Improving=True)
  DO
    Segment training sequences into N states (Using Fuzzy Viterbi);
    Cluster vectors in each state into M clusters;
    Update model parameters from the resulting states and clusters;
    Refine model parameters using the fuzzy reestimation formulas;
    IF the new model is not better than the old model, set Improving=False;
  END WHILE.
```

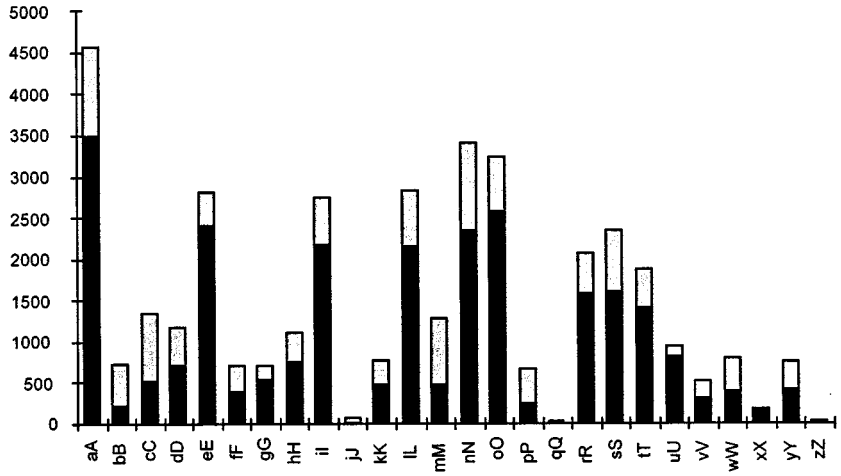


Fig. 8. Distribution of training samples.

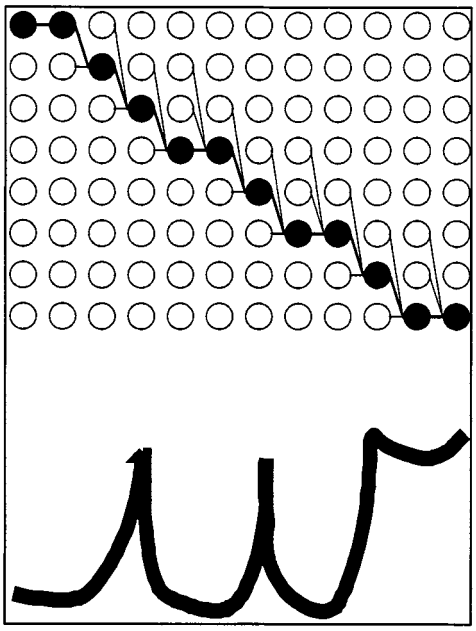


Fig. 9. Fuzzy states membership estimation.

After clustering the vectors in each state, the model parameters are updated using weighted mean and weighted covariance. The weights are determined from the fuzzy optimal state sequence and the fuzzy forward and backward variables in the following manner:

1) Compute the fuzzy forward and backward variables ($\hat{\alpha}_t(i)$ and $\hat{\beta}_t(i)$).
2) Find the optimal state sequence, $\{q_1 q_2 \cdots q_T\}$, using the fuzzy Viterbi.
3) Prune some of the paths by modifying the fuzzy forward variables as follows:

```
FOR t from 2 to T DO
  j = q_t
  FOR i from 1 to (j - 2) DO α̂_{t-1}(i) = 0 END
  FOR i from (j + 1) to T DO α̂_{t-1}(i) = 0 END
END.
```

4) Compute the weight of O_t in S_i called $\mu_i(O_t)$ by

$$\mu_i(O_t) = \frac{\hat{\alpha}_t(i)\hat{\beta}_t(i)}{\sum_{j=1}^{N} \hat{\alpha}_t(j)\hat{\beta}_t(j)}.$$

The rationale behind this pruning is to reduce the effect of the state sequences that are far from the estimated optimal state sequence (Fig. 9).

E. Lexicon Pruning

A lexicon pruning component is used to improve the recognition process by limiting the number of lexicon entries, especially when dealing with large size lexicons. Indeed, an exhaustive search through all the lexicon entries is likely to have side effects on the recognition robustness in addition to being very time consuming. Lexicon pruning is actually a global recognition process. Since the words to be recognized possess some discriminant features like ascenders, decsenders, loops, and overall length, this process usually builds a coarse description of the character segments inside words along a set of features. For now, our system uses the overall length of the word to avoid matching against some of the strings in the provided lexicons. The system checks for the average character width, if it is less than ten or greater than 160 columns a minimal value for the confidence is assigned to the corresponding lexicon string.

F. Matching Strategies and Post Processing

Since we have different models for upper and lower character classes, we match against each string in the lexicon using two strategies. The first strategy constructs a word model assuming that all characters in the word are upper case characters. The second strategy constructs a word model assuming that the first character is an upper case and the rest are lower case characters. Therefore, two scores are computed for each string in the lexicon using the Viterbi algorithm and the maximum is taken as the confidence for the given string.

