Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques

Magdi Mohamed and Paul Gader

Abstract—A lexicon-based, handwritten word recognition system combining segmentation-free and segmentation-based techniques is described. The segmentation-free technique constructs a continuous density hidden Markov model for each lexicon string. The segmentation-based technique uses dynamic programming to match word images and strings. The combination module uses differences in classifier capabilities to achieve significantly better performance.

Index Terms—Hidden Markov models, dynamic programming, handwritten word recognition, character recognition, neural networks, character segmentation.

1 Introduction

AUTOMATIC reading of handwritten words is a difficult problem, not only because of the great amount of variations involved in the shape of characters, but also because of the overlapping and the interconnection of the neighboring characters. Furthermore, when observed in isolation, characters are often ambiguous and require context to minimize the classification errors. The existing development efforts have involved long evolutions of differing classification algorithms, usually resulting in a final design that is an engineering combination of many techniques [12].

The problem of handwritten word recognition consists of many difficult subproblems and each requires serious effort to understand and resolve. One of the most important problems in segmentation-based word recognition is assigning character confidence values for the segments of words to reflect the ambiguity among character classes. 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, etc. The choice of one or another nominal method for evaluating features is as important as the choice of what features to evaluate and method for measuring them [4], [6], [8], [9], [10].

There are as many as 52 classes, when considering upper and lower case characters as different classes, with various levels of difficulty for recognition according to the complexity, style variations, and the ambiguity among them. Isolated character recognition is further complicated by the differences such as upper and lower, cursive representations of each letter, and the number of inherently broken multi-stroke characters. A significant factor in the ultimate success of an effort is the gathering of an adequately complete collection of samples under realistic conditions. Another problem in character recognition is that the distribution of samples among the different classes in the training set is usually non uniform (see Fig. 4) and may also follow a different distribution than the expected occurrence or relative importance of the class in operation.

Computer recognition systems for handwritten characters generally have no difficulty recognizing samples which are well-formed and closely resemble the standards for each class. However, the fact that many pieces of characters and unions of pieces of characters with other characters can look like characters, results in a very high potential for false matches. A glance at the current research (cf. [1], [23], [24], [25], [26], [27], [29], [30]) and a little thought should convince a reader that handwritten word recognition by computer consists of more than just isolating and reading the individual characters in a word. Contextual information is not only useful, it is often necessary. People are able to read words with illegible and ambiguous characters. Many alphabetic characters are ambiguous when read out of context. In fact, the same pixel pattern can represent different characters in different words. Some examples of ambiguity are shown in Fig. 1. There is still a need for development of techniques that perform recognition well on characters and that also recognize when confronted with non-characters. Gader et al. [31], [32] approached this problem by using hybrid fuzzy neural systems.

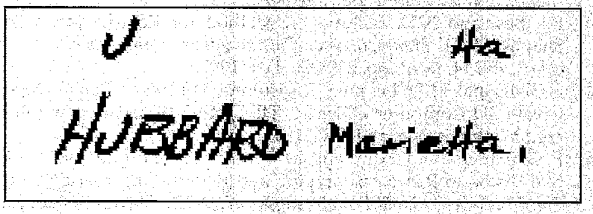


Fig. 1. Ambiguous patterns in handwritten words.

One implication of these considerations is that segmentation is ambiguous and prone to failure. Another implication is that designing a robust handwritten character classifier that is to be used for a segmentation-based word recognition system is a difficult task. A segmentation-free technique is proposed in this paper to improve the performance when the handwritten words are not easily segmented. The technique uses continuous density hidden Markov character models to construct word models. Our approach is the first published approach to use continuous density HMMs for segmentation-free handwritten word recognition. The observation vectors used by the models do not require segmentation of a word image into characters. Thus, our approach has advantages over segmentation-based approaches [2]. Other advantages are described in Section 2.

