Pergamon

Pattern Recognition, Vol. 29, No. 1, pp. 77–96, 1996
Elsevier Science Ltd

0031-3203/96 $15.00 + .00

0031-3203(95)00072-0

CHARACTER SEGMENTATION IN HANDWRITTEN WORDS—AN OVERVIEW

YI LU and M. SHRIDHAR
Department of Electrical and Computer Engineering, The University of Michigan–Dearborn, Dearborn, MI 48128-1491, U.S.A.

(*Received* 10 *May* 1994; *in revised form* 24 *April* 1995; *received for publication* 17 *May* 1995)

Abstract—This paper is the second part of a review series on the character segmentation techniques. In this paper, we present an overview on the most important techniques used in segmenting characters from handwritten words. It is well-recognized that it is difficult to segment individual characters from handwritten words without the support from recogniton and context analysis. One common characteristic of all the existing handwritten word recogniton algorithms is that the character segmentation process is closely coupled with the recognition process. This review consists of three major portions, handprinted word segmentation, handwritten numeral segmentation and cursive word segmentation. Every algorithm discussed in the paper is accompanied with a flow chart to give a clear grasp of the algorithm. One section summarizes the terms and measurements commonly used in handwritten character segmentation. The bibliography contains a comprehensive list of work in handwritten character segmentation and recognition.

Handwritten words Cursive words Handprinted words Character segmentation
Character recognition Word recognition Touching characters Broken characters

1. INTRODUCTION

This paper is the second part of a review series on the character segmentation techniques. The first part of the review series covers the character segmentation techniques in machine-printed text.[1] In this paper, we present an overview on the techniques in segmenting handwritten characters. The authors believe that this is the first comprehensive discussion in literature on techniques for segmenting handwritten words into characters.

The objective of automatic document processing is to recognize text, graphics and pictures in digital images and extract the desired information as would a human. In many applications, the contents of text are handwritten words. The recognition of isolated letters, both cursive and hand-printed, has achieved impressive progress in the last 30 years, and many effective techniques have been developed to solve the problem and the results are in the range of 90% plus [KaP87, MSY92, SBM80]. However, classifiers in an optical character recognition (OCR) system often need to face the input of text which consists lines of words and sometimes graphs. Usually the first task for a document process is to separate graphs and pictures from text. Textual and graphical are two categories of document processing dealing respectively with the text and the graphics components of a document image. Textual processing includes [GoK92]:

- determining the skew (any tilt at which the document may have been scanned);
- finding columns, paragraphs, text lines, and words;
- performing OCR.

Finding text columns and paragraphs usually belongs to document layout analysis. Extracting lines, words and characters are often referred to as line, word, and character segmentation. This paper is confined to character segmentation in handwritten word images.

The goal of a character segmentation algorithm is to partition a word image into regions, each containing an isolated and complete character. For machine-printed text, as presented in reference (1), a word image is often segmented into isolated characters which in turn serve as input to a recognizer. However, it is extremely difficult to segment characters in handwritten words without the support from recognition algorithms. Therefore, unlike the problem of machine-printed character recognition, the handwritten character segmentation and recognition are often closely coupled. This paper provides a comprehensive overview on the character segmentation techniques, namely techniques that extract characters from handwritten word images. The classification features, feature extraction methods and classification algorithms are out of the scope of this paper. Excellent surveys on handwritten character recognition can be found in references (2, 3, 4).

Handwriting recognition systems can be divided into two broad types: Optical Character Readers (OCR), where a whole page of handwritten or a mixed

(a) Hand-printed words

(b) Cursive words

Fig. 1. Examples of hand-printed and cursive handwritten words.

of handwritten and machine-printed text is processed, and On-Line Character Recognition (OLCR), where the characters are converted and recognized interactively as they are formed. This paper focuses on the segmentation problems in the domain of OCR. A good review on OLCR in handwriting text can be found in references (5, 6).

This paper confines the handwritten text domain to Romance or Anglo-Saxon languages, and the word images are assumed binary unless otherwise indicated.

In general, there are three major categories of word recognition approaches:

- Global approaches

They are also called holistic approaches. These approaches recognize a word as a whole entity. Classification features are extracted from the entire word image, such as the number of vertical strokes, ascenders, descenders, and loops, and a decision is made according to word classifiers.

- Segmentation-based approaches.

A word image is first partitioned into a sequence of segments, the segments are recognized by character classifiers, and then the recognized segments are matched with the possible words.

- Hybrid approaches.

These approaches combine global and segmentation-based methods into the segmentation-recognition processes.

Recognition based on the entire word is difficult and complicated. It is also time and memory intensive since all the representations of the words in the vocabulary under investigation need to be kept and matched with the input word image. Another drawback is that incorrectly spelled words cannot be recognized since they do not belong to the vocabulary. Representative work of entire word recognition can be found in references (7–17).