TABLE I
NUMBER OF HANDWRITTEN WORD IMAGES IN THE TRAINING AND TESTING SETS

Number of training and testing data						
	training set			testing set		
subset	cities	states	ZIPs	cities	states	ZIPs
BB	363	277	269	37	31	30
BC	190	217	190	21	23	21
BD	3106	2490	2201	317	252	238
BL	877	1014	875	97	114	97
BS	564	467	450	60	50	49
BR	n.a.	n.a.	3962	n.a.	n.a.	n.a.
BU	n.a	n.a.	1072	n.a.	n.a.	n.a.
totals	5100	4465	9019	532	470	435

The segmentation of the word into characters is achieved as a byproduct of the Viterbi matching process for each string in the lexicon. To avoid inconsistent matches, we check for the character widths. If the number of columns for any character in the given string is less than three or greater than 164 the corresponding confidence is set to a minimal value. This is a very crude postprocessing technique. A more sophisticated technique that uses other consistency measures is required to avoid false matches.

III. DATABASE AND PREPROCESSING

A. The Database

The database used for our experiments consists of handwritten words and characters extracted from the first CDROM image database produced by the Center of excellence for document analysis and recognition (CEDAR) at the State University of New York (SUNY) at Buffalo. This database is widely available and serves as a benchmark for evaluation purposes [12]. We first describe the database and then the preprocessing steps we applied to the original images. The handwritten words (cities, states, and ZIP codes) are provided on the CDROM in full eight-bit gray scale. These data were divided into training and testing sets at SUNY by randomly choosing approximately 10% of the ZIP code images and placing those images as well as the city and state name from the corresponding addresses in the test set. The remainder of the data were retained for training purposes as shown in Table I.

Each city or state word in the test data is provided with three lexicons that simulate the results of recognition of the corresponding ZIP code. The lexicons contain lists of all the city or state words that could match the ZIP code that corresponds to those words when one, two, or three of the digits (randomly chosen) are allowed to vary. The data from the USPS database of cities, states, and ZIP codes were used to determine the lexicons. This file is also included in the database.

B. Preprocessing

The following preprocessing steps have been implemented, fine tuned, and applied to all word images in the database: binarization, line removal, border cleaning, tilt correction, slant correction, smoothing and scaling. Binarization was performed using Otsu's thresholding medthod [17]. We describe the other steps in detail in the following sections.

1) Line Removal and Border Cleaning: The line-removal algorithm checks for the existence of underlines and/or lines above a given word image using a binary morphological opening operation by a horizontal bar. The size of the horizontal bar (structuring element) varies with the width of the word image. The detected lines are removed from the given image. Since this process might remove some parts from the word image, a morphological conditional dilation operation is used to recover these parts. Formally, the line removal algorithm proceeds as follows:

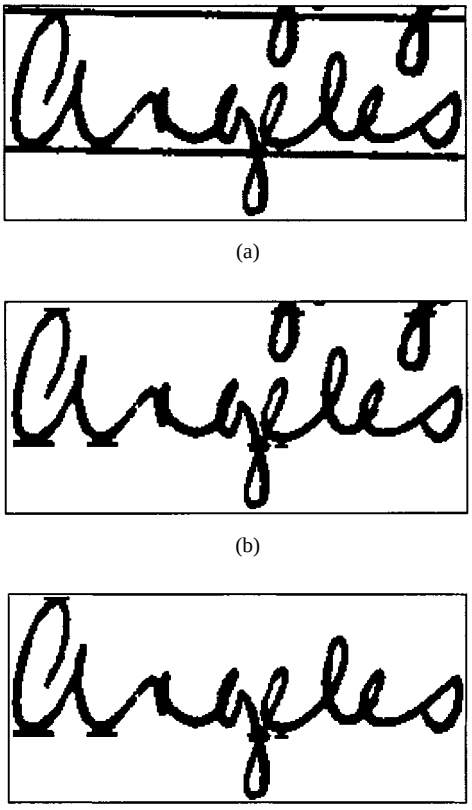
I a word binary image
H a horizontal bar structuring element of width that varies linearly with the width of the word binary image
s an estimate of the stroke width of the word binary image
D a disk structuring element of radius that varies linearly with s
$M = I \circ H$ the result of opening I by H
$T_1 = (M \oplus D) \cap I$
$T_2 = (I - T_1)$
$J_0 = T_2$
FOR m from 1 to s DO
$J_m = (J_{m-1} \oplus D) \cap I$
END.
$R = J_s$ the resultant binary image.

Word image borders are sometimes surrounded by character segments from adjacent fields that get captured during imaging. A similar technique to that used for line removal is applied here to check for connected components touching the borders of the image. All components having sizes below a specified size are eliminated from the word image. The border cleaning algorithm can be summarized as follows.

```
1) Find all connected components in the word
image.
2) Mark connected components that touch the
boundaries of the word image.
3) FOR each connected component that touches a
boundary.
   IF its size is less than a threshold value.
   THEN remove it from the word image.
END
```

The size of the connected components touching one or more of the image boundaries is considered to be the height or width of the bounding box surrounding the connected component depending on whether it touches a horizontal or vertical boundary respectively. The threshold varies linearly with the image height or width depending on whether the connected component touches a horizontal or vertical boundary respectively (see Fig. 10).

2) Tilt and Slant Correction: This is one of the most difficult preprocessing steps since it involves estimating angles to rotate and shear the original word image. It is also very important for the system since we use the columns of the images to



(a) (b) (c)

Fig. 10. Line removal and border cleaning. (a) Before line removal. (b) After line removal. (c) After border cleaning.