The purpose of this paper is to describe a handwritten word recognition system using a combination of segmentation-free model-based and segmentation-based systems. The paper reviews the main features of the segmentation-free technique and gives a detailed description of automatic learning of Markov models for recognition of handwritten words. The segmentation-based technique was described in [7]. For the sake of completeness a brief description of this technique is provided. Descriptions of the database, the preprocessing steps, and the experimental results from the individual and combined techniques are also provided.

2 Hidden Markov Models

The statistical methods of hidden Markov modeling were initially introduced and studied in the late 1960s. They have been found to be extremely useful for a wide spectrum of applications. This technique was applied to speech recognition problems prior to its application to handwritten word recognition problems. Since the recognition of handwritten words has many similarities with speech recognition, researchers tried to apply this technique to

• *The authors are with Department of Electrical and Computer Engineering, University of Missouri, Columbia, MO 65211.*
E-mail: {mohamed, gader}@ece.missouri.edu.

Manuscript received Nov. 18, 1994; revised Jan. 22, 1996. Recommended for acceptance by M. Mohiuddin.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number P96012.

0162-8828/96$05.00 ©1996 IEEE

handwritten word recognition applications. Chen et al. [3] 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 successful application of HMM in a character segmentation-based word recognition was presented by Chen et al. [2]. Chen used continuous density, variable duration hidden Markov model (CDVDHMM) to build character models. The character models are left-to-right HMMs in which it is possible to skip any number of states. The system recognized 100 lexicon strings with approximately 80% success out of 3,000 testing words. 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 [11] 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. This technique is appropriate for cursive words when the individual character identities are often not distinguished except in the context of the word.

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 HMMs 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 HMMs is the inversion of the covariance matrices. We overcame this difficulty by reducing the dimensionality using principal component analysis and approximating the densities as a mixture of Gaussian densities with diagonal covariance matrices. A detailed description of all system modules is given in [33]. The results provided in this paper demonstrate that each of the segmentation-free and segmentation-based techniques behaves reasonably well when confronted with various styles of input words and much better for special cases. The combination of the two techniques resulted in a significant improvement in the overall recognition rates.

2.1 Overview of HMM

A probabilistic function of a hidden Markov chain is a stochastic process generated by two interrelated mechanisms, an underlying Markov chain having a finite number of states, and a set of random functions, one of which is associated with each state. At discrete instants of time, the process is assumed to be in some state and an observation is generated by the random function corresponding to the current state [28]. The underlying Markov chain then changes states according to its transition probability matrix. The observer sees only the output of the random functions associated with each state and cannot directly observe the states of the underlying Markov chain; hence the term hidden Markov model.

In principle, the underlying Markov chain may be of any order and the outputs from its states may be multivariate random processes having some continuous joint probability density function. We use continuous density Markov chains of order one, i.e., those of which the probability of transition to any state depends only upon that state and its predecessor. There are a finite number, say N, of states in the model. At each clock time a new state is entered based upon a transition probability distribution which depends on the previous state (the Markovian property). After each transition is made, an observation output symbol is produced according to a probability distribution which depends on the current state. This probability distribution is held fixed for the state regardless of when and how the state is entered. There are thus N such observation probability distributions which, of course, represent random variables or stochastic processes.

2.2 Feature Extraction

The description of a word binary image as an ordered list of observation vectors $\{O_1 O_2 \ldots O_3\}$ is accomplished by encoding a combination of features computed from each column in the given preprocessed image. In this sense, each column in a word image is represented by a feature vector (an observation vector). A column belongs to each state of each character model within a certain probability. This representation uses what 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.

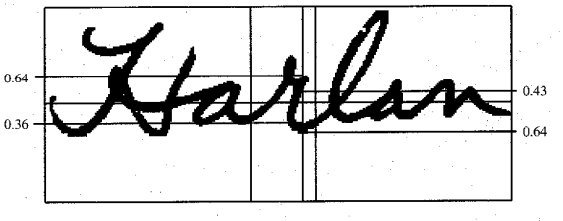


Fig. 2. Transition feature computations.