In most existing OCR systems, word recognition algorithms fall in the last two categories in which character segmentation still plays a critical role. Handwritten words, based on the form of written communication, can be divided into two categories: cursive scripts *versus* hand-printed characters. In practice, a combination of the two is likely to occur in a word image. Figure 1 shows an example of hand-printed and cursive handwritten words. This review covers techniques for solving both problems. We first present in Section 2 the common terms and measurements used in handwritten character segmentation. We present algorithms for segmenting handprinted words in Section 3, segmenting handwritten numerals in Section 4, and segmenting cursive word images in Section 5. Section 6 concludes the paper by discussing future research trends in handwritten text segmentation and recognition. Every algorithm discussed in the paper is accompanied with a flow chart to give readers a clear grasp of the algorithm, and the performance of the segmentation-recognition system is also discussed. The bibliography contains a comprehensive list of work in handwritten character segmentation and recognition.

2. COMMON TERMS AND MEASUREMENTS

The following is a list of most commonly used terms and measurements in handwritten character segmentation.

- **ascender of a character:**
  the portion of a character above the main body of the character;
- **aspect ratio of a character:**
  the character width divided by the character height;
- **axis of a word:**
  the shortest path traverses from one extremity to the other and it remains in the body of the word;
- **base line:**
  the row where the descender starts;
- **bounding box of a character or a word:**
  a minimum rectangular box that contains the character or the word,
- **character pitch:**
  the space a character occupies;
- **character width:**
  the number of columns a character occupies;
- **character height:**

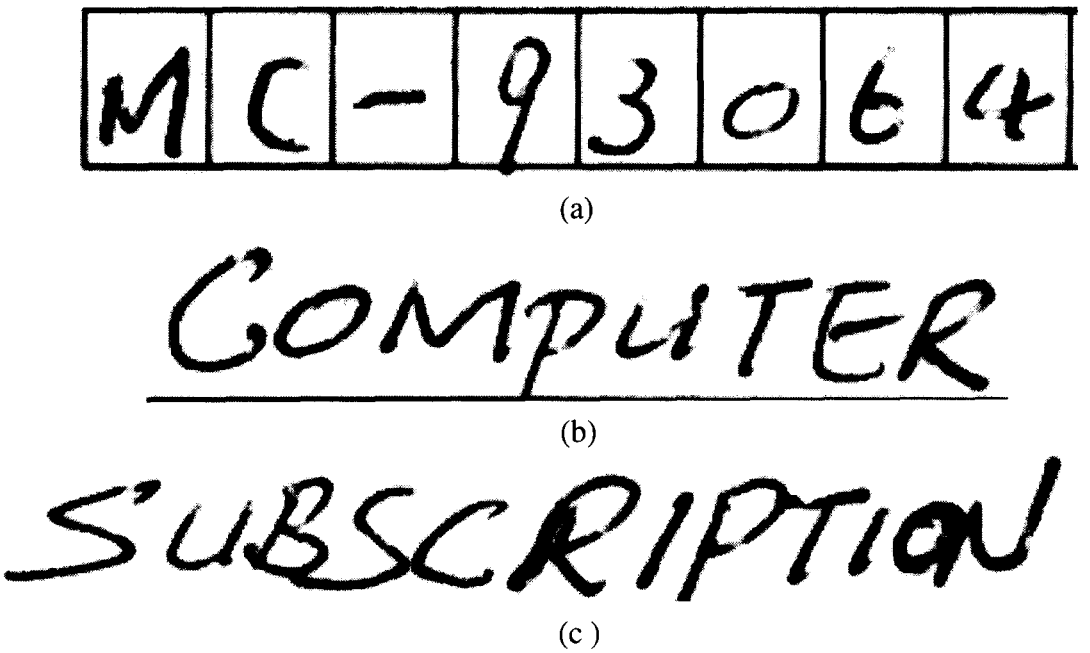


Fig. 2. Examples of handprinted words; (a) boxed characters, (b) spaced characters; and (c) touching characters.

the number of rows a character occupies;

- **character gap:**
  distance between two characters;
- **character interval:**
  distance from the center of a character to the center of its successive character;
- **connected component (cc):**
  a group of foreground points connected by either four neighbors or eight neighbors;
- **descender of a character:**
  the portion under the main body of the character;
- **horizontal projection:**
  a function of row index and its value at index i is the summation of all the foreground pixels occurring at row i;
- **lexicon:**
  a dictionary of legal words;
- **occupancy ratio:**
  the number of on-pixels in the area divided by the total number of pixels in the area;
- **runs in the horizontal or vertical direction:**
  a run at column (row) i is a continuous sequence of foreground pixels occurring at column (row) i;
- **top base line:**
  the row where the ascender starts;
- **vertical projection:**
  a function of column index and its value at index i is the summation of all the foreground pixels occurring at column i;
- **x-height of a line:**
  the number of rows between the ascender and descender.