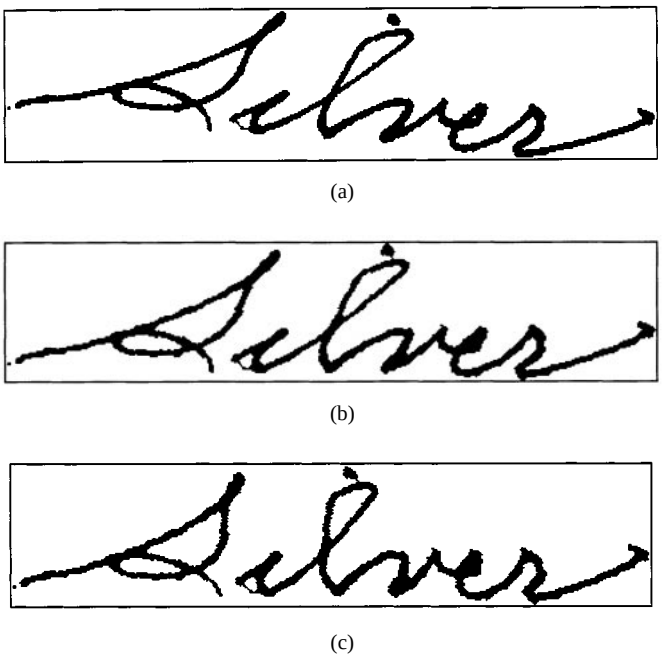
define our features. To obtain an estimate for the tilt angle we first estimated the core of the given image applying a morphological closing operation followed by an opening operation by a horizontal bar. The elongation angle of the core image is used as an estimate for the tilt angle and a rotation transformation is performed to correct for tilting.

The tilt-correction algorithm proceeds as follows.

I a word binary image
H a horizontal bar structuring element of width that varies linearly with the width of the word binary image
$W_1 = I \cdot H$ the result of closing I by H
$W_2 = W_1 \circ H$ the result of opening W_1 by H
M the largest connected component in W_2
θ the elongation degrees of M
R the resultant image from rotating I by θ degrees.

Before applying the rotation transformation, the tilt-correction algorithm checks to see whether θ is within a reasonable interval, otherwise the original image is returned (see Fig. 11).

For the slant angle, we first opened the tilt-corrected word image by a vertical bar to obtain a set of potentially slanted lines. The mean value of the elongation angles of these slanted lines is considered as an estimate for the word slant angle. A shear transformation is then applied to the tilt-corrected image in order to correct for the estimated slant.



(a) (b) (c)

Fig. 11. Tilt and slant correction. (a) Before tilt correction. (b) After tilt correction. (c) After slant correction.

The slant-correction algorithm proceeds as follows:

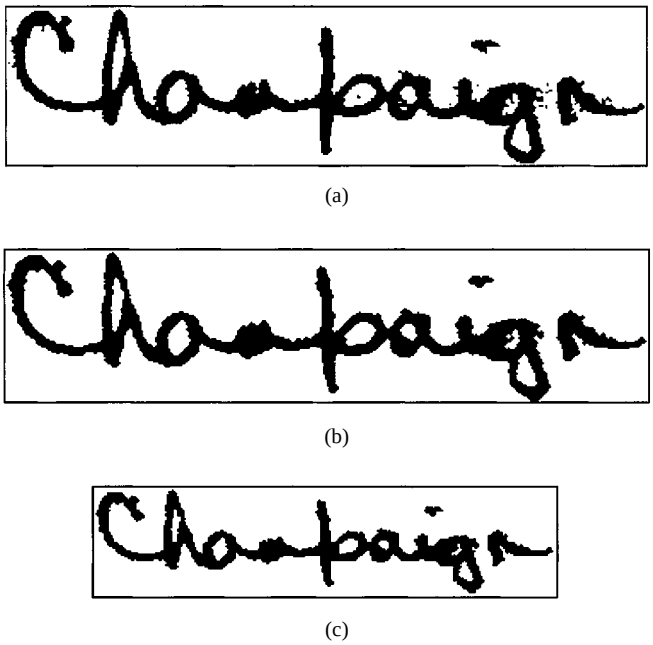
I a word binary image
V a vertical bar structuring elements of height that varies linearly with the stroke width of the word binary image
$C = I \circ V$ the result of opening I by V
FOR each connected component C_i in C
$\theta_i =$ elongation degrees of C_i
END
θ the average value of the elongation degrees θ_i
R the resultant image from shearing I by $(\pi/2 - \theta)$ degrees.

As for the tilt correction, before applying the shear transformation, the slant-correction algorithm checks to see whether $(\pi/2 - \theta)$ is within a reasonable interval, otherwise the original image is returned.

3) Smoothing and Scaling: A morphological smoothing operation is performed as follows. The word image is first closed and then opened by a disk of size one. This operation also helps to remove salt-and-pepper noise. Finally, the preprocessed word image is scaled to a fixed height (sixty four) keeping the aspect ratio as for the original image (Fig. 12).