The first step of the feature computation is to estimate the center-line of the word using its horizontal projection. The centroid of the horizontal projection of the word image is used to estimate 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 virtual bounding-box is created by adding blank rows above or below the word image until the estimated center line lies in the middle of the resultant bounding-box. The core region 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 (see Fig. 2). 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 *MXNT* is set to 4. If there are less than *MXNT* transitions on a column, then the "nonexistent" transitions are assigned a value of 0. Two more features were added representing the gradient of the north and south contours resulting in a total of ten features per each image column. The gradient of the north contour is computed as the difference between the heights of the first transitions from the top in successive columns. The south contour is computed similarly using the first transitions from the bottom. When computing features for training 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 a linear interpolation technique. Setting $T = 24$ effectively performs width normalization while scaling the transition features inside the word image performs height normalization.

2.3 HMMs Structure and Training

For the purpose of isolated handwritten word recognition, it is useful to consider a special class of absorbing Markov chains that leads to what is called 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 appropriate for building word models from 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 fixed the number of states for all character models to twelve, but we also allowed skipping one state such that if a character class needs less than twelve states some of the skipping probabilities could be relatively high.

A continuous density hidden Markov model is trained for each upper and lower character class using the transition features computed inside word images. The dimensionality of the observation vector is reduced from 10 to 5 using principal component analysis based upon the covariance matrix of all class feature vectors. We used twelve states for each model and two Gaussian mixtures per state. Initially the mean and covariance matrix for each state mixture are estimated after dividing the resampled training transition vectors equally between the states. The standard reestimation formulae are used to train the models assuming diagonal covariance matrices for the Gaussian mixtures [28]. The character models were trained on the word images included in SUNY database for training purposes [14]. The training data is hand-segmented, where the character boundaries are manually identified inside word images. For each character class we used all the available training samples. For example, 3,474 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. 4. The database is fully described in Section 4.

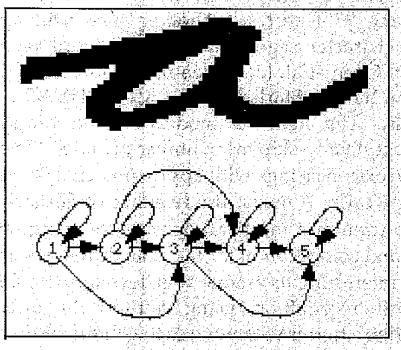


Fig. 3. A character HMM assuming five states.

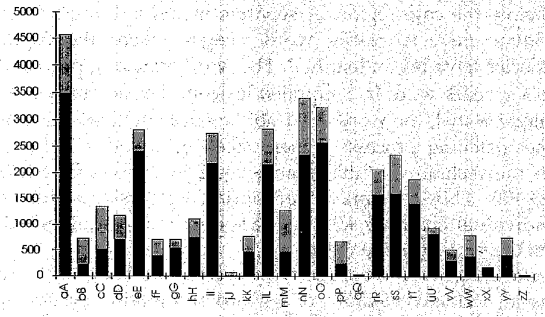


Fig. 4. Distribution of training samples.

2.4 Matching Strategies

The word strings provided by the lexicon are in upper-case format. 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 the 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. The segmentation of the word into characters is performed as a byproduct of the model matching process, and can be obtained from the optimal state sequence.

3 Dynamic Programming

The second word recognition algorithm uses image segmentation and dynamic programming. This technique has been fully described in [7]. Some of the most successful results have come from segmentation based techniques that rely on dynamic programming [15], [16], [17], [19], [22]. These approaches are often referred to as lexicon-driven approaches because an optimal segmentation is generated for each string in the lexicon.

The inputs are a binary image of a word and a lexicon. A word image is segmented into subimages called primitive segments, or

primitives. Each primitive ideally consists of a character or a part of a character. A segment is either a primitive or a union of primitives. Ordering primitives from left to right yields a partial ordering on the segments. A segmentation into characters is a path through the partially ordered set of segments. Dynamic programming is used to find the best path, that is, the best sequence of segments to match a given string. The value of a path is the sum of the match scores of the segments in the path to the characters in the string as given by the output activations of the neural networks. An overview is shown in Fig. 5.