3. SEGMENTATION OF HAND-PRINTED WORDS

Text in this category is produced under the assumption that at the end of each character the writer lifts his (her) pen and then starts another character. Characters can be further divided into the following categories:

(1) boxed characters,
(2) spaced characters, and
(3) touching characters.

Figure 2 shows one example for each category. The text in the first two categories pose no difficulties in segmenting characters. The major problem in segmenting handprinted words is to split touching characters. When a word image is handprinted, contact between adjacent characters may occur due to style of writing as well as to crowding of symbols. A stroke belonging to one character may touch a stroke of the other in some cases. Casey and Horne found, based on their study from approximately 10,000 uppercase letters in the NIST data base, the amount of overlap in most touching characters was slight.[18] Unlike machine printing, the location of the contact point can be at any elevation and the segmentation boundary can be nonlinear. However, some of the techniques for splitting machine printed touching characters discussed in reference (1) are applicable to solve this problem.

In this section, we discuss two different types of algorithms for segmenting handprinted characters, one is based on contour behaviors and another is based on distance transform to find the best path to split merged characters.

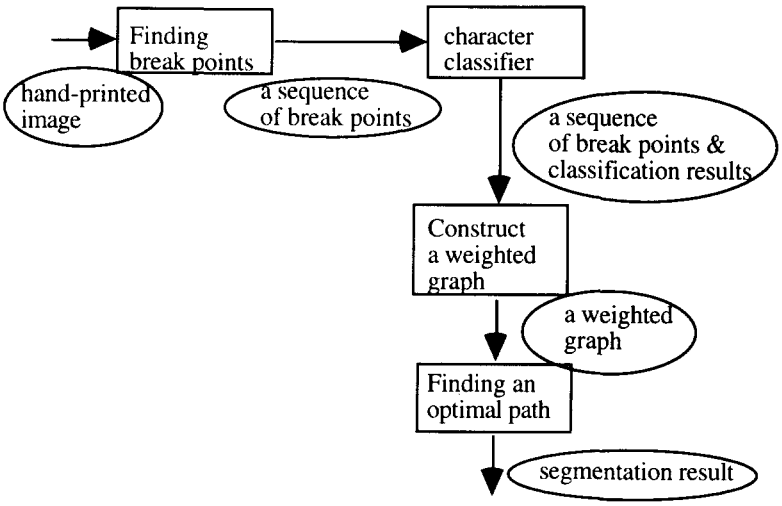


Fig. 3. Control of segmentation and recognition of hand-printed words.

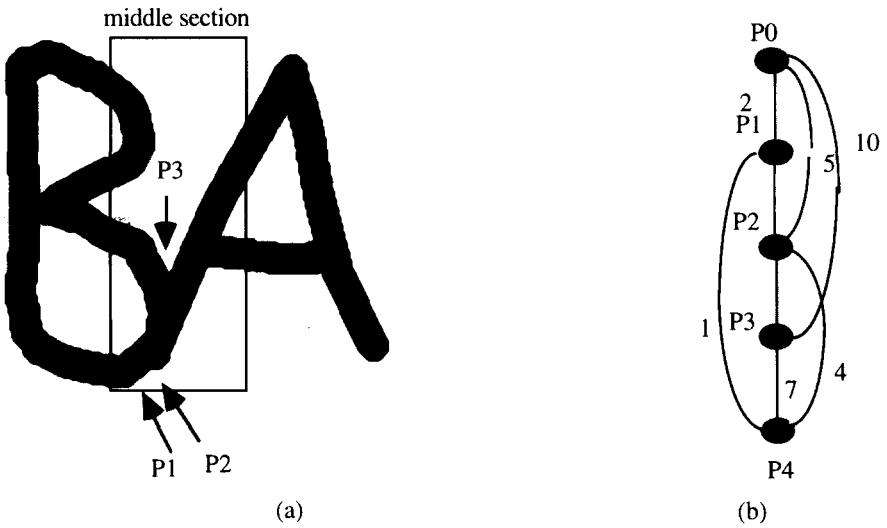


Fig. 4. An example to illustrate Casey and Horne's algorithm. (a) Two touched handprinted characters and the candidate break points, P1, P2, and P3. (b) The graph model of (a).

