Pergamon

Pattern Recognition, Vol. 31, No. 1, pp. 41–51, 1998

0031-3203/98 $17.00+.00

PII: S0031-3203(97)00023-X

HANDWRITTEN PHRASE RECOGNITION AS APPLIED TO STREET NAME IMAGES

GYEONGHWAN KIM* and VENU GOVINDARAJU

Center of Excellence for Document Analysis and Recognition (CEDAR), Department of Computer Science, State University of New York at Buffalo, 500 Lee Entrance, Amherst, NY 14228-2567, U.S.A.

(*Received* 27 *November* 1995; *in revised form* 28 *January* 1997)

Abstract—A phrase recognition method for recognition of street name images is presented in this paper. Some of the challenges posed by the problem are: (i) patron errors, (ii) non-standardized way of abbreviating names, and (iii) variable number of words in a street name image. A neural network has been designed to segment words in a phrase, using distance between components and style of writing. Experiments show perfect word segmentation performance of 85%. Substring matching is attempted only between the main body of a lexicon entry and the word segments of an image. Efforts to reduce computational complexity are successfully made by the sharing of character segmentation results between the segmentation and recognition phases. 83% phrase recognition accuracy was achieved on a test set. © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd.

Handwriting recognition Connected components Segmentation Neural network Address interpretation

1. INTRODUCTION

Machine recognition of unconstrained handwriting is a challenging task. Recent research in recognition of general handwritten text has revealed a number of complex problems.[1] While recognition of isolated units of writing, such as a character or a word, has been extensively studied in literature,[2–4] emphasis on the next logical step of recognizing words and phrases has been lacking. One realizes the importance of recognizing phrases in applications where a computer program has to read fields containing handwritten responses. Such applications warrant segmentation of the field into its constituent words as the first processing step.

The stages in the architecture of a phrase recognition system are (i) *segmentation*, concerns extracting words from a phrase, (ii) *word recognition*, concerns handwritten word recognition algorithms with and without lexicon, and (iii) *postprocessing*, concerns use of linguistic constraints to improve recognition performance. Our focus in this paper is on the first two stages. The importance of correct word segmentation should be evident from the sequential architecture of the phrase recognition system. Correct word segmentation is of crucial importance since errors made in the segmentation stage get compounded by the time phrase recognition is performed.

Algorithms dealing with word segmentation that have been reported in the literature thus far are based primarily on analysis of geometric relationship of adjacent components in an image. Consequently, the leading and trailing ligatures of a word, often exaggerated in writings with flourish, hinder accurate segmentation. We propose a method to overcome the problem of interference between ligatures of adjacent components by extracting information from character segments.

Recognizing the street name is an integral part of an address interpretation system.[5] Street name images usually contain more than a single word and the number of words can vary. Different ways of abbreviating prefixes and suffixes also make the segmentation task difficult. Computational complexity, a serious concern while matching lexicon entries and all possible combinations of segments, is addressed by the use of knowledge of how often different characters get split and in what manner.

In this paper, we describe a recognition driven strategy for segmenting a handwritten phrase into its constituent words by encoding the author's writing style in terms of spacing using a neural network. Our approach begins with locating the bounding boxes of character segments. The center-line of each box is determined using the distribution of pixels within the box. Intervals and heights of the center lines are treated as key parameters and input to a neural network designed for locating word breaks. A lexicon is generated so that it contains only the keywords ("main" part of the street name). Matching between the lexicon and the input image (street name) is performed by a dynamic programming based word recognizer.[6]

Organization of this paper is as follows. Section 2 describes previous work on general word segmentation and recognition. Section 3 outlines general problems in the task of street name recognition and the requirements incumbent on any word recognizer that needs to be

* Author to whom correspondence should be addressed. Tel.: 716-645-6164; Fax: 716-645-6176; e-mail: gkim@cedar.buffalo.edu

extended to handle phrases. A new strategy for locating word breaks in a street name image and the methodology of matching the main part of lexicon entries and a group of segments are described in the section. Section 4 is about experiments and results. Summary of the work and some concluding remarks are presented in Section 5.

2. PREVIOUS WORK

2.1. Word segmentation

Most previous work in recognizing handwritten text assumes that words are already isolated or that isolation is trivial (e.g. words are separated by large amounts of white space),[7] or that words are written in boxes whose location is known.[8] Few published reports describe methods of separating words in unconstrained handwriting.[9] Brady showed how certain filters can highlight the word gaps in grey-level machine-printed text images.[10] The emphasis was on offering an explanation for certain psychological text-interpretation studies. The method was not tested on handwritten text. There are limited number of research reports that present a comprehensive discussion on this topic.[11–15]

The commonly adopted computational approach to word separation using spatial distance cues consists of the following steps: (i) determine the connected components in the given line, (ii) compute the distance (or gap) between pairs of adjacent components, (iii) sort the gaps in descending order of magnitude, and (iv) classify the gaps into inter-word gaps and inter-character gaps by choosing a threshold. Gaps greater than the threshold are deemed to be word separation points.

Computing the distance between adjacent components [(ii) above] is an open issue which warrants further research. The objective is to obtain an estimate of the inter-component gaps as perceived by humans. The first and most straightforward estimation method computes the horizontal distance between the *bounding boxes* of adjacent components, where the bounding box of a component is defined as the smallest rectangle enclosing the component. The second method[11] uses *run-lengths* and *Euclidean distances* between connected components and heuristics. The heuristics handle cases where adjacent components do not have sufficient overlap in the x and y direction. The third technique approximates the gaps between components by the distance between their convex hulls.[14]Fig. 1 illustrates the gaps estimated by these techniques between two connected components. The bounding box method fails to report any gap in the example [Fig. 1(a)]. The convex hull method reports a reasonable estimate of the gap in the example [Fig. 1(c)].