3.1 Segmentation

The segmentation process is a refinement of that described in [18], [21]. We describe it briefly. Connected components are computed, punctuation is detected and removed, and bars, such as those in "D", "E", "F", etc., are grouped with larger connected components. The result is a set of initial segments that generally consist of images of one or more characters. Those consisting of more than one character need to be split. Each segment in the initial segmentation is passed through a splitting module. The splitting module uses the distance transform to detect possible locations to split initial segments. The distance transform encodes each pixel in the background using the distance to the stroke. Split paths are formed that stay as far from the stroke as possible without turning too much. Heuristics are used to define starting points for the paths based on the shape of the image and to modify the distance function. The sequence of images resulting from splitting and initial segmentation are postprocessed to correct for images that are very small or very complex. The postprocessed sequence is the set of primitive segments.

3.2 Dynamic Programming Matching

The dynamic programming matcher takes a word image, a string, and the word image primitives and returns a value between 0 and 1 indicating the degree to which the word image represents the string. The algorithm is implemented using an array approach. The rows of the array correspond to the characters in the string. The columns of the array correspond to primitives. The ij element is the best match between the first i characters in the string and the first j primitives. This value may be $-\infty$ if there is no legal match. A union of two segments is considered legal if the two segments pass a sequence of tests. The tests measure closeness and complexity.

4 The Database and Preprocessing

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 at Buffalo (SUNY). This database is widely available and serves as a benchmark for evaluation purposes [14]. We first describe the database and then the preprocessing steps performed to the original images. The handwritten words (cities, states, and ZIP codes) are provided on the CDROM in full 8-bit gray scale. These data were divided into training and testing sets at SUNY by randomly choosing approximately ten percent 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 was retained for training purposes. This breakdown is shown in Table 1.

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.

TABLE 1
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 procedure which we used for binarization was first presented by Otsu [20]. It is a very simple technique, utilizing the zeroth and first-order cumulative moments of the gray-level histogram. 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 [5]. 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 (spanning) operation is used to recover these parts. Word image borders are sometimes surrounded by letter 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 (see Fig. 6).

Tilt and slant correction is one of the most difficult preprocessing steps since it involves estimating angles to rotate and shear the original word image. To obtain an estimate for the tilt angle we first estimated the core region 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. 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 detailed description of the preprocessing techniques is provided in [33]. For the 317 images in the BD test data set, the tilt and slant correction technique failed on 24 images where either the image became worse after applying the technique or there is a significant (tilt or slant) that is left unchanged.

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 for getting rid of salt-and-pepper kind of noise. Finally the preprocessed word image is scaled to a fixed height (64), keeping the aspect ratio as for the original image.

