Chaincode Contour Processing for Handwritten Word Recognition

S. Madhvanath, *Member, IEEE Computer Society*,
G. Kim, *Member, IEEE Computer Society*, and
V. Govindaraju, *Senior Member, IEEE*

Abstract—Contour representations of binary images of handwritten words afford considerable reduction in storage requirements while providing lossless representation. On the other hand, the one-dimensional nature of contours presents interesting challenges for processing images for handwritten word recognition. Our experiments indicate that significant gains are to be realized in both speed and recognition accuracy by using a contour representation in handwriting applications.

Index Terms—Image processing, chain code, handwriting recognition, preprocessing, segmentation, feature extraction.

1 Introduction

In a digitized image, all pixels that have foreground intensity level and are connected to each other by a path of pixels form a connected component. In general, recognition applications deal with connected components instead of pixels to minimize operations on a large sized image array. Several methods have been adopted for the representation of connected components to optimize computer resources and processing time. *Line adjacency graph* (LAG) is a technique employing run length encoding to represent objects [1]. Symmetric Axis Transformation (SAT), also called medial axis, is a representation that goes orthogonal to the boundary, while the conventional description traverses the boundary [2]. *Chain code* is a linear structure that results from quantization of the trajectory traced by the centers of adjacent boundary elements in an image array [3], [4]. Each data node in the structure represents one of eight grid nodes that surround the previous data node. We present the chaincode-based image representation and manipulation techniques used in most handwriting recognition systems. The techniques include methods for preprocessing, segmentation, and feature extraction. Preprocessing deals with noise removal, slant correction, and smoothing of contours (Section 2.1). Extremas in the chaincoded contours are used to find reference lines and word level features (Section 2.2). Section 3 describes experimental results that support our claim that image manipulation in chaincode is efficient and desirable.

2 Chaincode Processing

The chaincode representation is proceduraly described as follows: Given a binary image, it is scanned from top to bottom and right to left and transitions from white (background) to black (foreground) are detected. The contour is then traced counterclockwise (clockwise for interior contours) and expressed as an array of contour elements (Fig. 1a). Each contour element represents a pixel on the contour, and contains fields for the x, y coordinates of the pixel, the slope or direction of the contour *into* the pixel, and auxiliary information such as curvature. The slope convention used by the algorithms described is as shown in Fig. 1b. The techniques for processing chaincode can be applied at three different levels: 1) image level, 2) word Level, and 3) character level.

2.1 Image Level Processing

Techniques at the image level are of general purpose, namely: 1) noise removal, 2) slant correction, and 3) smoothing.

1. **Noise removal:** Scanning devices and transmission media introduce noise during the image capture process. Such noise can be easily identified in chaincoded contours by computing the size of connected components. Components which have a small size with respect to the average stroke width (defined in Section 2.3, item 2) and/or have "odd" shapes are removed.
2. **Slant angle estimation and correction:** Handwritten words often have a slant which needs to be corrected before segmentation and recognition can commence (Fig. 2). Vertical line elements from contours are extracted by tracing chain code components using a pair of one-dimensional filters. Each filter is an eight-element array of different weights used for detecting vertical lines having opposite directions. A convolution operation between the filter and five consecutive components is applied iteratively by sliding the filter one component at a time. Coordinates of the start and end points of each line element extracted provides the slant angle. Global slant angle is taken as the average of all angles of all the line elements. The angles are given weights proportional to the length of the line elements in the vertical direction.

   Some researchers [6], [7], [8], [9] have demonstrated recognition methods that do not use slant correction. In such methods, compensation for slant must be made during the training process where word exemplars of a variety of slants must be used. Table 1 shows the advantage of preprocessing when the word recognizer described in [10] is used. Yacoubi et al. [11] also favor a separate preprocessing stage that involves correction for word slant.
3. **Smoothing contours:** Smoothing eliminates small blobs on a contour and illegal combinations of chain code slopes introduced during slant correction. An illegal combination is one where the following of the chaincode slopes becomes ambiguous. In a valid chaincode, the slope of a pixel uniquely determines the position of the next pixel. All groups of three adjacent pixels are analyzed and classified into different types based on the adjustment warranted to make them valid. The adjustment consists of changing the slopes and coordinates of certain pixels.

2.2 Word Level

Techniques that are useful in recognition of words are described in this subsection: 1) determination of upper and lower contours of the word, 2) determination of significant local extrema on the contour, 3) determination of reference lines, and 4) determination of word length.