2.2. Word recognition

In order to recognize phrases, such as street names, we must first preview the architecture of word recognizers and look for ways of extending the architecture to handle phrases. Word recognizers traditionally require a lexicon to recognize the word in question. The lexicon is a list of possible words (phrases) that could possibly occur in that

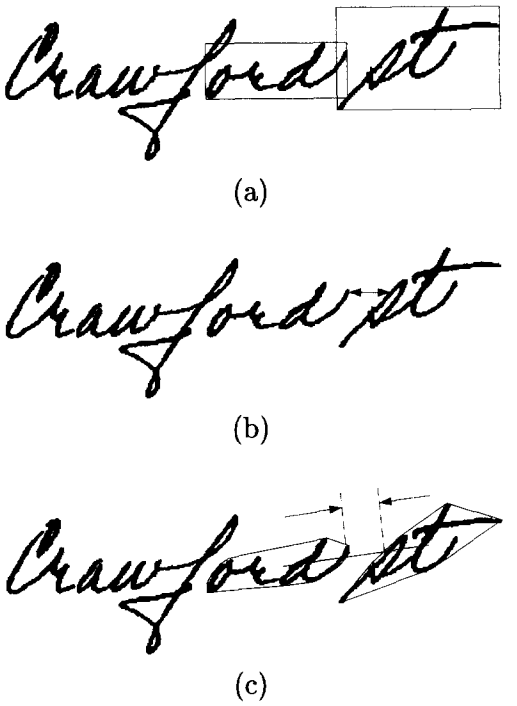


Fig. 1. Inter-component gaps using geometric relationship between components generated by (a) bounding box method, (b) run length based method, and (c) convex hull based method.

image. This lexicon is usually determined by the application at hand. For example, if the phrase in question is extracted from the legal amount of a bank check then the lexicon for each word in the phrase can be a list of about 40 words (e.g. one, two, three,..., ten, twenty,..., hundred,...). If the application is one of reading the employee name from a form of personal information, the lexicon is a list of names of all the employees. In the case of street names on mail pieces, the lexicon is a list of street names in a given ZIP code. Thus the size and nature of a lexicon can vary depending on the application.

The first task of a traditional word recognizer is to locate the character segmentation points. A segment can be a complete character or part of a character. Our method [described in detail elsewhere[6]] proceeds as follows. A segment or a combination of consecutive segments is provided to a feature extraction module. The feature extraction module turns the segments into a multi-dimensional vector based on global and local features of the input segment(s). In the recognition module, comparison between the feature vector and code words of a character cluster (built during the training phase) is performed. The procedure is repeated for all segments and lexicon entries and the best matching entry is obtained using dynamic programming.

3. METHODOLOGY

Sample images of phrases extracted from street lines of addresses are shown in Fig. 2. A simplistic approach where street name images are treated as single words by

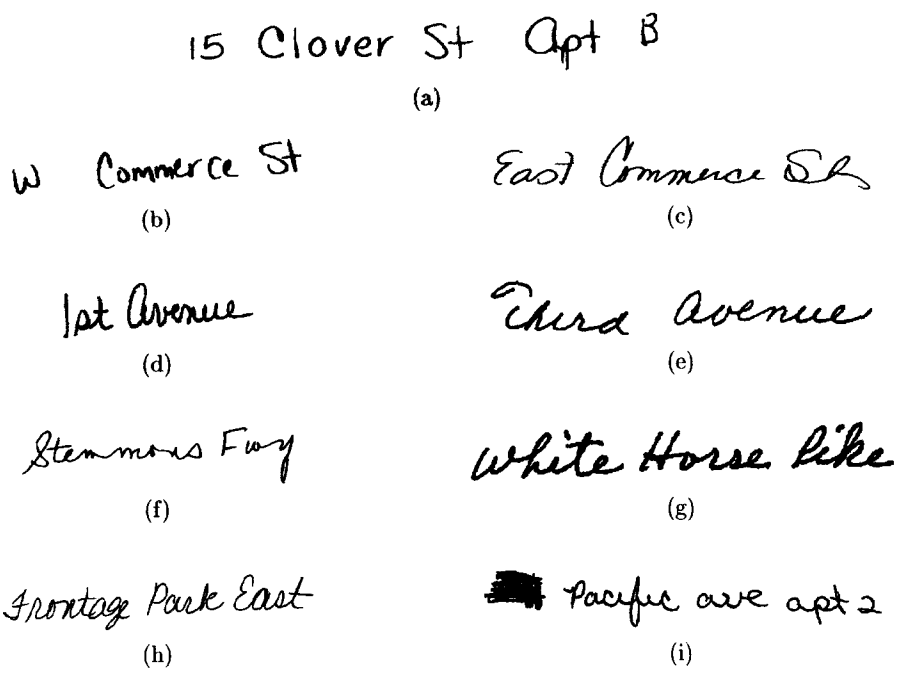


Fig. 2. Examples of street name images: (a) street line address, (b)–(c) the same street name with different pre-directionals, (d)–(e) numeric street name, (f)–(g) suffix, (h) directional suffix, and (i) miscellaneous addons.

ignoring spaces between words in both images and lexicon entries cannot deal with all images due to their diverse ways of representation.