C. Character-Level Truthing

The original training word images are not truthed to the character level, i.e., the information about where each character inside a word starts and ends is not provided. We identified the horizontal positions of the start and end of each character inside the word images provided in the traing sets. This process is very crucial for our proposed word recognition system since we



(a) (b) (c)

Fig. 12. Smoothing and scaling. (a) Before smoothing. (b) After smoothing. (c) After scaling.

need to compute features for character modules inside the word image.

IV. Experimental Results

We report the results of the baseline handwritten word recognition, provide examples of successes and failures, describe experiments to find the best structure for the character models, report performance of the individual classical and fuzzy models and combination techniques, and discuss research problems. The following experiments are described in this section.

1) For the initial experiments, we fixed the structure of the character HMM to be $N = 5$ and $M = 1$ and used the classical baseline HMM word recognition system with only transition features. The number of principal features used is $D = 4$. We analyzed the performance of the system using the standard testing set.
2) We varied the number of states from $N = 5$ to $N = 14$ and the number of mixtures per state from $M = 1$ to $M = 3$ in order to find the best model structure for the classical HMM. Here we added two more gradient features to the transition features. The number of principal features used is $D = 5$.
3) After finding the best structure for the classical HMM (which happened to be $N = 12$ and $M = 3$), We trained the GHMM using this structure. The GHMM is tested using the same testing data.

For all of the above experiments, we also report the performance when combining the results of the HMM and the GHMM using the Borda count method. The Borda count for a string in a given lexicon is the sum of the number of strings ranked below it by each word recognizer. The combined ranking is given by sorting the lexicon strings so that their Borda counts are in descending order.

TABLE II
Results of Initial Experiments

HMM Word Recognition Results D=4, N=5, and M=1			
Test Set Size = 317		Lexicon Size ~ 10	
Rank	Not Using Relative Width Statistics	Using Relative Width Statistics	Borda Count Decision Combination
1	68.8	71.6	76.7
2	83.9	83.3	87.7
3	89.3	89.3	91.2
4	93.4	91.8	92.8
5	94.6	93.4	93.7
6	95.3	95.0	95.3
7	96.5	96.2	96.2
8	96.5	96.5	97.2
9	97.2	96.9	97.5
10	97.5	97.5	97.8

A. Initial Experiments

To build word models by concatenating character models, values have to be assigned for the between-character transition probabilities. At first, we applied a stationary approach and set both the between-character transition probability and the probability of staying at the last state of the previous character to 0.5 for all character breaks. This implementation resulted in 68.8% top rank word recognition score using the BD city words data set (317 images) and the smallest lexicon (lexicon size ~ ten entries).

For the second implementation we tried to vary those probabilities with time utilizing the relative character width information estimated from the training data for all character classes. The location of each character break is estimated by partitioning the word image according to the mean values of the character widths. The between character transition probabilities are varied exponentially as a function of the distance between the column location and the character break location. This nonstationary approach resulted in 71.6% top rank word-recognition score for the same testing data set. A significant improvement (76.7% top rank recognition) was achieved when combining the two word recognition results using the Borda count method. A summary of the results is shown in Table II.

B. Performance Analysis of Initial Experiments

It worth noting that we ignored blank columns when matching against lexicon strings, i.e., the input to the Viterbi algorithm is a sequence of nonzero feature vectors. It seems that this kind of information is useful and should be utilized "carefully" for printed words and also sometimes as a guess for the first character break. The segmentations (character breaks) given by the Viterbi algorithm for some correctly classified and misclassified words are shown in Figs. 13 and 14.

It is interesting to find that for some of the long words even though not all characters breaks are found correctly, they are still correctly classified as long as a significant number of the correct character breaks are discovered.

Although some of the word images are not correctly preprocessed, reasonable approximations for the character breaks are discovered by the system. Sometimes when the word string truth

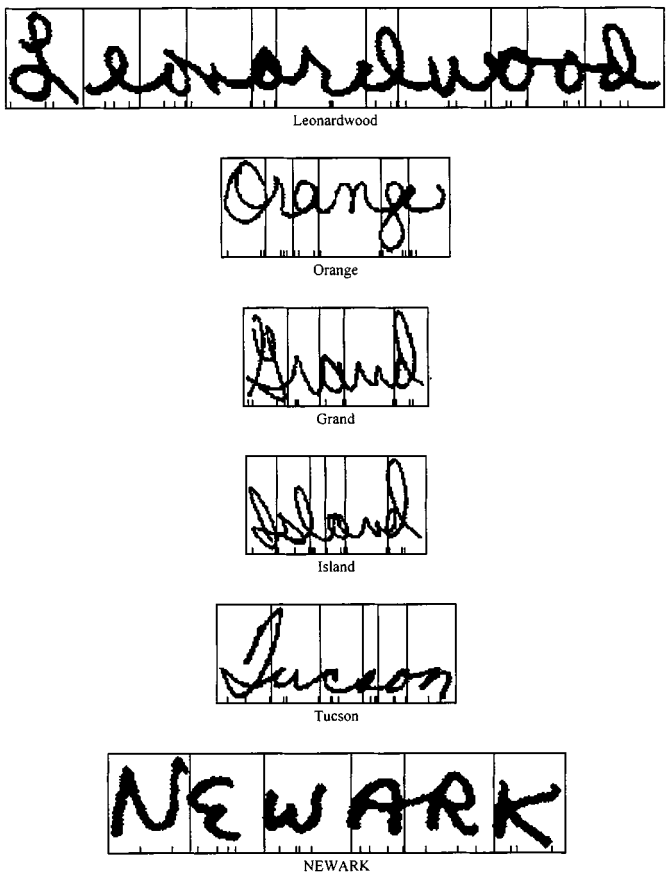


Fig. 13. Correctly classified samples from initial experiments.