1. **Upper and lower contours:** Division of an exterior contour into upper and lower parts (Fig. 3) involves the detection of two "turnover" points on the contour—the points at which the lower contour changes to upper contour and vice versa. Given that the exterior contour is traced in the counterclockwise direction, the upper

- S. Madhvanath is with IBM Almaden, 650 Harry Road, San Jose, CA 95120.
- G. Kim is with the Department of Electrical Engineering, Sogang University, C.P.O. Box 1142, Korea.
- V. Govindaraju is with CEDAR, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14228. E-mail: govind@cedar.buffalo.edu.

*Manuscript received 17 Aug. 1998; revised 21 Apr. 1999.
Recommended for acceptance by R. Plamondon.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107988.*

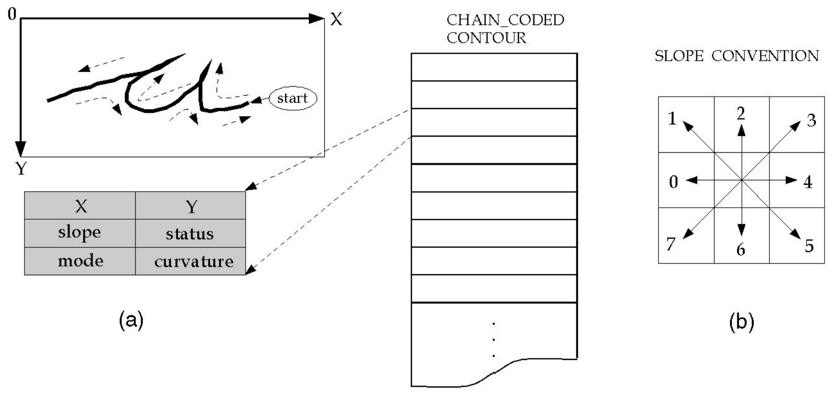

Fig. 1. Chain code contour representation: (a) contour element, (b) slope convention. *Data* field in the array contains positional and slope information of each component of the traced contour. Properties stored in the *information* fields are: coordinates of bounding box of a contour, number of components in the corresponding data fields, area of the closed contour, and a flag which indicates whether the contour is interior or exterior.

contour runs predominantly from right to left and the lower predominantly from left to right. The left end of the leading ligature and the right end of the ending ligature constitute turnover points.

2. **Local Contour Extrema:** Local extrema represent a powerful abstraction of the shape of the word. Local Y-extrema is simply derived from the y-coordinates of the pixels on the chaincoded contour. The primary challenge is the detection and rejection of spurious extrema arising from irregularities in the contour. Heuristic filters are used to eliminate spurious extrema. These include checks on the distance of the detected extremum from previous extrema detected, spatial postion relative to other extrema, the slope of the contour in the neighborhood of the extremum, and so forth. The majority of spurious extrema in discrete writing are due to ends of strokes used to form characters and generally occur in the middle zone of the word. Fragmentation in the binary image due to suboptimal binarization or poor resolution often leads to broken strokes and spurious extrema.

Since the upper contour flows effectively from right to left, left to right excursions of the contour must be followed by right to left retraces, and this leads to additional extrema being detected. Extrema found on such uncharacteristic excursions must be discarded lest they lead to spurious ascenders. A similar problem arises with right-to-left stretches of the lower contour in the context of spurious descenders.

Let us refer to extrema on such uncharacteristic excursions of the upper and lower contours as being directionally invalid. Directional validity is conveniently determined from the slopes of the contour elements into and out of the extremum and is sufficient to discard most spurious extrema.

We describe here an algorithm based on the observation that if every extremum on the upper (lower) contour was labeled with the symbols "i" for invalid and "v" for valid with respect to contour direction, the strings of extrema labels generated may be expressed in terms of a simple context free grammar defined by: $S \rightarrow Sv\ vS\ iSi\ vSv\ \lambda$, where λ is the empty string.

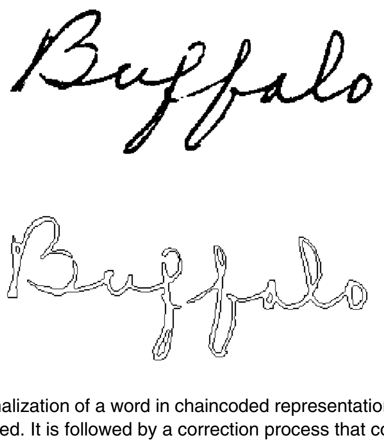

Fig. 2. Slant normalization of a word in chaincoded representation: First, the angle of slant is estimated. It is followed by a correction process that consists of a shear transformation that shifts all the pixels to compensate for the slant angle.