A system, which performs matching between main bodies of a street name image and lexicon entries, is described in this paper. Fig. 3 illustrates an overview of the system. Inputs to the system are a binary image of a phrase and a lexicon. The phrase in the example has three words: a prefix ("S"), followed by the "main" body or name ("Michigan") and a suffix ("Ave"). The lexicon also contains "main" part only. The pre-processing step involves noise removal, slant correction and smoothing. Objective of word segmentation is to return several sub-images, where each sub-image contains a single word. As shall be seen in the course of this paper, because of the design of our method, it is alright for the word segmentation module to return sub-images which contain parts of a word. In the example of Fig. 3, the desired output is three sub-images containing the words: "S", "Michigan", and "Ave", respectively. Recognition concerns the matching between the lexicon and the "main" part of the image. Based on the goodness of the matching, a ranked list of the lexicon entries is obtained as the output.

In the subsequent sections we present an elegant phrase handling methodology by extending an existing word recognition scheme that operates on single words, and developing a new word segmentation method. There are two major parts to the methodology: segmentation and recognition.

3.1. Segmentation

Word segmentation methods based on spatial cues alone (as described in Section 2.1) have limited success for two reasons. First, handwriting does not necessarily adhere to the rule of placing larger gaps between words than between characters. Second, the perceived space between components cannot be easily estimated by a 1-dimensional scalar metric.

Deriving a computational methodology for word segmentation is quite complex. Humans use several cues to perform word separation: (i) spatial separation between components, (ii) presence of punctuation marks, (iii) presence of upper case characters following a string of lower case characters, (iv) transition between numerals and alpha characters, and (v) actual recognition of words prior to word separation. Among these, the cue of spatial separation is most commonly used.

To overcome the drawbacks of conventional methods, a word segmentation method, which employs some of the cues that humans use and additional information such as author's writing style in terms of spacing, is introduced in this section. The style is captured by characterizing the variation of spacing between adjacent characters as a function of the corresponding characters themselves. The notion of expecting greater space between characters with leading or trailing ligatures is encoded into the segmentation scheme. Furthermore, transitions from a string of lower case characters to an upper case character are utilized in the scheme. Such cues are extracted from character segmentation results, not from only connected components.

Our character segmentation algorithm locates segmentation points between characters in a word and is based on the following two assumptions: (i) the number of segments per character must be at most 4, and (ii) all touching characters should be separated. Segmentation points are determined using features like ligatures and

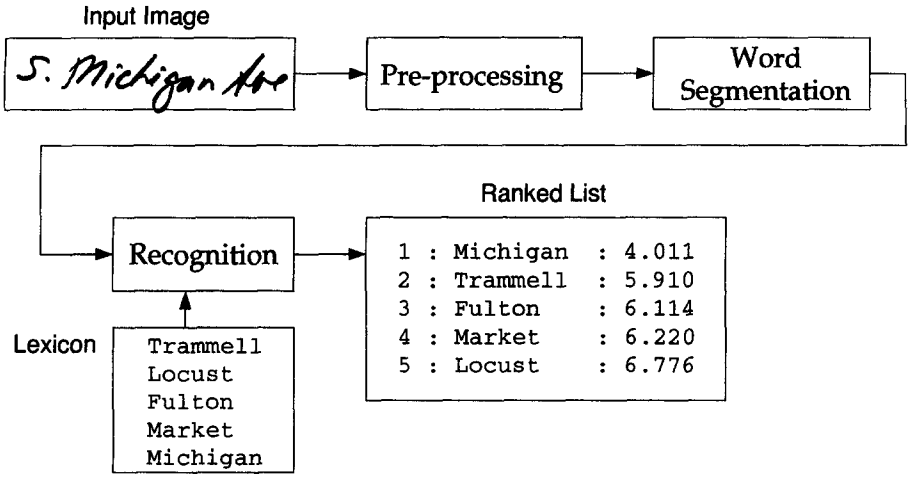


Fig. 3. System overview: Input is a phrase of multiple words and a lexicon of single words (extracted from the main part of a street name). Output is a ranked lexicon sorted in the order of "goodness" of match between the image and lexicon entries.

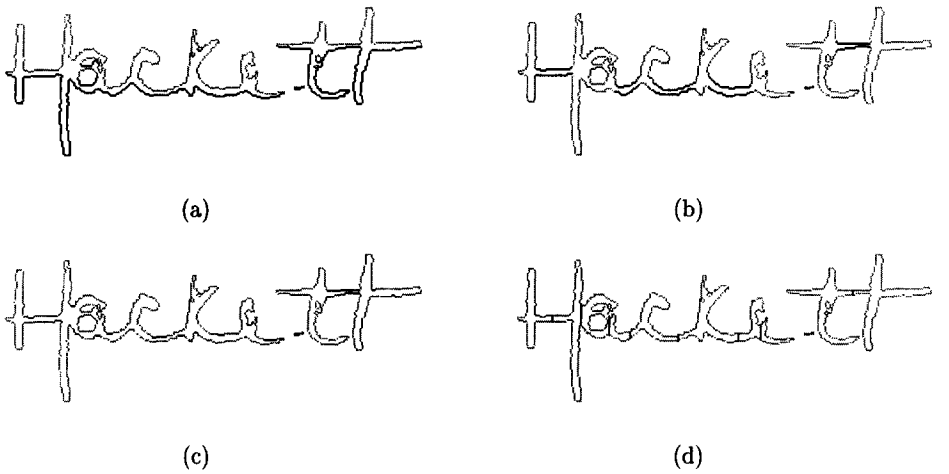


Fig. 4. Character segmentation: (a) splitting upper- and lower- contours, (b) ligatures, (c) concavities, and (d) segmentation points.

concavities (Fig. 4). If the distance between y-coordinates of the upper-half and lower-half of the outer contour for a x-coordinate is less than or equal to the average stroke width, then the x-coordinate is marked as a ligature [Fig. 4(b)]. Concavities in the upper contour and convexities in the lower contour are located by examination of slope changes in contours [Fig. 4(c)]. Based on the ligatures and concavity features, final segmentation points are determined [Fig. 4(d)].