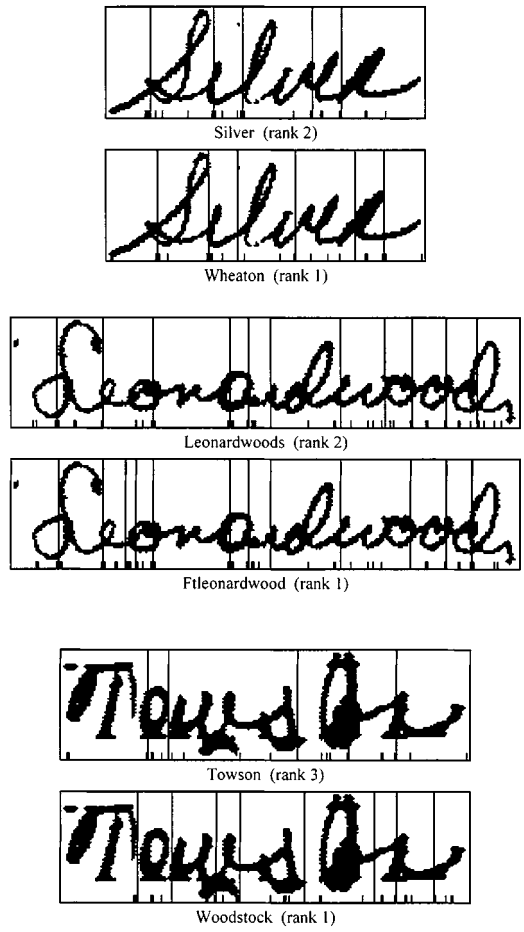


Fig. 14. Misclassified samples from initial experiments.

is given in a wrong case sequence the system matches against the actual sequence.

It is also clear that some of the characters like "m," "w," and capitals need more states while others like "i," and "e" may need less than five states. It seems that allowing to skip one state is helping to some degree, but it is still important to increase the number of states for the upper case characters. Special care should be given to the first character since it is usually "well drawn" and also to the last character since it sometimes ends with "flair" or touches a punctuation mark. Cursive upper case characters are the most complex among all character classes. They involve a great amount of variation in the style, size, and the stroke width.

A major problem occurs when the first character or some other characters like "t" or "g" overshadow or undershadow their neighbors. Possible solutions for this problem could be exploring new feature sets, training models for character pairs like "ti," "ng," etc. Another problem is that the system is not taking into account horizontal and global information. Some information like the gradient of the top and bottom contours, horizontal transitions, connected components, state duration might help for discriminating among lexicon entries.

It is very important to note that postprocessing is sometimes very dangerous since it can exclude the correct choice. It is more appropriate to use any possible information inside the matching process itself (if it is possible) and to try to avoid all kinds of early commitments.

C. Performance of the Classical and Fuzzy Models

The initial experiments are promising, serving primarily to highlight the subproblems that became the primary objects of study for the baseline HMM word-recognition system.

High levels of performance require that large amounts of specific knowledge be organized and brought to bear on the problem. It is evident from even a casual look at the misclassified examples that more features are required to enhance the performance. The effect of adding the two gradient features to the observation vector and increasing the number of principal features from $D = 4$ to $D = 5$ was found to be fruitful (see Fig. 15). The number of states was varied from $N = 5$ to $N = 14$ while setting the number of clusters per state to $M = 2$ and $M = 3$. In essence, adding more states and mixtures means storing more information which is thought to represent different styles in character classes. The results from these experiments are shown in Fig. 16.

As mentioned before, it worthwhile to consider the effect of varying both the number of states and the number of clusters per state in order to achieve better performance. Since this task requires repeating the training and testing experiments and, therefore, more computing effort we used the smallest lexicons to evaluate different model structures. The best word-recognition results for the baseline system are given by the models using 12 states $(N = 12)$ and three mixtures $(M = 3)$.

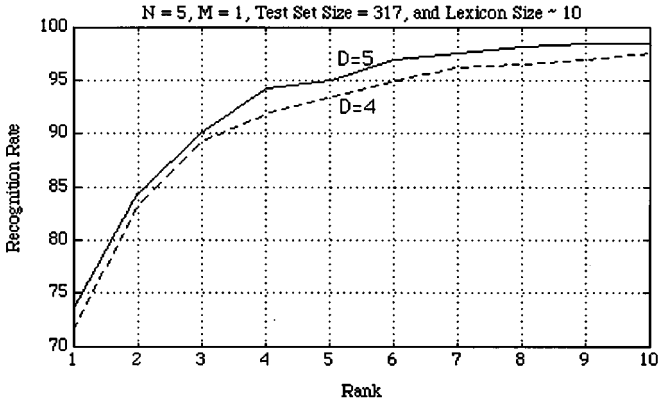


Fig. 15. Effect of adding gradient features.