3.1. *Segmenting handprinted words using contour feature*

The representative work using contour-based approaches can be found in references (18, 19). Here we discuss the algorithm presented by Casey and Horne. This algorithm, illustrated in Fig. 3, suggested that when two characters are touching, there should be a concavity located at the point of contact. Therefore, the algorithm examined the middle section of a touching character image for concavities. Local curvature was estimated through edge direction, which was obtained by a contour-following algorithm. A break point was a locally concave point if it had nonvertical direction and was within a parameter D in distance from a point on the opposite side of the same contour. The break points for a given image formed a set of admissible segmentation choices upon which a set of classifier results were generated. A segmentation of the image may be specified by choosing any subset of these break points. Figure 4(a) shows an example of touched characters and the candidate break points.

Casey and Horne used a graph model to describe the control of the above segmentation-recognition processes. The interior nodes of the graph represented the break points, the first and last nodes represented the left and right sides of the image, and every branch was associated with a cost. Figure 4(b) shows the graph model corresponding to the example shown in Figure 4(a). P0 and P4 represent the left and right sides of the touched character image, P1, P2 and P3 are the candidate break points in the word image in Fig. 4(a). Each arc between two nodes represents the possible character segment bounded by the two points. Casey and

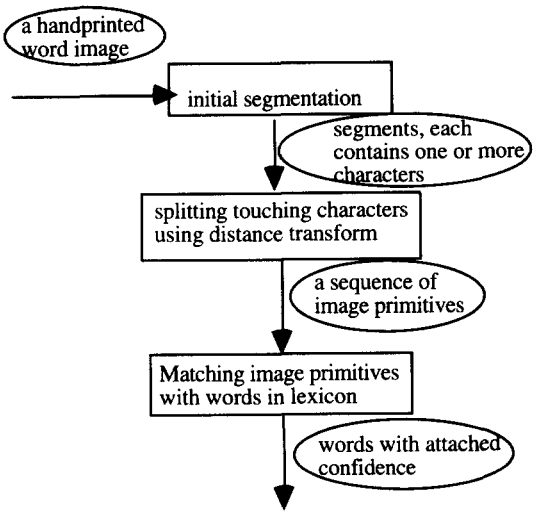


Fig. 5. Word segmentation based on distance transform.

Horne pointed out that the cost values were generally both application and system dependent. The cost function may depend on character width, classification confidence values, contextual evaluations, etc. Under this representation, the optimal segmentation amounts to finding a minimal cost path.

The segmentation method was implemented in a prototype OCR system at IBM Almaden Research Center, and the segmentation-recognition system has been applied to advanced postal address recognition.

3.2. *Segmenting handprinted words using distance transforms*

Gader *et al.*[20] proposed a distance transform-based algorithm to segment handprinted words. The algorithm, illustrated in Fig. 5, consisted of three major steps, the initial segmentation, the splitting and the merging steps. The initial segmentation step first used the connected component method to compute the initial character segments. Then it detected and removed all the punctuation marks, and grouped broken characters by connecting the horizontal bars. The output from the initial segmentation step should contain either one or multiple characters. The splitting step was based on the distance transform technique. The distance transform is a classic technique in binary digital image processing. Assume **A** is the set of foreground pixels and A^c is the set of background pixels in an image **I**. Two types of distance transforms can be defined depending on application: every pixel x in **I** was assigned a number that is the distance between either x and A^c or x and A. Figure 6 shows an example of the city-block distance transform.

Gader *et al.*'s algorithm transformed the background pixels to the distance value to the nearest stroke. After the distance transform, pixels with small distance transform values formed the possible splitting paths. A splitting path was formed by the pixels that stayed as far from the strokes as possible without turning too much. Heuristics were used to define starting points of the splitting paths and to modify the distance function. The result of the segmentation process was a sequence of segmented primitives, each of which contained either a single character or a partial character. Finally, a dynamic programming algorithm assisted by a neural network classifier was used to match the sequence of primitives from the segmentation process with the words in a lexicon. The dynamic programming algorithm first computed the best path through the space of primitives based on the values of the nodes on the path. The value of each node on a path was the best matched classes and the associated confidence value provided by the neural network classifier. The value of a path was computed by averaging the values of the nodes on the entire path. The dynamic programming algorithm took a list of the primitives and a word in the lexicon as input and output a matching confidence value between 0 and 1.

Gader *et al.*'s algorithm was tested on the data from the U.S. Postal Service (USPS) mail pieces. The data set contained over 3000 address blocks images, and the test set contained 319 word images. The performance of the word segmentation-recognition system on the

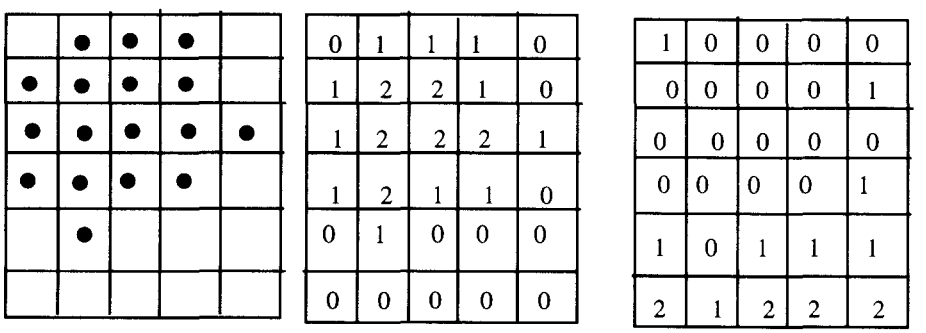


Fig. 6. An example of city-block distance transform.