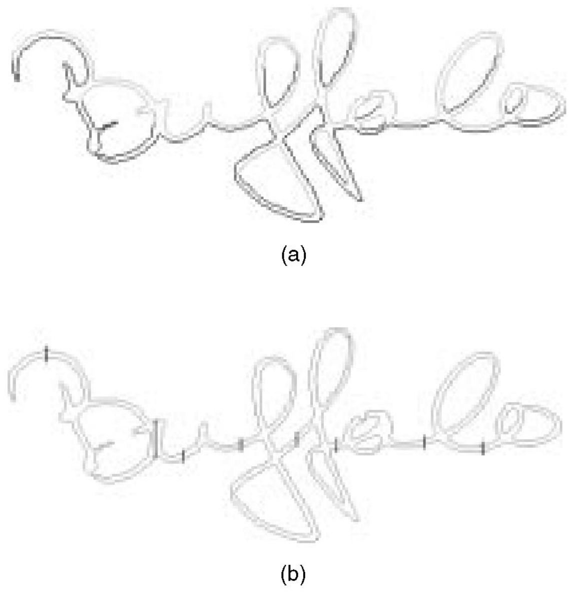

Fig. 3. (a) Splitting a contour to upper and lower parts. The upper contour is shown in dotted lines and the lower contour is shown in bold lines. (b) Character segmentation points are potentially found at the central part of ligatures; a segmentation point is found between "B" and "u" by heuristic rules that cut at points where the maxima of the upper contour is in close proximity to the minima of a lower contour.

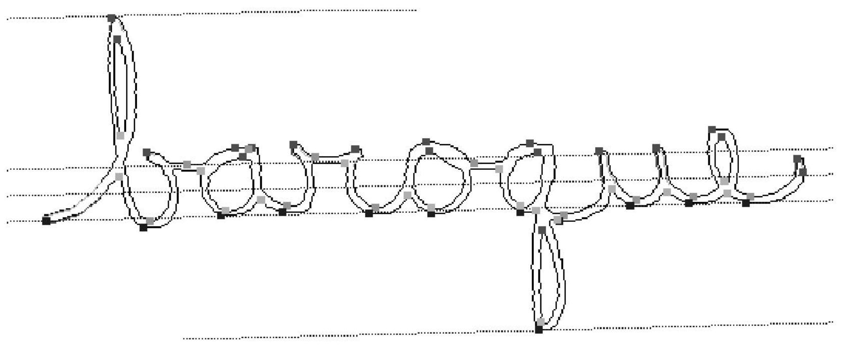

Fig. 4. Local contour extrema: maximas and minimas on the exterior and interior contours are marked along with the reference lines. The loop associated with "b" that rises above the reference lines in the middle is called an *ascender* and the loop associated with the "q" that falls below the reference lines in the middle is called a *descender*.

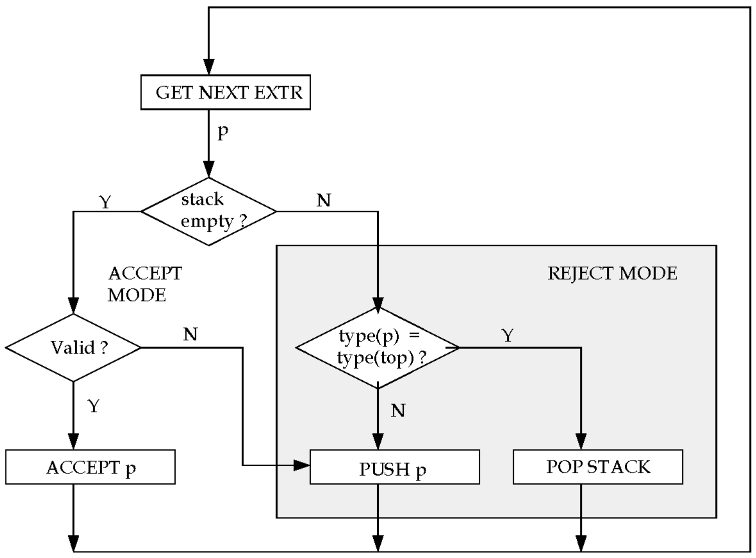

Fig. 5. Stack-based algorithm for parsing extrema on upper/lower contour. The accepted extrema are exactly the ones desired. p refers to the in-coming slope.