To explain the word segmentation procedure, let us consider the input street name image of Fig. 5(a). It is to be noted that the gap between "n" and "A", is significantly less than gaps between "r" and "e", and "n" and "t". The input image is converted into a contour representation and the slant angle is estimated and corrected [Fig. 5(b)].[16] The original character segmentation algorithm segments the image into characters and subcharacters [Fig. 5(c)].

Four reference lines are used to extract the spatial features: upper bound, upper half, lower half, and lower bound.[11,17] However, all these lines cannot be accurately located, specially when dealing with phrases (as opposed to single words). Therefore, we choose to define two reference lines: *base line* and *middle line* [Fig. 5(d)]. The base line is the best-fit line of concavities in the lower contours. The middle line is the best-fit line of y-coordinates of the center of mass of each character segment. Based on the two reference lines, segments are examined. Rules are developed to determine which segments can be merged, if at all. For example, in Fig. 5(d), a horizontal stroke over the vertical stroke of letters "T" and "r" is merged into the vertical stroke of "T". Fig. 5(e) shows bounding boxes after the merge. A vertical line is located based on the distribution of pixels inside the box as shown in Fig. 5(f). The heights of the bounding boxes and the intervals between two adjacent

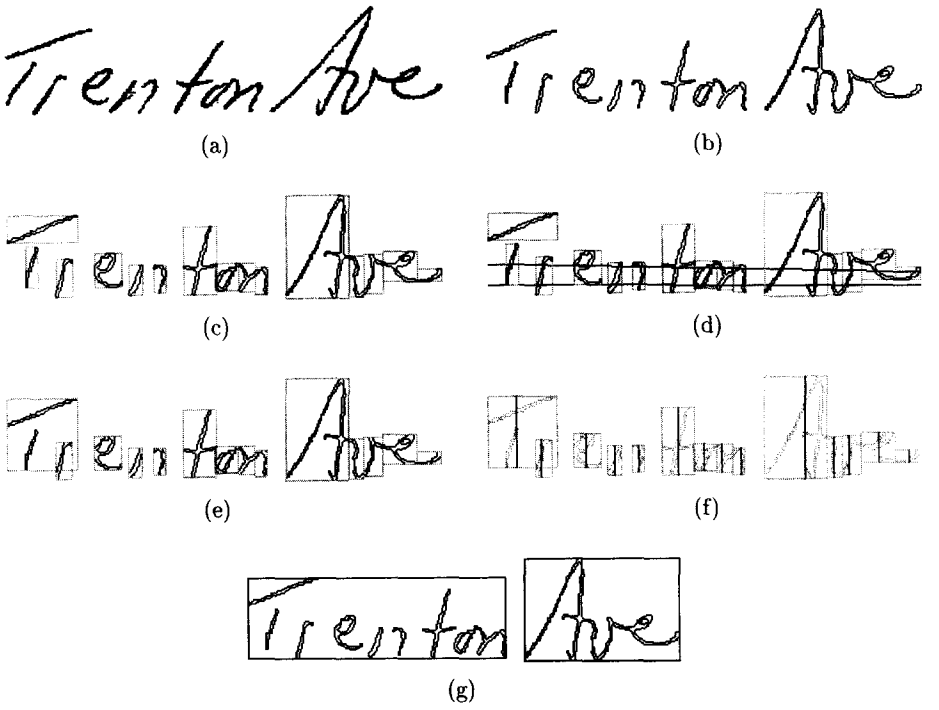


Fig. 5. Character segmentation: (a) an input street name image, (b) slant normalized, (c) character based segmentation, (d) base line and middle line, (e) merged segments, (f) center line of each segments, and (f) word segmentation result.

boxes characterize the author's writing style in terms of inter-character and inter-word spacing.

Now the problem is one of extracting information which can represent the above mentioned cues for word segmentation. Let B_i and B_{i+1} denote a pair of adjacent bounding boxes after the merging, and let C_i be the center line of the bounding box B_i. Let I_i be the interval between C_i and C_{i+1} and H_i be the height of B_i and defined as follows:

$$I_i = X(C_{i+1}) - X(C_i), H_i = \begin{cases} Y_u(C_i) - \text{Baseline}(X(C_i)) & \text{if Baseline}(X(C_i)) \geq Y_b(C_i), \\ Y_u(C_i) - Y_b(C_i) & \text{otherwise,} \end{cases} \quad (1)$$

where $X(C_i)$, $Y_u(C_i)$, and $Y_b(C_i)$ represent the x-coordinate, upper and bottom y-coordinates of a center line C_i. Baseline$(\cdot)$ returns the y-coordinate of C_i at the base line.