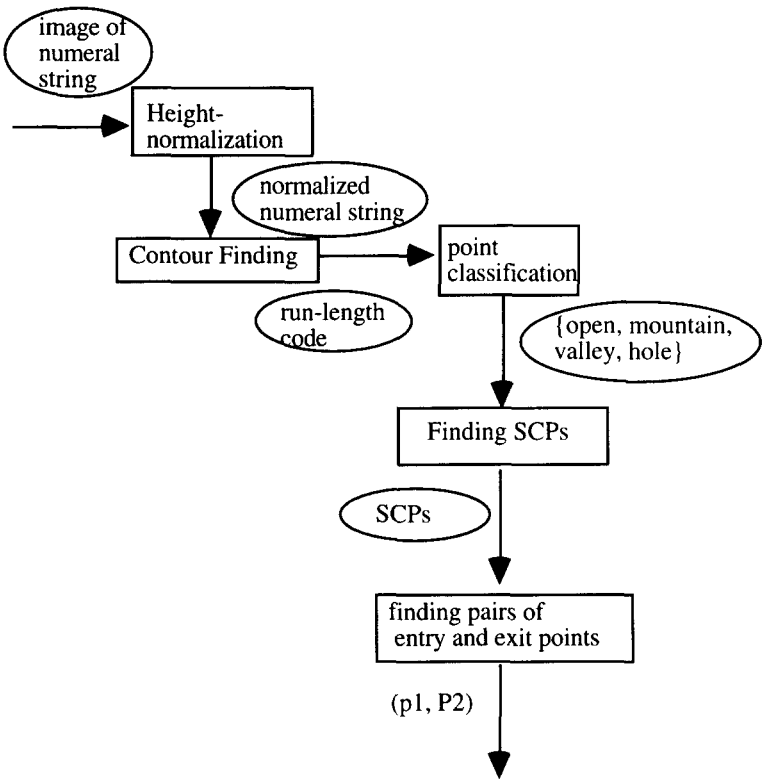


Fig. 7. Finding the entry and exit points of a cut path between connected digits.

test data with a lexicon size over 470 was 67.02% correct at the top rank with no thresholds applied to the word confidence value.

4. SEGMENTATION OF HANDWRITTEN NUMERALS

The segmentation of handwritten numerals has important applications in many areas within which the recognition of ZIP codes on postal mail pieces and the reading of dollar amounts on bank cheques received much attention in the research and development community. In these two application domains, the common approach is to extract the numerical field first and then segment the extracted numerical string into individual digits. However, the two applications have different constraints which can facilitate the segmentation and recognition process. In the check reading application, the digit recognition results can be verified by the cursive amount, whereas in the ZIP code recognition, the number of digits is fixed, e.g. on U.S. mail pieces, the ZIP code is either five or nine digits. These constraints have been proved useful in the following segmentation algorithms. Three different methods are discussed here. The first segmentation method is merely feature-driven with no support from recognition results nor context. The other two methods were developed in the application environment of Postal ZIP Code recognition.

4.1. *Structural feature-based segmentation*

Many feature-based segmentation methods have been developed for segmenting numeral strings.[21,22,23] The method we present here is by Strathy *et al.*[22] This method used structural features to find the correct entry and exit points of a path that will separate the leftmost digit from a string of two connected digits. Figure 7 illustrates the detailed segmentation steps employed in this method. The most important characteristic of this method is that it allows the separating path to be either a straight line or a curve.

The input image was first normalized in height and then the contour was found and represented in run-length coding. Each point on the contour was assigned one of the following region types:

(1) open region.
(2) mountain region,
(3) valley region, or
(4) hole region.