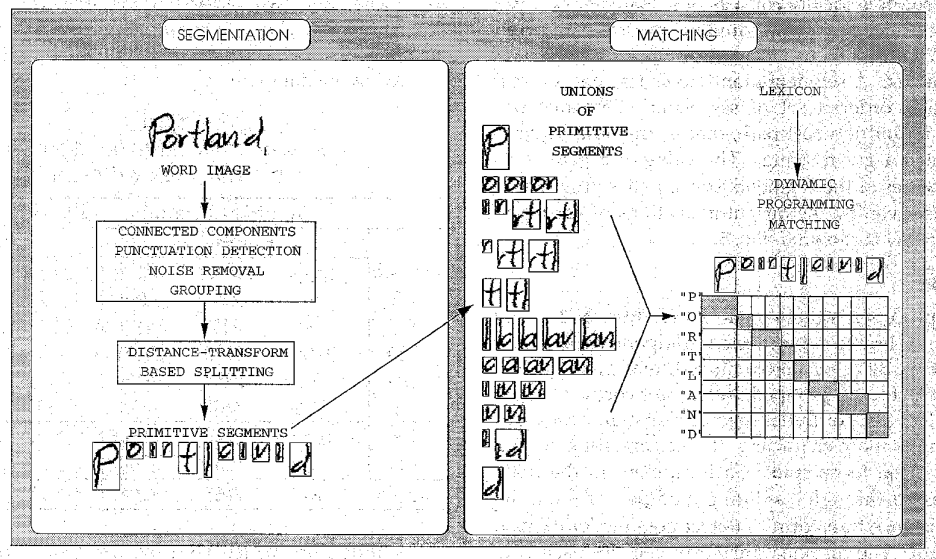


Fig. 5. Overview of segmentation-based word recognition algorithm.

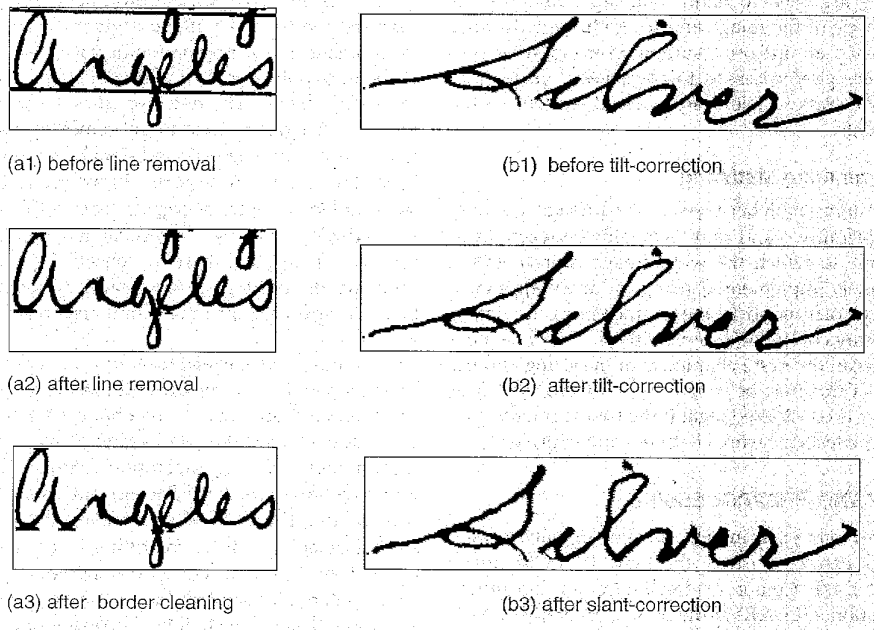


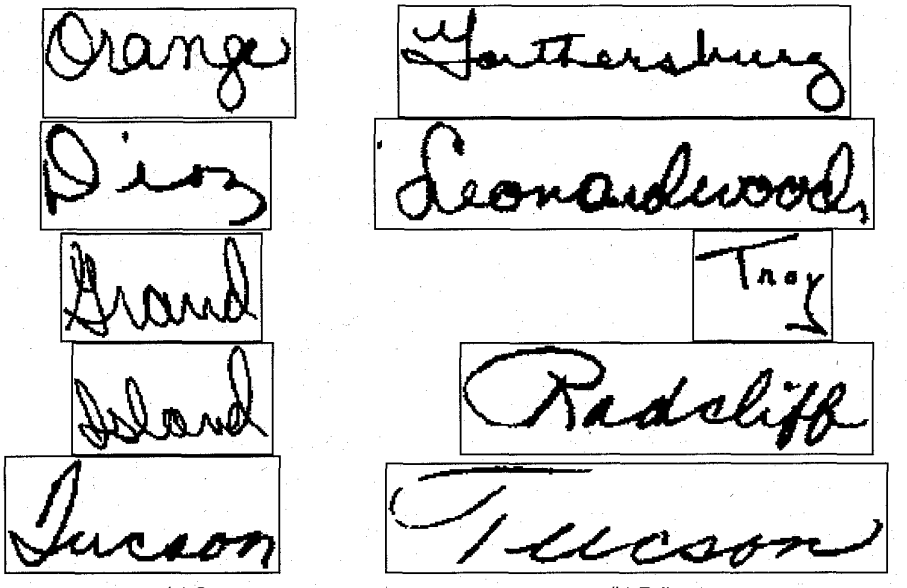
Fig. 6. Preprocessing steps: (a) cleaning, (b) tilt and slant correction.