We adopt a simple neural network to determine the segmentation points. The neural network has eight inputs units, four hidden units and one output unit as shown in Fig. 6. Input parameters to the neural network are properties of bounding boxes described above. H_is and I_is are normalized and averages and standard deviations are computed. Following eight parameters are selected for inputs of the neural network: two consecutive intervals (I_i and I_{i+1}) and heights (H_i and H_{i+1}), average of intervals (I_{av}) and heights (H_{av}), flag for inter-component transition (F_{tr}), and flag for the last segment (F_l). It should be noted that the height information as well as interval information of segments are included to encode the author's writing style. An experiment was performed to justify the input selection and results are shown in Section 4. Fig. 5(g) shows the word segmentation result of the example image using the method. Fig. 7 shows a sample image and the corresponding patterns for testing and training.

3.2. Recognition

Generating lexicon for the street name recognition task is somewhat involved. As can be seen in Fig. 2, there are several variations of prefixes and suffixes (e.g. "*Fwy*"

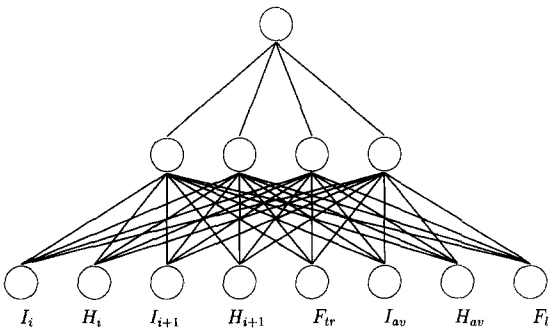
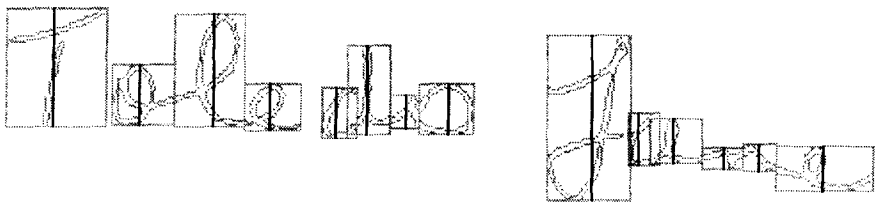


Fig. 6. Structure of the neural network for word segmentation.



(a)

i	I_i	H_i	I_{i+1}	H_{i+1}	F_{tr}	I_{av}	H_{av}	F_l	**Truth**
1	0.714	0.469	0.616	0.922	1.000	0.479	0.497	0.000	**0.000**
2	0.616	0.922	0.470	0.391	0.000	0.479	0.497	0.000	**0.000**
3	0.470	0.391	0.551	0.406	0.000	0.479	0.497	0.000	**0.000**
4	0.551	0.406	0.260	0.734	1.000	0.479	0.497	0.000	**0.000**
5	0.260	0.734	0.324	0.281	0.000	0.479	0.497	0.000	**0.000**
6	0.324	0.281	0.357	0.422	0.000	0.479	0.497	0.000	**0.000**
7	0.357	0.422	1.000	1.000	0.000	0.479	0.497	0.000	**0.000**
8	1.000	1.000	0.389	0.406	1.000	0.479	0.497	0.000	**1.000**
9	0.389	0.406	0.292	0.375	1.000	0.479	0.497	0.000	**0.000**
10	0.292	0.375	0.422	0.188	0.000	0.479	0.497	0.000	**0.000**
11	0.422	0.188	0.292	0.234	0.000	0.479	0.497	0.000	**0.000**
12	0.292	0.234	0.535	0.266	0.000	0.479	0.497	0.000	**0.000**
13	0.535	0.266	0.000	0.000	0.000	0.479	0.497	1.000	**0.000**

(b)

Fig. 7. A sample image and training patterns: (a) result of character segmentation and merging, and (b) patterns for the neural network. The last column, Truth, is provided when the network is trained.

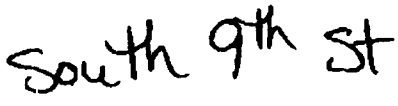
instead of "*Freeway*") that can appear in an image. Moreover, incorrect suffixes (and their omission altogether) introduced by patrons are common (e.g. "*Street*" instead of "*Avenue*"). One possible solution would be to expand each lexicon entry in all conceivable ways (e.g. "*Main Street*" to "*Main*", "*Main Street*", "*Main St*"). However, this approach increases the lexicon size, consequently, the matching process becomes inefficient. A particularly difficult instance to be noted in this regard is that of numeric street names (Fig. 8) wherein to account for a single street name the lexicon must contain 18 different forms of the name in which it can occur in a real image.

Additional complexities can occur due to the presence of extraneous words such as "*apartment*" and "*suite*" appearing at the end of a street name [Fig. 2(i)]. Certainly, the lexicon entries do not anticipate their presence. Furthermore, crossing out of writing [Fig. 2(i)] is an additional source of concern.

The final goal is to find the best match between a word in the lexicon and the "main" word in the street name image. We have described in detail a lexicon driven handwritten word recognition algorithm based on dynamic matching elsewhere to handle isolated words.[6] In the matching procedure, comparisons between feature vectors of several possible combinations of segments and reference feature vectors of codewords are made to find the best match. Since codewords are trained at the character level and a character can be composed of up to four segments, a segment or a combination of segments is compared to the codewords of reference characters within a permissible window in the first phase of matching. For each match, the minimum distance value of each comparison is retained.