These features are illustrated in Fig. 8. The next step in the algorithm was to compute the significant contour points (SCP). The SCPs were either the corner points in mountain, valley and open regions or the points that were a small distance away from their neighboring SCPs and satisfy one of the following conditions:

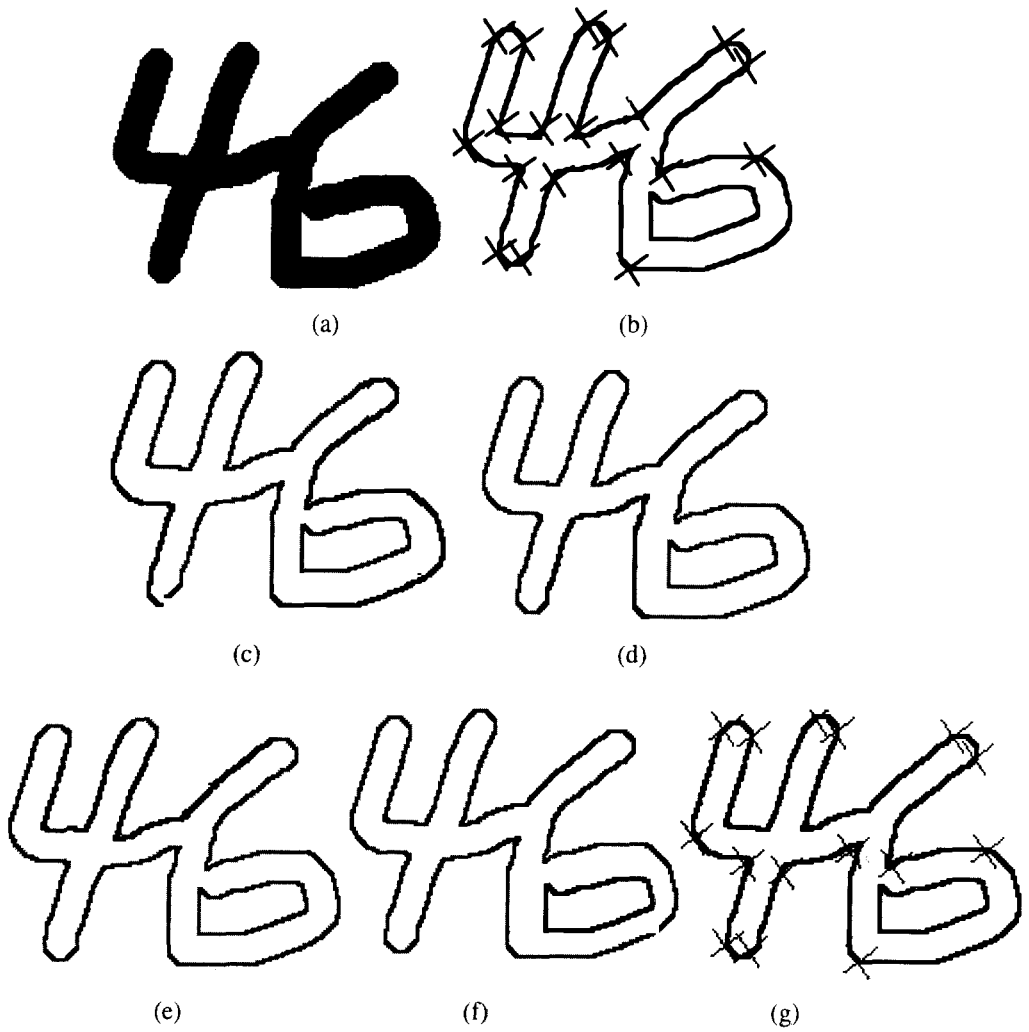


Fig. 8. Illustration of structural features. (a) A pair of touching digits; (b) corner points; (c) open regions; (d) mountain region; (e) valley regions; (f) hole region; (g) significant contour points (SCPs).

- the minimum of each valley,
- the maximum of each mountain, or
- the exit points of the imaginary straight lines formed by extending the contour through the stroke at concave corners in mountains and valleys.

From the SCPs the algorithm tried to find pairs of points (P1, P2) such that P1 could be an entry and P2 could be an exit point for a correct cut path. P1 and P2 were chosen from different region types based on the highest credit score. The credit assignment was as follows:

(1) a fixed number of credits for each mountain/valley pair,

(2) a fixed number of credits for corner points,

(3) credits for the nearness to each other of the points in the pair,

(4) credits for the sharpness of the concavity at each of the SCP points,

(5) credits for the nearness of an SCP's valley (mountain) to the top (bottom) of the image,

(6) credits for the distance of an SCP from the bottom (top) of its mountain (valley),

(7) credits for the degree to which a valley corner is above a mountain corner in the image,

(8) credits for the degree to which stroke pixels outnumber background pixels in an imaginary straight line drawn between the pair, and

(9) credits for the nearness of the pair to the left side of the image.

The pair found by this method was used for guiding cut from the entry SCP to the exit SCP.

The segmentation system was trained on 212 images of touching digit strings extracted from the bu0 100 and bu0 200 subdirectories of the USPS Office of Advanced Technology Database of handwritten cities, states, ZIP codes, digits, and alphabetic characters. The system was tested on two sets, one contained 460 pairs of touching digits and the other contained 191 touching digit strings. The results were reported in

categories of first, second, etc., up to five attempts. The performance ranged from 48 to 97% on the first test set and 57.6 to 94.2% on the second test set.