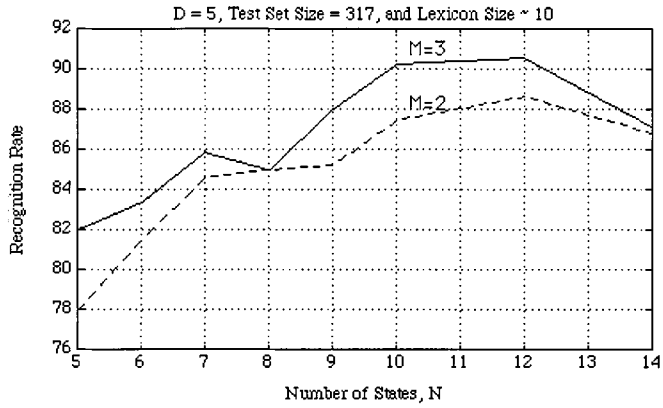


Fig. 16. Effect of changing number of states (N) and number of mixtures (M).

TABLE III
CLASSICAL AND FUZZY HIDDEN MARKOV MODELS PERFORMANCE

Word Recognition Results
D=5, M=3, and N=12
Test Set Size = 317 Lexicon Size ~ 10

Rank	HMM Probability Measure	FHMM Possibility Measure	Borda Count Decision Combination
1	90.5	91.5	94.3
2	95.0	96.2	97.2
3	96.5	97.5	97.5
4	98.1	98.4	98.4
5	98.7	98.7	98.7
6	99.4	99.1	99.1
7	99.4	99.4	99.4
8	99.7	99.4	99.7
9	99.7	99.4	99.7
10	99.7	99.7	99.7

Word Recognition Results
D=5, M=3, and N=12
Test Set Size = 317 Lexicon Size ~ 100

Rank	HMM Probability Measure	FHMM Possibility Measure	Borda Count Decision Combination
1	71.6	73.2	78.2
2	82.3	83.0	86.1
3	86.8	87.1	89.6
4	89.3	90.2	92.8
5	90.2	91.2	93.1
6	90.9	92.4	94.0
7	91.8	93.7	95.0
8	92.4	95.0	95.6
9	94.0	95.6	95.9
10	94.3	95.6	95.9

After fixing the model structures to 12 states and three clusters per state ($N = 12$ and $M = 3$), experiments were carried for the classical and fuzzy HMM's using lexicons with average sizes 10 and 100 and the BD test set. The fuzzy model was found to outperform the classical one as shown in Table III and Fig. 17.

The combination of the two individual classifier using the Borda count method resulted in higher recognition rates. We used the Borda count method, which is the simplest decision combination technique for word recognition to demonstrate the fact that the individual classical and fuzzy classifiers make different mistakes and, therefore, give better results when combined.

The mixed success achieved as a result of combining the two classifiers has again served to highlight the fact that high levels of performance require that large amounts of specific knowledge be organized and brought to bear on the problem.

V. CONCLUSION

We have presented a complete scheme for off-line handwritten word recognition. The challenge for a handwritten word recognition is to employ various kind of knowledge about the application domain and analyze the word image in order to make inferences about its identity. An important ingredient to the knowledge base on which the analyzing processes are operating is the collection of procedures and techniques for image processing, feature extraction, and classification. The heart of our scientific aim for this study is defined as the design and engineering of an off-line handwritten word recognition system at the conceptual and theoretical level as well as at the methodological and implementational ones.

The presented system is a complete scheme for totally unconstrained handwritten word recognition using HMM's and a segmentation-free approach. The system applies the contextual knowledge provided by the lexicons in the process of recognizing individual words to attain higher levels of accuracy. A very large database is used to be aware of the writing variations, insure the algorithms robustness, and to obtain significant estimates of efficiency rates. A given assumption to our present system is that the characters inside a word image could be reasonably separated by vertical lines. After binariztion of the input gray word image, many preprocessing steps are used to ensure satisfaction of this requirement.