5 Experimental Results

The recognition rates for each module separately and for the combination of the modules are shown in Table 2. All rates are for the 317 images in the BD city names test set. None of these images are used in any of the training. The lexicons used were those provided in the CEDAR CDROM. We used the lexicon set having average dictionary size of 100 [14]. If the correct string was not in a given lexicon, we inserted it. This usually creates a more difficult problem since misspellings are often the reason for the correct string not being in the lexicon. Since misspelled strings are very similar to the correctly spelled strings, the probability of error is high.

The results of the HMM classifier were combined using a combination of thresholds and Borda count [13]. The segmentation based method produces a confidence value between 0 and 1 that is comparable for different word images. The HMM method does not produce a confidence value that can be compared for different word images. The decision logic for computing the rank of a string in the lexicon is as follows:

```
IF the segmentation-based system confidence is
HIGH
THEN use the ranks produced by the segmenta-
tion-based system
ELSE IF the segmentation-based system confi-
dence is LOW
    THEN use the ranks produced by the HMM
    ELSE  SET output rank = segmentation-based
    rank + HMM rank
```



(a) Success (b) Failure

Fig. 7. Examples of success and failure modes for HMM technique.

The values of the parameters HIGH and LOW were set to be 0.40 and 0.03, respectively.

A major problem for the HMM segmentation-free technique occurs when character like "t" or "g" overlap their neighbors (see Fig. 7). Possible solutions for this problem could be exploring new feature sets, training models for character pairs like "ti", "ng", and common subwords like "ton".

TABLE 2
HANDWRITTEN WORD RECOGNITION RESULTS

Rank	DP	HMM	Combination
1	83.9	78.9	89.3
2	90.5	85.5	92.7
3	92.4	88.6	94.0
4	94.6	91.2	95.3
5	94.6	92.1	95.3
6	95.6	92.7	95.6
7	95.6	93.4	96.2
8	95.6	93.7	96.2
9	95.6	94.0	96.8
10	95.9	94.3	97.2

6 CONCLUSION

A system is described that performs handwritten word recognition using a combination of segmentation-free and segmentation-based techniques. The segmentation-free technique apply lexicon information and character level information provided by HMMs. The results show that HMM provides a good approximation of the probability densities. The segmentation-based technique combines the lexicon information with the character-level recognition results obtained from neural networks using dynamic programming. A very important feature of theses techniques is that each of them behaves reasonably well when confronted with various styles of input words and much better for special cases. The combination of the two techniques yielded a significant improvement in the overall recognition results.

ACKNOWLEDGMENT

The segmentation-based dynamic programming research was supported by the Environmental Research Institute of Michigan and the United States Postal Service.

REFERENCES

[1] A.M. Gillies, "Word Verification for the Contextual Analysis of Address Block Images," *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1990, pp. 247-255.

[2] M. Chen and A. Kundu, "An Alternative to Variable Duration HMM in Handwritten Word Recognition," *Proc. Third Int'l Workshop on Frontiers in Handwriting Recognition*, Buffalo, N.Y., pp. 82-92, May 1993.

[3] M. Chen, "Off-Line Handwritten Word Recognition Using Hidden Markov Models," *Proc. United States Postal Service Advanced Technology Conference*, Washington, D.C., pp. 563-579, Nov. 1992.

[4] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam, "Computer Recognition of Unconstrained Handwritten Numerals," *Proc. IEEE*, vol. 80, no. 7, July 1992, pp. 1,162-1,180.