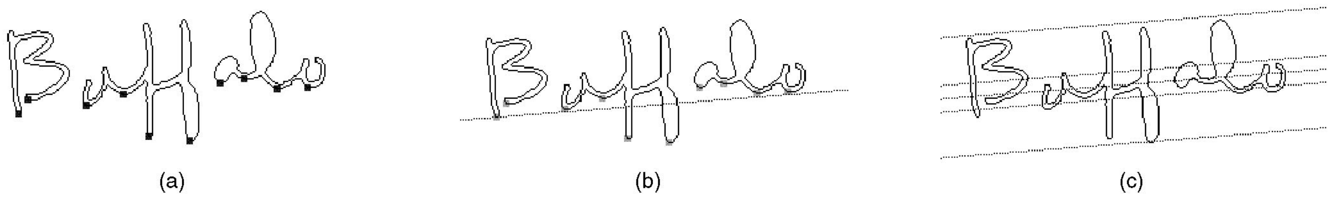

Fig. 6. (a) Exterior contour of word showing lower-contour minima. (b) Baseline determined as regression line through minima. (c) Angular reference lines from angular histogram at skew angle.

Each string S is composed of a strictly alternating sequence of maxima and minima. However, S is not constrained to contain equal numbers of maxima and minima, or to start or end with a valid extremum.

The productions $S \rightarrow iSi$ and $S \rightarrow vSv$ imply that a stack is required to parse strings of extrema. Initially, the system is the default state of accepting every extremum encountered. The first invalid extremum encountered is pushed on the stack and puts the system into a reject state. While in the reject state, no extrema are accepted—not even valid ones. In fact, a valid extremum encountered immediately after the invalid one gets *pushed* on the stack. The stack is popped only when the type of the next maximum is the same as that of the maximum on top of the stack. The system remains in the reject state until the stack is completely empty, at which point it returns to the accept state. It can be seen that the extrema discarded by the algorithm are exactly the ones on uncharacteristic excursions of upper and lower contours (Fig. 5).

3. **Reference Lines:** A time-tested technique for global reference line determination is based on the vertical histogram of pixels [12]. The method makes the implicit assumption that the word was written horizontally and scanned in without skew. Consequently, the method fails for words with significant baseline skew. Unfortunately, baseline skew is present to varying degrees in most freeform handwriting. Baseline skew can be corrected by

TABLE 1
Impact of the Preprocessing in Chaincode

	with pre-processing			w/o pre-processing		
Lexicon size	10	100	1000	10	100	1000
top 1%	96.8	88.5	74.0	91.1	75.8	58.9
top 2%	98.6	93.4	83.9	96.1	84.4	67.1

estimating the skew of the word and applying a rotation or shear transformation on the image.

An alternative to skew correction is the use of reference lines of the form $r_i(x) = m \times x + c$, where m is the skew angle and c is the offset of the corresponding reference line from the x-axis. Angled reference lines may be computed from the angular histogram at the skew angle.

The best-fit line through the minima points may be determined by a least-square linear regression procedure (Fig. 6). Minima that do not fall in the vicinity of the implicit baseline either correspond to descenders, or are spurious.

The halfline may be determined as the regression line through upper contour maxima. However, upper contour maxima are often poorly aligned, and spurious points are difficult to detect and remove, especially in more discrete writing (noncursive) styles [6]. Consequently, the resulting halfline is often erroneous. Reference lines computed from the angular histogram at the skew angle of the baseline have proved to be more reliable (Fig. 6c).

It must be noted that the above described regression method performs suboptimally on short words because of lack of sufficient number of minima points [8], [9].

4. **Word Length, Ascenders, and Descenders:** The number of minima on the lower contour of a word image is a good estimate of the length of a word. It is expected that such a length would be proportional to the number of letters in the word. Fig. 4 shows the length estimate of the word as 11 and also shows one ascender and one descender. The positions of *ascsenders* and *descenders* along with the length of the word are used as features. An ascender refers to the portion of a letter in a word that rises above the general body of the word. A descender refers to the portion of the word that falls below the general body of the word. Ascenders and descenders can be appropriately selected from the extremas of the upper and lower contour of the word.