In the training phase of the word recognizer, segmentation statistics of each character are obtained. The segmentation statistics represent the possible ways in which a training character image can be split into segments.[18] However, unlike Chen *et al.*,[18] where segmentation statistics are used to determine the transition probability between segments of a character, we use the statistics to define the size of the matching window. It has been proven that the statistical information can be used to improve recognition accuracy as well as speed.[6] Table 1 defines variable duration for each character based on the statistics obtained during the training phase.[6]

Ways to minimize the computational complexity are sought by maximizing the advantage of the lexicon



(a)

S9thSt	SNinthStreet
South9thSt	SouthNinthStreet
9thSt	NinthStreet
SNinthSt	S9th
SouthNinthSt	South9th
NinthSt	9th
S9thStreet	SNinth
South9thStreet	SouthNinth
9thStreet	Ninth

(b)

Fig. 8. An example of lexicon expansion: (a) a numeric street name and (b) expanded lexicon entries for the image.

Table 1. Variable duration (min:max): the lower limit as well as the upper limit of the number of segments are specified and used for determining the size of matching window in the matching procedures.

character	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
duration	1 : 2	1 : 1	1 : 3	1 : 3	1 : 3	1 : 3	1 : 3	1 : 3	1 : 2
character	*9*	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
duration	1 : 2	1 : 3	1 : 3	1 : 3	1 : 3	1 : 3	1 : 3	1 : 3	1 : 3
character	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*
duration	1 : 2	1 : 3	1 : 3	1 : 2	2 : 4	1 : 3	1 : 3	1 : 3	1 : 3
character	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
duration	1 : 3	1 : 3	1 : 3	1 : 3	1 : 3	2 : 4	1 : 3	1 : 3	1 : 3

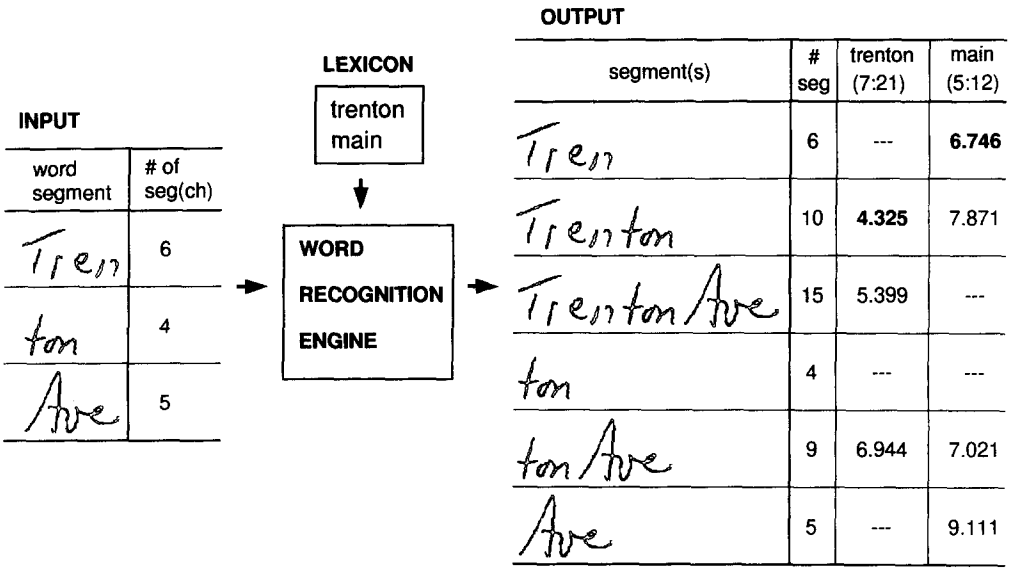


Fig. 9. Matching word segments with lexicon entries.

driven approach. To explain the matching procedure in this application, let us assume that the example image shown in Fig. 5(a) is split into three segments as shown in Fig. 9 instead of two [Fig. 5(g)]. Let us also assume that there are two lexicon entries, "trenton" and "main".

Each word segment is submitted to the word recognition engine along with the number of character segments. Matching between a word segment (or a combined word segments) and the lexicon entries is performed for all possible combinations of the word segments keeping the matching confidence. The final matching result for each lexicon entry is represented by the best result saved. In Fig. 9 the matching confidence value of "trenton" is 4.325 and "main" is 6.746.

A way of minimizing the computational complexity by extending the concept introduced in the word recognition engine is considered. In this paper, we define the lower and upper bounds of variable duration for each lexicon

entry (i.e. a word) using the following:

$$\text{lex}_{\min}(i) = \sum_j \text{dur}_{\min}(\text{lex\_entry}[j]) \quad \text{for } j = 0 \text{ to } N_c(i) - 1,$$
$$\text{lex}_{\max} = \sum_j \text{dur}_{\max}(\text{lex\_entry}[j]) \quad \text{for } j = 0 \text{ to } N_c(i) - 1, \tag{2}$$

where $N_c(i)$ represents number of characters in ith entry in the lexicon, lex_entry[·] represents character string of the lexicon entry, $\text{dur}_{\min}(\cdot)$ and $\text{dur}_{\max}(\cdot)$ return the lower and upper bounds of variable duration of a character defined in the Table 1, and $\text{lex}_{\min}(i)$ and $\text{lex}_{\max}(i)$ are the lower and upper bounds of variable duration of ith lexicon entry.

Using equation (2) the lower and upper bounds of variable duration of the lexicon entries are defined as following:

trenton : min $= t(1) + r(1) + e(1) + n(1) + t(1) + o(1) + n(1) = 7$
max $= t(3) + r(3) + e(3) + n(3) + t(3) + o(3) + n(3) = 21$
main : min $= m(2) + a(1) + i(1) + n(1) = 5$
max $= m(4) + a(3) + i(2) + n(3) = 12$

The following rule is used to skip unnecessary computation.

for each lexicon entry
if ($WS_{\text{num}} < \text{lex}_{\min}$ or $WS_{\text{num}} > \text{lex}_{\min}$) *reject*
else perform matching

where, WS_{num} is the number of character segments for the word segment under testing. In the example, four cases (marked with "—" at the output table) are rejected based on this rule at an early stage.