4.2. *Segmentation of ZIP codes*

The next two algorithms, one proposed by Kimura and Shridhar[24,25] and the other by Matan *et al.*,[26] have the following common characteristics.

- both began with slant correction and extraction of connected components,
- explicitly employed the heuristics such as number of digits in a ZIP code, and
- verify the recognition results using valid ZIP code databases.

4.2.1. *Kimura and Shridhar's algorithm*

This algorithm has four major steps, initial segmentation, multicharacter detection, splitting module and recognition (see Fig. 9). Our discussion will focus on the first three steps.

The initial segmentation tried to combine the labeled connected components of a broken character. All connected components were sorted according to their leftmost x-coordinates, and then the pair of adjacent components that had maximum horizontal overlapping was merged repeatedly until there was no pair with horizontal overlapping exceeding a given threshold. The horizontal overlapping ratio H_{or} was defined as:

$$H_{or} = \mathrm{Max}\{w_{12}/w_1, w_{12}/w_2\},$$

where w_1, w_2 are the widths of the first and the second

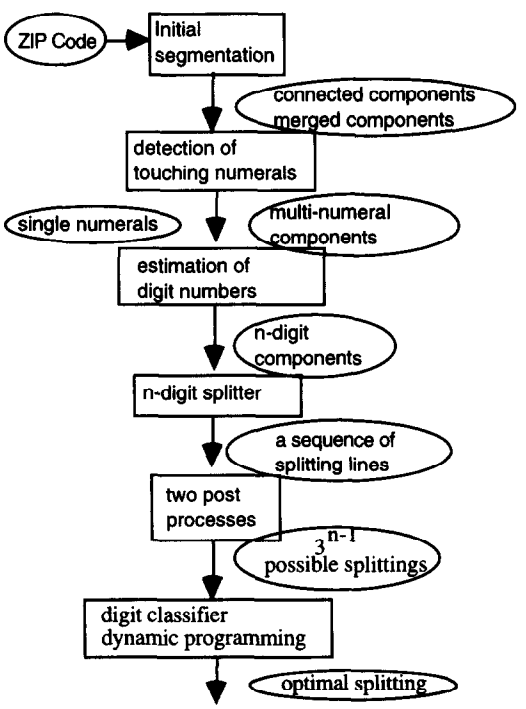


Fig. 9. Numeral segmentation based on vertical and slant splitting.

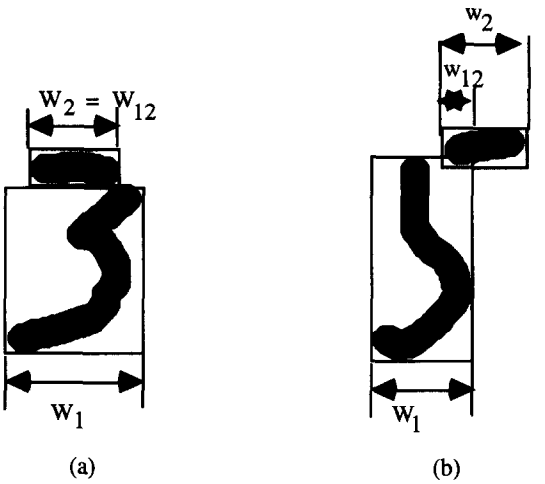


Fig. 10. Illustration of merging two components of the same numeral. (a) Measures of horizontal overlapping area; (b) an example of two-stroke "5".

component respectively and w_{12} the width of the horizontal overlapping of the two components (see Fig. 10). In addition, a special measure to evaluate the proximity of two strokes of a broken "5" [see Fig. 10(b)] was defined using the enclosing rectangles of the strokes.

Following this initial segmentation, a numeral recognition algorithm was applied to each component and the components that have high likelihood to be isolated numerals were recognized. The remaining components were supplied to a connected numeral detector and classified according to the estimated number of the component numerals. The number of numerals in each component was estimated based on the knowledge that the number of numerals in a ZIP code was known to be either 5 or 9. If the estimated number of numerals within a component was determined to be two or more, the component was sent to the splitting algorithm.