2.3 Character Level

Techniques that are useful in character recognition are described in this section. Techniques that fall into this category are: 1) segmentation of words into characters, 2) stroke width, and 3) feature extraction.

1. **Segmentation:** Analytical processing in handwritten word applications involves segmenting a word into its constituent characters (Fig. 3b). If the distance between y-coordinates of the upper-half and lower-half of the outer contour for an x-coordinate is less than or equal to the average stroke width (described in next paragraph), then the x-coordinate is marked as an element of a ligature. This procedure is repeated for subsequent x-coordinates. Ligatures in the extremities are eliminated.
2. **Stroke Width:** By tracing contours from the left-most point to the right-most point, the following y-distances are computed for each x-coordinate: 1) distance between upper and lower trace of the outer contour, 2) distance between the upper trace of the inner contour and upper trace of the outer contour, and 3) the distance between lower trace of the inner contour and lower trace of the outer contour. Based on the number of occurrences of each distance value for the image, a histogram is obtained. The peak of the histogram gives a good estimate of the average stroke width.
3. **Features:** Several features extracted from the chain code representation have proved to be extremely useful in efficient character recognition. One such successful implementation using histograms of chaincode features has been described in [13]. Segment(s) of characters can be divided into 3×3 subimages and local features can be collected from each subimage. For example, distribution of the eight directional slopes for each subimage can form a feature vector of length 72 ($8 \times 3 \times 3$).

3 Experiments and Results

In this section, we present the experimental results to illustrate the benefits of chaincode representation and processing corresponding to each of the three levels (image, word, and character) described.

1. **Image Level:** We used a test set of 3,000 word images, including city names, firm names, personal names, street names, and state names, collected from live mail pieces. Table 1 shows the improvement in a word recognizer's results ([10]) when preprocessing is used. Performance degradation is significant for large lexicons.

   The recognizer first segments the word at potential character boundaries. Neighboring segments are then grouped together and sent to a character recognizer. The possible choices of characters with the respective confidence values is returned by the OCR. The segmentation points are viewed as nodes of a graph. Finding the word

TABLE 2
Lexicon Reduction Using Word Level Features Extracted from Chaincode

Total	Rejects	Top 100	200	300	400	500	**600**	**700**
768	7	571	665	700	735	747	755	758
%	0.9	75.0	87.4	92.1	96.6	98.2	99.2	99.6

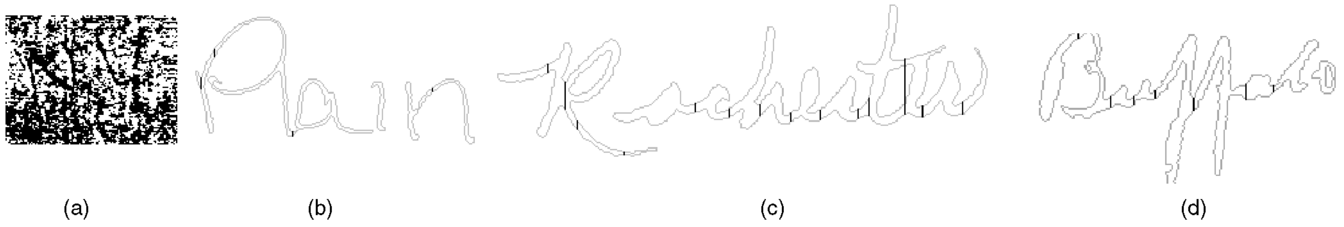

Fig. 7. (a) Noise in the image could not be removed; (b) segmentation method described in the paper failed; (c) broken contours could not be joined during the smoothing process; and (d) the ascenders of the "f" were not picked.

which best matches a lexicon entry is transformed into a search problem. Each path in the graph has an associated cost based on character choice confidences. The approach can be viewed as a process of accounting for all the segments generated by a given lexicon entry. Lexicon entries are ordered according to the "goodness" of match.

2. **Word Level:** A lexicon reduction method takes as input a lexicon (which is assumed to contain the truth value) and returns a reduced lexicon. Such a module may then be used as a front end for a conventional classifier which works with the image and the reduced lexicon. In fact, a word classifier that provides a ranked lexicon as output can theoretically serve as a lexicon reducer, simply by considering the top N entries of its output. Typically, lexicon reduction methods use word length and presence of ascenders/descenders because they are fast to compute.