Furthermore, if a part of a word segment has been compared with a particular character in a lexicon entry, the same comparison against the same character is avoided inside the word recognition engine by saving the matching results.

4. EXPERIMENTS AND RESULTS

Experiments of extension of the recognition algorithm to street names follow. Street name images collected from live mail pieces were used for training and testing the neural network. 518 images were used for the training. 7802 patterns were collected by visual examination with desirable word segmentation points marked manually.

Table 2. Word segmentation performance

	No. of images	Perfect segmentation	Over-segmentation	Error
Set 1	518	446 (86.1%)	26 (5.0%)	46 (8.9%)
Set 2	107	88 (82.2%)	8 (7.5%)	11 (10.3%)

4.1. Word segmentation

The network was tested on two sets of data: the training set, and a new set of 107 images comprising of failure cases reported by the present handwritten address interpretation system at CEDAR.[5] Table 2 shows the word segmentation results on two sets of images. 82–86% test images are perfectly segmented. Oversegmentation cases (5.0–7.5%) can be actually classified as correct segmentation since the recognition procedure will combine the broken parts anyway. This gives an effective segmentation rate of 91%. The segmentation performance is better than the numbers reported using conventional methods.[11,19]

Fig. 10(a)–(d) show examples of perfect segmentation, given the tolerance of the recognition methodology that follows. These may be considered to be errors if alternate recognition methods were to follow. Fig. 10(e)–(g) are typical errors discovered in the test.

4.2. Recognition

Among the 107 street name images in the testing set, 11 images which were classified as word segmentation error, were excluded from the recognition phase. Table 3 shows the test results in terms of recognition accuracy and speed using three different methods. Method 1 uses the word recognizer and an expanded lexicon by regard-

Table 3. Recognition performance S: Success; F: Failure – the numbers in ( ) represent the number of cases resulted in better recognition confidence comparing to that of Method 1.

		Method 1		Method 2		Method 3	
Accuracy		S	F	S	F	S	F
		79	17	83 (60)	13	81 (56)	15
time (msec)	image	115.0		89.0		122.8	
	recog	523.8		1087.7		581.6	
	total	638.8		1176.7		704.4	

Fig. 10. Examples of segmentation results from our system: (a)–(d) segmentation points perfectly located, (e)–(g) failures, and (h) over-segmentation cases.

ing the entire street name block as a unit. Method 2 uses the word segmentation method described in this paper to locate word breaks, and word segments are submitted to the word recognizer with the lexicon containing only the "main" part of the names. Method 3 uses the word segmentation method described in this paper. The difference between Method 2 and Method 3 lies in the way in which they use character segmentation results. In Method 2, character segmentation is performed twice, once in the beginning to collect information to feed to the neural network and once during recognition of word segments. In Method 3, character segmentation results obtained during word segmentation are used during recognition as well. It should be noted that the ratio of lexicon sizes used in Method 1 against Method 2 and Method 3 is 4.6:1. Average time (on Sparc10) consumed by each method is shown in the lower half of Table 3. It should be noted that the number of cases resulted in better recognition confidence comparing to that of Method 1 [the numbers in ( ) in Table 3] is significantly increased, as well as increase of total number of correctly recognized cases.

Table 4. Word segmentation performance with different input selection for the neural network

	Network	Perfect	Over-seg.	Error
Set 1 (518)	Proposed method	446 (86.1%)	26 (5.0%)	46 (8.9%)
	Temp_net1	398 (76.8%)	35 (6.8%)	85 (16.4%)
	Temp_net2	393 (75.9%)	29 (5.6%)	96 (18.5%)
Set 2 (107)	Proposed method	88 (82.2%)	8 (7.5%)	11 (10.3%)
	Temp_net1	81 (75.7%)	8 (7.5%)	18 (16.8%)
	Temp_net2	82 (76.6%)	6 (5.6%)	19 (17.8%)

4.3. *Justification of input selection*

To prove the effectiveness of the features selected for the word segmentation some tests were performed by limiting the number of input parameters to the neural network. Two simple neural networks were created with a 5–3–1 structure: Temp_net1 and Temp_net2. Temp_net1 uses features I_i, I_{i+1}, F_{tr}, I_{av}, and F_l and Temp_net2 uses features I_i, H_i, F_{tr}, I_{av}, and H_{av} as inputs. Temp_net1 is to prove the effectiveness of height information and Temp_net2 is to prove the effectiveness of parameter selection from two consecutive character segments. As we can see in the Table 4, lack of height information (Temp_net1) and use of an interval and height information from a single segment (Temp_net2) cause a significant drop in segmentation performance.

5. SUMMARY

In this paper, we have introduced a strategy for phrase recognition as applied to street name images. A trainable word segmentation algorithm using a neural network was designed to overcome the drawbacks of conventional methods which use distance metrics of components in an image and are easily affected by ligatures of adjacent components. Effectiveness of feature selection for the word segmentation (height and interval information of center-lines of character segments and the parameter selection from two consecutive character segments) was established. The features capture the author's writing style as a function of spacing between adjacent characters. The concept of variable duration of characters for word recognition is successfully expanded to phrase recognition.