The detection of meaningful objects and their relations is accomplished by computing transition and gradient features and constructing word models from the constituent character models. We used a segmentation-free continuous density hidden Markov modeling approach to improve the performance of the existing techniques in the literature. Our approach is the first to use continuous density HMM's for a segmentation-free handwritten word recognition. In addition, we developed a generalization of the classical hidden Markov models using fuzzy measures and fuzzy integrals resulting in a fuzzy hidden

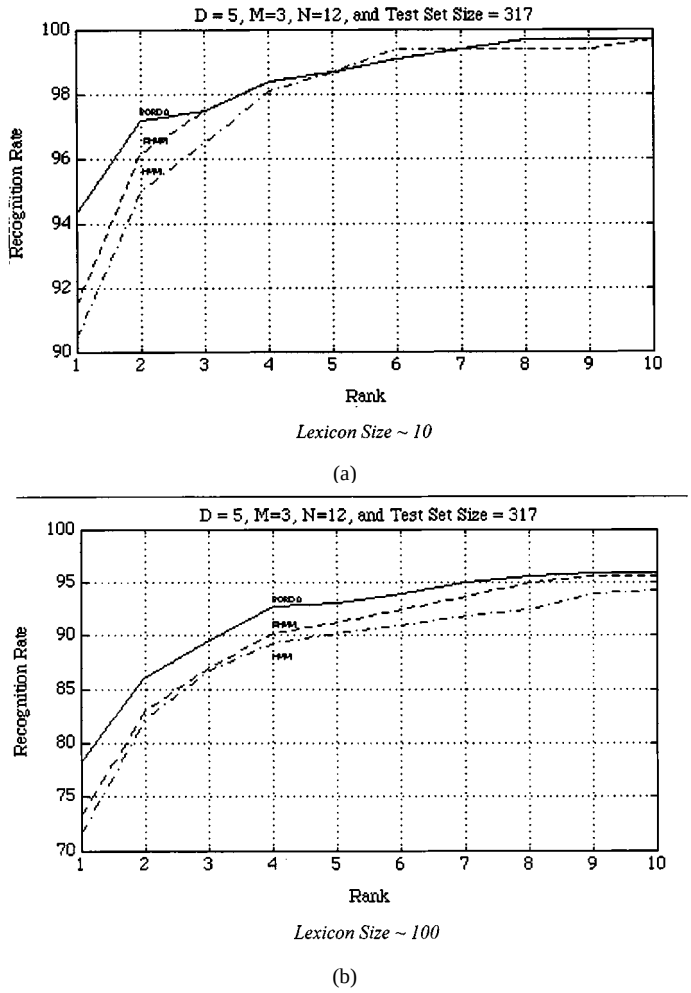


Fig. 17. Classical and fuzzy hidden Markov models performance. (a) Lexicon size ~ 10. (b) Lexicon size ~ 100.

Markov modeling framework. Our generalization relies heavily on the use of the conditional fuzzy measure. An attractive property of this generalization is the fact that if we used Choquet integral as the general fuzzy integral, multiplication as the fuzzy intersection operator, and a probability measure as the fuzzy measure then the generalized HMM reduces to the original probabilistic hidden Markov mode. In this sense, the classical model is one of many models provided by the generalization.

Methods for training the classical and fuzzy models are described. Learning the optimal model parameters is the most difficult problem for both the classical and fuzzy models where there is no closed-form solution. The described learning algorithms are iterative techniques seeking optimal solutions. One of the advantages of using HMM's is that the parameters do have meaning associated with them such as the expectation of transitions among states, the means and covariance matrices of clusters of feature vectors inside each state. This provides useful information for choosing initial values, which is very helpful for finding good approximation of the modeling parameters by applying the reestimation formulas inside the training algorithm.

Several experimental results involving the classical HMM's, the generalized HMM's assuming a possibility fuzzy measure, and combination strategies were presented. Real data obtained from the U.S. Post Office mail stream is used for conducting the experiments. The results show that the proposed methods are promising and effective. We demonstrated that the generalized HMM's can be used to achieve significantly higher classification rates for the application of off-line handwritten word recognition.

The generalization also serves as a tool for constructing multiple classifiers. It is increasingly held that the use of a combination of multiple classifiers may be able to improve significantly the performance of a handwritten word-recognition system. Although the fuzzy models use the same features as the classical ones, fusing the information using a nonlinear integration results in different character model parameters when training and also different matching scores when testing. The classical HMM and the fuzzy HMM classifiers trained with exactly the same training data, the same structures, and the same initial model parameter values make different mistakes and, therefore, give better results when combined.

We have shown that classical and fuzzy HMM's, given the right encoding method, are a versatile pattern matching tool for solving the off-line handwritten word-recognition problem. The generalized (fuzzy) HMM shares the ability to model sequential processes with the classical one and, therefore, can also be used for similar applications of the classical HMM. The generalization provides more freedom and flexibility to aggregate the sequential information obtained from the observation sequence. With such flexibility and freedom to choose comes the obligation to choose right.

There are many possible future improvements for the implemented system that require the availability of more training data. The most important one is to directly model some of the common subwords such as "tt," "ton," "Te," and "ing" where some of the characters overshadow their neighbors. Another improvement could be achieved by training more than a single model per each single character class by manually dividing the training data into cursive and printed word styles.

A difficulty arises when matching the word image against the lexicon strings as a result of considering two possible character case sequences for each string inside a lexicon. One possible approach to deal with this difficulty is to build a single model for each character instead of two—one for the lower case and the other for the upper case. The problem with this approach is that there is a large degree of variations between the lower and upper case shapes for most of the character classes. A more reasonable approach is to develop techniques for combining character models and use it inside the word matching process. This approach is also useful when dealing with models of different styles for the same character class.

ACKNOWLEDGMENT

The authors would like to thank Prof. J. Keller, Prof. X. Zuang, Prof. R. Krishnapuram, and Prof. P. Blackwell of the University of Missouri at Columbia for their valuable discussions and suggestions. The authors would also like to thank the referees for their appreciated detailed review of the manscript.

REFERENCES

[1] C. Y. Suen, C. Nadal, R. Legault, T. Mai, and L. Lam, "Computer recognition of unconstrained handwritten numerals," *Proc. IEEE*, vol. 80, pp. 1162–1180, 1992.

[2] M. Parizeu, "A fuzzy-syntactic approach to allograph modeling for cursive script recognition," *IEEE Trans. Pattern Recogn. Mach. Intell.*, vol. 17, no. 7, July 1995.