[5] E.R. Dougherty, *An Introduction to Morphological Image Processing*, vol. TT9. Bellingham, Wash.: SPIE Press, 1992.

[6] P.D. Gader, A.M. Gillies, and D. Hepp, "Handwritten Character Recognition," E. Dougherty, ed., *Digital Image Processing Methods*. New York: Marcel Dekker, 1994, pp. 223-261.

[7] P.D. Gader, M. Mohamed, and J.H. Chiang, "Handwritten Word Recognition With Character and Inter-Character Neural Networks," *IEEE Trans. Sys. Man Cybernetics*, in press.

[8] P.D. Gader, M. Mohamed, and J. Chiang, "Comparison of Crisp and Fuzzy Character Networks in Handwritten Word Recognition," *Proc. North American Fuzzy Information Processing Soc. Conf.*, Puerto Vallarta, Mexico, pp. 257-266, 1992.

[9] P. Gader, M. Mohamed, and J. Chiang, "Comparison of Crisp and Fuzzy Character Neural Networks in Handwritten Word Recognition," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 3, pp. 357-364, 1995.

[10] P.D. Gader, M. Mohamed, and J. Chiang, "Fuzzy and Crisp Handwritten Alphabetic Character Recognition Using Neural Networks," *Proc. Artificial Neural Networks in Engineering*, St. Louis, Mo., pp. 421-427, Nov. 1992.

[11] A. Gillies, "Cursive Word Recognition Using Hidden Markov Models," *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1992, pp. 557-563.

[12] A. Gillies, D. Hepp, and P. Gader, "A System for Recognizing Handwritten Words," Technical Report submitted to the United States Postal Service, Office of Advanced Technology, Nov. 1992.
[13] T.K. Ho, J.J. Hull, and S.N. Srihari, "Decision Combination in Multiple Classifer Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66-75, Jan. 1994.
[14] J.J. Hull, "A Database for Handwritten Text Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 481-496, May 1994.
[15] F. Kimura, M. Shridhar, and N. Narasimhamurthi, "Lexicon Directed Segmentation—Recognition Procedure for Unconstrained Handwritten Words," *Proc. Third Int'l Workshop on Frontiers in Handwriting Recognition*, Buffalo, N.Y., May 1993, pp. 122-132.
[16] F. Kimura, M. Shridhar, S. Tsuruoka, and Z. Chen, "Context Directed Handwritten Word Recognition for Postal Service Applications," *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1992, pp. 199-214.
[17] E. Lecolinet and J. Crettez, "A Grapheme-Based Segmentation Technique for Cursive Script Recognition," *Proc. First Int'l Conf. Document Analysis and Recognition*, Saint Malo, France, Sept.-Oct. 1991, pp. 740-748.
[18] P.D. Gader, M.P. Whalen, M.J. Ganzberger, and D. Hepp, "Handprinted Word Recognition on a NIST Data Set," *Machine Vision and Its Applications*, vol. 8, pp. 31-40, 1995.
[19] C. Nohl, C. Burges, and J. Ben, "Character-Based Handwritten Address Word Recognition With Lexicon," *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1992, pp. 167-180.
[20] O. Nobuyuki, "A Threshold Selection Method From Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, Jan. 1979.
[21] P.D. Gader and M.P. Whalen, "Advanced Research in Handwritten ZIP Code Recognition," Final Technical Report to USPS Office of Advanced Technology, Dec. 1990.
[22] P.D. Gader, M. Mohamed, and J.-H. Chiang, "Segmentation-Based Handwritten Word Recognition," *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1992, pp. 215-226.
[23] *Proc. Int'l Conf. Document Analysis and Recognition*, Saint Malo, France, Oct. 1991.
[24] *Proc. Third Int'l Workshop on Frontiers in Handwriting Recognition*, Buffalo, N.Y., May 1993.
[25] *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1992.
[26] R.A. Duderstadt, "Isolated Word Recognition for Postal Address Processing," *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1990, pp. 233-247.
[27] M. Bozinovic and S. Srihari, "Off-Line Cursive Script Word Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 1, pp. 68-83, 1989.
[28] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, 1989, vol. 77, pp. 257-286.
[29] T.K. Ho, "A Word Shape Analysis Approach to Recognition of Degraded Word Images," *Proc. United States Postal Service Advanced Technology Conf.*, Washington, D.C., Nov. 1990, pp. 207-233.
[30] T.K. Ho, J.J. Hull, and S.N. Srihari, "Word Recognition With Multi-Level Contextual Knowledge," *Proc. Int'l Conf. Document Analysis and Recognition*, Saint Malo, France, Oct. 1991, pp. 905-915.
[31] P.D. Gader, J.M. Keller, R. Krishnapauram, J.-H. Chiang, and M. Mohamed, "Neural and Fuzzy Methods in Handwriting Recognition," manuscript.
[32] J.-H. Chiang, "Hybrid Fuzzy Neural Systems for Robust Handwritten Word Recognition," Univ. of Missouri, Columbia, PhD dissertation, 1995.
[33] M.A. Mohamed, "Handwritten Word Recognition Using Generalized Hidden Markov Models," Univ. of Missouri, Columbia, PhD dissertation, 1995.