Acknowledgements—The authors would like to thank Uma Mahadevan, CEDAR, for the discussion on different distance metrics presented in Section 2. Furthermore, Uma Mahadevan was instrumental in bringing the importance of word separation in text recognition applications to our attention.

REFERENCES

1. V. Govindaraju, R. K. Srihari and S. N. Srihari, Handwritten text recognition, in *Proc. Conf. on Document Analysis System*, Kaiserslautern, Germany, 157–171 (1994).
2. J. C. Simon, Off-line cursive word recognition, *Proc. IEEE* **80**(7), 1150–1161 (1992).
3. M. Gilloux, M. Leroux and J.-M. Bertille, Strategies for handwritten words recognition using hidden Markov models, in *Int. Conf. on Document Analysis and Recognition*, Tsukuba Science City, Japan, 299–304 (1993).
4. F. Kimura, M. Shridhar and N. Narasimhamurthi, Lexicon directed segmentation—Recognition procedure for unconstrained handwritten words, in *The Third Int. Workshop on Frontiers in Handwriting Recognition*, Buffalo, New York, 122–131 (1993).
5. V. Govindaraju, A. Shekhawat and S. N. Srihari, Interpretation of handwritten addresses in U.S. mail stream, in *The Third Int. Workshop on Frontiers in Handwriting Recognition*, Buffalo, New York, 197–206 (1993).
6. G. Kim and V. Govindaraju, Handwritten word recognition for real-time applications, in *Third Int. Conf. on Document Analysis and Recognition* (ICDAR-95), Montreal, Canada, 24–27 (1995).
7. H. Baird and K. Thompson, Reading chess, *IEEE Trans. Pattern Analysis Mach. Intell.* **12**, 552–559 (1990).
8. R. O. Duda and P. E. Hart, Experiments in recognition of hand-printed text, in *AFIPS Conf. Proc.*, 1139–1149 (1968).
9. E. Cohen, J. Hull and S. N. Srihari, Understanding handwritten text in a structured environment: Determining ZIP codes from addresses, *Int. J. Pattern Recognition Artif. Intell.* **5**(1/2), 221–264 (1991).
10. M. Brady, Toward a computational theory of early visual processing in reading, *Visible Language* **15**(2), 138–215 (1981).
11. G. Seni and E. Cohen, External word segmentation of off-line handwritten text lines, *Pattern Recognition* **27**(1), 41–52 (1994).
12. M. D. Garris, Unconstrained handprint recognition using a limited lexicon, in *Proc. SPIE Symp. on Electronic Imaging Science and Technology* (Document Recognition), San Jose, California, Vol. 2181, 36–46 (1994).
13. M. J. Ganzberger, R. M. Rovner, A. M. Gillies and D. J. Hepp, Matching database record to handwritten text, in *Proc. SPIE Symp. on Electronic Imaging Science and Technology* (Document Recognition), San Jose, California, Vol. 2181, 66–75 (1994).
14. U. Mahadevan and R. C. Nagabushnam, Gap metrics for word separation in handwritten lines, in *Third Int. Conf. on Document Analysis and Recognition* (ICDAR-95), Montreal, Canada, 124–127 (1995).
15. J. C. Simon, O. Baret and N. Gorski, A System for the recognition of handwritten literal amounts of checks, in *Proc. Conf. on Document Analysis System*, Kaiserslautern, Germany, 135–155 (1994).
16. G. Kim and V. Govindaraju, Efficient chain code based image manipulation for handwritten word recognition, in *Proc. SPIE Symp. on Electronic Imaging Science and Technology* (Document Recognition III), San Jose, California, Vol. 2660, 262–272 (1996).
17. R. M. Bozinovic and S. N. Srihari, Off-line cursive script word recognition, *IEEE Trans. Pattern Analysis Mach. Intell.* **11**(1), 68–83 (1989).
18. M.-Y. Chen, A. Kundu and J. Zhou, Off-line handwritten word recognition using a hidden Markov model type stochastic network, *IEEE Trans. Pattern Analysis Mach. Intell.* **16**(5), 481–496 (1994).
19. U. Mahadevan and S. N. Srihari, Hypotheses generation for word separation in handwritten lines, in *Fifth Int. Workshop on Frontiers in Handwriting Recognition* (IWFHR V), U.K., 453–456 (1996).

About the Author—GYEONGHWAN KIM received the B.S. (cum laude) and the M.S. in Electronics Engineering from the Sogang University, Korea, in 1984 and 1986 respectively. He received the Ph.D. degree in Electrical and Computer Engineering from the State University of New York at Buffalo, in 1996. From 1986 to 1991 he was a researcher in the Goldstar (currently LG) Precision Technology Institute (Korea). He is currently with the Center of Excellence for Document Analysis and Recognition (CEDAR) at SUNY Buffalo as a Research Scientist. His research interests include in the area of handwriting recognition, document analysis, neural networks, and human-computer interface.

About the Author — VENU GOVINDARAJU is the Associate Director of the Center of Excellence for Document Analysis and Recognition at SUNY Buffalo. He is the Project Manager and Senior Research Scientist for the CEDAR Handwritten Address Interpretation Project. He holds the Research Assistant Professorship in the Department of Computer Science at SUNY Buffalo. Govindaraju received his B.Tech. (Hons.) in Computer Science from the Indian Institute of Technology at Kharagpur in 1986, and his M.S. and Ph.D. in Computer Science from SUNY Buffalo in 1988 and 1992 respectively. Dr Govindaraju has co-authored more than 50 research papers in various conferences and journals in the fields pattern recognition and computer vision.