[3] M. Shridhar and F. Kimura, "Handwritten address interpretation using word recognition with and without lexicon," in *IEEE Int. Conf. Syst., Man, Cybern.*, Vancouver, BC, Canada, Oct. 1995, pp. 2341–2346.

[4] L. Lam, C. Y. Suen, D. Guillevic, N. W. Startly, M. Cheriet, K. Liu, and J. N. Said, "Automatic processing of information on cheques," in *IEEE Int. Conf. Syst., Man, Cybern.*, Vancouver, BC, Canada, Oct. 1995, pp. 2347–2358.

[5] G. Didier and S. Ching, "Cursive script recognition: A sentence level recognition scheme," in *Int. Workshop Frontiers Handwriting Recogn.*, Taipei, Taiwan, Dec. 1994, pp. 216–223.

[6] P. D. Gader, J. M. Keller, R. Krishnapuram, J. H. Chiang, and M. Mohamed, "Neural and fuzzy methods in handwriting recognition," *IEEE Comput.*, vol. 30, pp. 79–86, Feb. 1997.

[7] P. D. Gader, M. Mohamed, and J.-H. Chiang, "Handwritten word recognition with character and inter-character neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 27, pp. 158–165, Feb. 1997.

[8] J. Chiang and P. D. Gader, "Hybrid fuzzy-neural systems in handwritten word recognition," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 497–510, Nov. 1997.

[9] A. Gillies, "Cursive word recognition using hidden Markov models," in *Proc. U.S. Postal Service Adv. Technol. Conf.*, Washington, DC, Nov. 1992, pp. 557–563.

[10] T. K. Ho, J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 66–75, Jan. 1994.

[11] T. S. Huang and C. Y. Suen, "A method of combining multiple experts for the recognition of unconstrained handwritten numerals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, pp. 90–94, Jan. 1995.

[12] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 550–554, May 1994.

[13] P. D. Gader, D. Hepp, B. Forester, T. Peurach, and B. T. Mitchell, "Pipelined systems for recognition of handwritten digits in USPS ZIP codes," presented at the SPSE's 43rd Annu. Conf., Rochester, NY, May, 1990 , pp. 60–63.

[14] P. D. Gader, M. Mohamed, and J. Keller, "Fusion of handwritten word classifiers," *Pattern Recogn. Lett.—Special Issue Fuzzy Pattern Recogn.*, vol. 17, no. 6, pp. 577–584, May 1996.

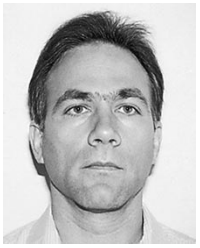
[15] M. Chen, "Off-line handwritten word recognition using hidden Markov models," in *Proc. U.S. Postal Service Adv. Technol. Conf.*, Washington, DC, Nov. 1992, pp. 563–579.

[16] M. Chen and A. Kundu, "An alternative to variable duration HMM in handwritten word recognition," in *Proc. 3rd Int. Workshop Frontiers Handwriting Recognition*, Buffalo, NY, May 1993, pp. 48–54.

[17] O. Nobuyuki, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.

Magdi A. Mohamed (M'93) received the B.Sc. degree in electrical engineering from the University of Khartoum, Sudan, in September 1983, and the M.S. (computer science) and Ph.D. (electrical and computer engineering) degrees from the University of Missouri-Columbia, in 1991 to 1995, respectively.

From 1985 to 1988, he worked as a Teaching Assistant at the Computer Center, University of Khartoum, Sudan. He also worked as a Computer Engineer for consultation and hardware support at Computer Man Ltd. in Khartoum, Sudan, and as an Electrical and Communication Engineer at Sudan National Broadcasting Corporation in Omdurman, Sudan. From 1991 to 1995 he was a Research and Teaching Assistant in the Department of Electrical and Computer Engineering, University of Missouri-Columbia. He worked as a Visiting Professor in the Computer Engineering and Computer Science Department at the University of Missouri-Columbia from 1995 to 1996. He is currently working as a Research Scientist at Motorola Human Interface Laboratories, Lexicus Division. His research interests include online and offline handwriting recognition, image processing, computer vision, fuzzy set theory, neural networks, pattern recognition, parallel and distributed computing, artificial intelligence, and fractals and chaos theory.



Paul Gader (M'87–SM'99) received the Ph.D. degree for research in image processing from the University of Florida, Gainesville, in 1986.

Since 1986, he has worked as a Senior Research Scientist at Honeywell Systems and Research Center, Minneapolis, MN, as an Assistant Professor of Mathematics at the University of Wisconsin-Oshkosh, and as a Research Engineer and Manager at the Environmental Research Institute of Michigan (ERIM), Ann Arbor, MI. He is currently an Associate Professor of Computer Engineering and Computer Science at the University of Missouri-Columbia. He has been actively involved in handwriting recognition research since 1989. He has performed research in handwritten and machine-printed line, word, and character segmentation, zip code and street number location and recognition, post office box detection and recognition, on handwritten digit and alphabetic character recognition, handwritten word recognition, and on multiple classifier fusion in digit, character, and word recognition. He is also conducting research in land-mine detection, mathematical morphology, medical imaging, automatic target recognition, and neural networks.