The Effect of Morphological Filters on Texture Boundary Localization

J. Alison Noble

Abstract—We extend the theoretical results in [1] to two distributions and provide a quantitative comparison of 1D morphological filter edge localization with two classical types of kernel-based smoothing filters (the mean and the close relative to morphological filters, the median). Implications in the context of statistical texture segmentation are briefly discussed.

Index Terms—Low-level processing, texture analysis, mathematical morphology, image filtering, median filter.

1 Introduction

THE relationship between the two morphological filters called the *clos-opening* filter and *open-closing* filter, and the median filter has been recently discussed in [2], [1], [3]. These morphological filters share a common property with the median in that they preserve edges and have good smoothing properties, particularly for eliminating impulse-like noise. They are often used in applications for suppressing noise in preference to linear filters [4].

The difference between these filters can be seen qualitatively in Fig. 1. This figure shows a noisy step which is filtered by one dimensional morphological clos-opening, open-closing and median filters of equivalent filter size. Note that the open-closing underestimates the signal w.r.t. the median response and the clos-opening overestimates it. The median is also more "spikey" (due to oscillations—see Section 2).

The connection between the median filter and two morphological noise suppression filters called the *clos-opening* and the *open-closing* is now well understood for the case of a constant signal in additive noise. For instance, unlike the median, the clos-opening and open-closing are *idempotent* operators [5]. In particular this means that reapplying a clos-opening or open-closing does not further change the signal structure—the root signals are extracted in one pass. A median does not share this property—in general, a root signal of a nonrecursive median is obtained after several passes of the filter. The clos-opening and open-closing are dual operators and bound the median root, $\left(X_{B^n}\right)^{B^n} \leq med_\infty^n \leq \left(X^{B^n}\right)_{B^n}$ [2]. The clos-opening and open-closing are biased point estimators of a constant signal embedded in noise [1]. The average of a clos-opening and open-closing gives an unbiased point estimate of a constant signal in the presence of symmetrically distributed noise [3]. Finally, morphological filters can perform better than a median at removing impulse noise [3]. In turn, a median filter is better than a moving-average filter at this task [6].

1. Mask sizes of equivalent filters are for the median filter $\underbrace{\bullet \ldots \bullet}_{n} \circ \underbrace{\bullet \ldots \bullet}_{n}$ ($2n+1$), and for the morphological filters $\circ \underbrace{\ldots \bullet}_{n+1}$ where $\circ$ is the central pixel in the mask.

• *The author is with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, England OX2 7JS.*
E-mail: noble@robots.ox.ac.uk.

Manuscript received Dec. 2, 1993; revised Jan. 24, 1996. Recommended for acceptance by J. Daugman.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number P96010.

0162-8828/96$05.00 ©1996 IEEE