   A set of 768 lowercase images of city names from actual mailpieces was used as the test set. The lexicon was a randomly generated list of 1,000 city names. The results are tabulated in Table 2. The lexicon was reduced using the features of word length, ascenders, and descenders. Lexicon can be reduced in half with a less than 2 percent error rate.
3. **Character Level:** Table 3 shows the speed efficiency of a character recognizer [10] that uses chaincode representation. For lexicons of size 1,000, the time consumed by the chaincode based preprocessing and feature extraction routines is less than 10 percent of the total time.

TABLE 3
Timing Analysis

Modules	Time consumed (ms)		
Lexicon size	10	100	1000
Pre-processing	10	10	11
Segmentation	15	15	15
Feature extraction	34	34	37
Sub-total	59	59	63
(% of total recognition time)	(28.9%)	(15.4%)	(9.1%)

4 Discussion

Contour representations are well-suited for representing handwritten word images. The storage is lossless and the processing time required is relatively small. While chaincode encoding of contours has been favored by researchers in the area of handwriting recognition ([10], [12], [13]), the representation does have some drawbacks. First, a simple shift of a single pixel can cause a break in the contour or introduce a point that does not adhere to the property that the contour must be a single pixel wide at all points of the contour. Such an event can occur during slant correction or noise removal. Another disadvantage would be in segmentation, where the ligatures that must be "cut" are far apart in the one-dimensional chaincoded contour although they are in close proximity in the two-dimensional image space. Further, contours can become exterior from interior (and vice versa) following the segmentation operations causing additional bookkeeping. Fig. 7 shows some examples where the methods described in this paper prove to be inadequate.

References

[1] T. Pavlidis, *Algorithms for Graphics and Image Processing.* Rockville, Md.: Computer Science Press, 1982.

[2] H. Blum and R. Nagel, "Shape Description Using Weighted Symmetric Axis Features," *Pattern Recognition,* vol. 10, pp. 167-180, 1978.

[3] H. Freeman, "Computer Processing of Line-Drawing Images," *Computing Surveys,* vol. 6, no. 1, pp. 57-97, 1974.

[4] C.-C. Lu and J.G. Dunham, "Highly Efficient Coding Schemes for Contour Lines Based on Chain Code Representations," *IEEE Trans. Comm.,* vol. 39, no. 10, pp. 1,511-1,514, 1991.

[5] G. Kim and V. Govindaraju, "Efficient Chain Code Based Image Manipulation for Handwritten Word Recognition," *Proc. SPIE Symp. Electronic Imaging Science and Technology (Document Recognition III),* vol. 2,660, pp. 262-272, San Jose, Calif., Feb. 1996.

[6] M.K. Brown and S. Ganapathy, "Preprocessing Techniques for Cursive Word Recognition" *IEEE Trans. Pattern Recognition,* vol. 16, no. 5, pp. 447-458, 1983.

[7] M.J.J. Holt, M.M. Beglou, and S. Datta, "Slant Independent Letter Segmentation for Off-line Cursive Script Recognition" *From Pixels to Features II,* S. Impedovo and J.C. Simon, eds., pp. 41-47, Elsevier Science, 1992.

[8] T. Caesar, J.M. Gloger, and E. Mandler, "Estimating the Baseline for Written Material" *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 382-385, Montreal, 1995

[9] M. Cote, E. Lecolinet, M. Cheriet, and C.Y. Suen, "Automatic Reading of Cursive Scripts Using a Reading Model and Preceptual Concepts, The PERCEPTO System," *Int'l J. Document Analysis and Recognition,* vol. 1, no. 1, pp. 3-17, 1998.

[10] G. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 4, pp. 366-379, Apr. 1997.

[11] E. Yacoubi, J.M. Bertille, and M. Gilloux, "Towards a More Effective Handwritten Word Recognition System" *Proc. Int'l Workshop Frontiers in Handwriting Recognition,* pp. 378-385, 1994.

[12] F. Kimura, M. Shridhar, and N. Narasimhamurthi, "Lexicon Driven Segmentation—Recognition Procedure for Unconstrained Handwritten Words," *Proc. Int'l Workshop Frontiers in Handwriting Recognition (IWFHR-3),* pp. 122-131, Buffalo, New York, 25-27 May 1993.

[13] F. Kimura, M. Shridhar, and Z. Chen, "Improvements of a Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 18-22, Tsukuba Science City, Japan, Oct. 1993.