Kimura and Shridhar developed a two-numeral splitter providing both vertical and slant splitting. The two-numeral splitter used a criterion J_0 to split the projection into left and right parts:

$$J_0 = S_b/S_w,$$
$$S_b = (N_1 N_2/N^2)(m_1 - m_2)^2$$
$$S_w = (N_1/N)\sigma_1^2 + (N_2/N)\sigma_2^2,$$

where N_i is the number of pixels in each part, $N = N_1 + N_2$, m_i and σ_i^2 $(i = 1, 2)$ are the mean and variance of the x-coordinates of the pixels in the left and right parts, respectively. S_w should be calculated for all possible x-coordinates and the optimum location x_1 and the maximum value was selected as the location of the splitting line. The extension of the above splitting algorithm to the nonvertically separable case was made possible with the application of discriminant analysis. Instead of the vertical projection, the projection to the direction of maximum separ-

ability was used for the valley point detection. The detail of the algorithm for splitting non-vertically separable numerals can be found in reference (25). Kimura and Shridhar claimed that the splitting algorithm was not affected by noise and irregularities on the character image due to the cumulative or integrative characteristics. However, the insensitivity of the local shape near the separating line could cause inexact splitting leading to superfluous cut, halfway cut of a stroke, and improper cut of a loop. Therefore, the following two post-processing procedures were suggested to remedy the problem. One post process was to re-merge small segments dismembered by the superfluous cuts. The process consisted of assigning labels to each connected component on both sides of the splitting line, determining the dominant components on each side, and changing the membership of nondominant components connected to the dominant component on the opposite side. Another post process was to correct the situations such as a halfway cut of a stroke or a cut of a loop. For the halfway cut, the cutting section should be shifted to between two adjacent slant runs where the run length first changes remarkably. For the cut of a loop, the section between pairs of two disconnected runs should be shifted to a place between a pair of single short run.

The above two-numeral splitter yielded three most probable pairs of separated numerals. A numeral classifier was applied to these separated numerals and the pair with the minimum sum of (minus log) likelihood for the two numerals was selected as the optimum split.

The three- or more-connected numeral spliter was a generalized version of the two-numeral spliter. The generalization was straightforward, except for the increasing complexity of the optimization strategy. Readers can find detailed description of this extension in references (24, 25). An n-numeral component was separated by $n-1$ splitting lines, each of which has three possible locations. As a result there were 3^{n-1} possible splitting configurations among which the optimum splitting needed to be selected. Kimura and Shridhar used dynamic programming to reduce the computational complexity for the worst case to $O(9(n-2)+6)$.

Kimura and Shridhar tested their algorithm on two USPS ZIP-code databases known as the "bd" and "bs" databases. The samples in the data were totally unconstrained handwritten ZIP-code fields, and each sample contained five or nine isolated or connected numerals along with some other elements such as hyphen, period, and noise. The initial segmentation algorithm was applied to the "bd" data, and 8738 isolated numerals were selected manually. These isolated numerals were used as design and test samples for the numeral classifier. The entire ZIP-code recognition procedure was applied to the "bs" data set to test the ZIP-code recognition system as well as the splitting algorithms. The correct recognition rate, for the isolated numerals, ranged from 94.30 to 97.83%. For 92 two-numeral components, the segmentation procedure automatically segmented 84 components correctly and 63 of these were correctly recognized. For 13 three-numeral components, the segmentation procedure automatically segmented eight components and all eight were correctly recognized.

4.2.2. *Matan et al.'s algorithm.* This system was also developed for reading postal ZIP codes.[26] The overall aim was to segment and recognize the correct number of digits within an entire ZIP code. Figure 11 illustrated the major segmentation steps in the system.

After the preprocessing, the first step in the system was the **validity check** with the input being a connected component. The **validity check** can be described as follows:

(1) if a component was too small, it was discarded,
(2) if the component was too large to be a single digit, it was passed to the next level of segmentation, otherwise
(3) it was evaluated by the digit classifier.

The components with recognition scores above a certain threshold were considered as being correctly segmented and classified. When all the connected components within a ZIP code were recognized with scores above the threshold and the number of segments was five or nine, both the segmentation and recognition processes were concluded.

The components with scores below the threshold were further processed by a unit known as the **vertical-cut segmenter**, which consisted of the following steps:

(1) the connected components marked as solved were erased from the image;
(2) the vertical pixel projection of the remaining image was calculated by passing it through an exponential filter, which gave a value close to $+1$ for areas of white space and close to zero when the projection value was well over the estimated stroke width;
(3) the output from the exponential filter was smoothed and then used as the projection score function of cuts.

The cut-candidates were the maximum points of the projection score function. A candidate cut-point with a projection score greater than a certain threshold was termed an obvious cut. If the number of obvious cuts plus the number of good components was equal to either five or nine then the segmentation process was carried out.

If the number of obvious cuts was less than what was needed, it implied that there were touching characters and a recognizer was used to find them. If the number of obvious cuts was more than what was needed, all obvious cuts were discarded, and the system should evaluate all candidate cuts using the recognizer. This process was necessary since broken digits were possible either within the input image or produced by the vertical cuts.

The segments divided by the obvious cuts and the connected components were called clumps. The upper

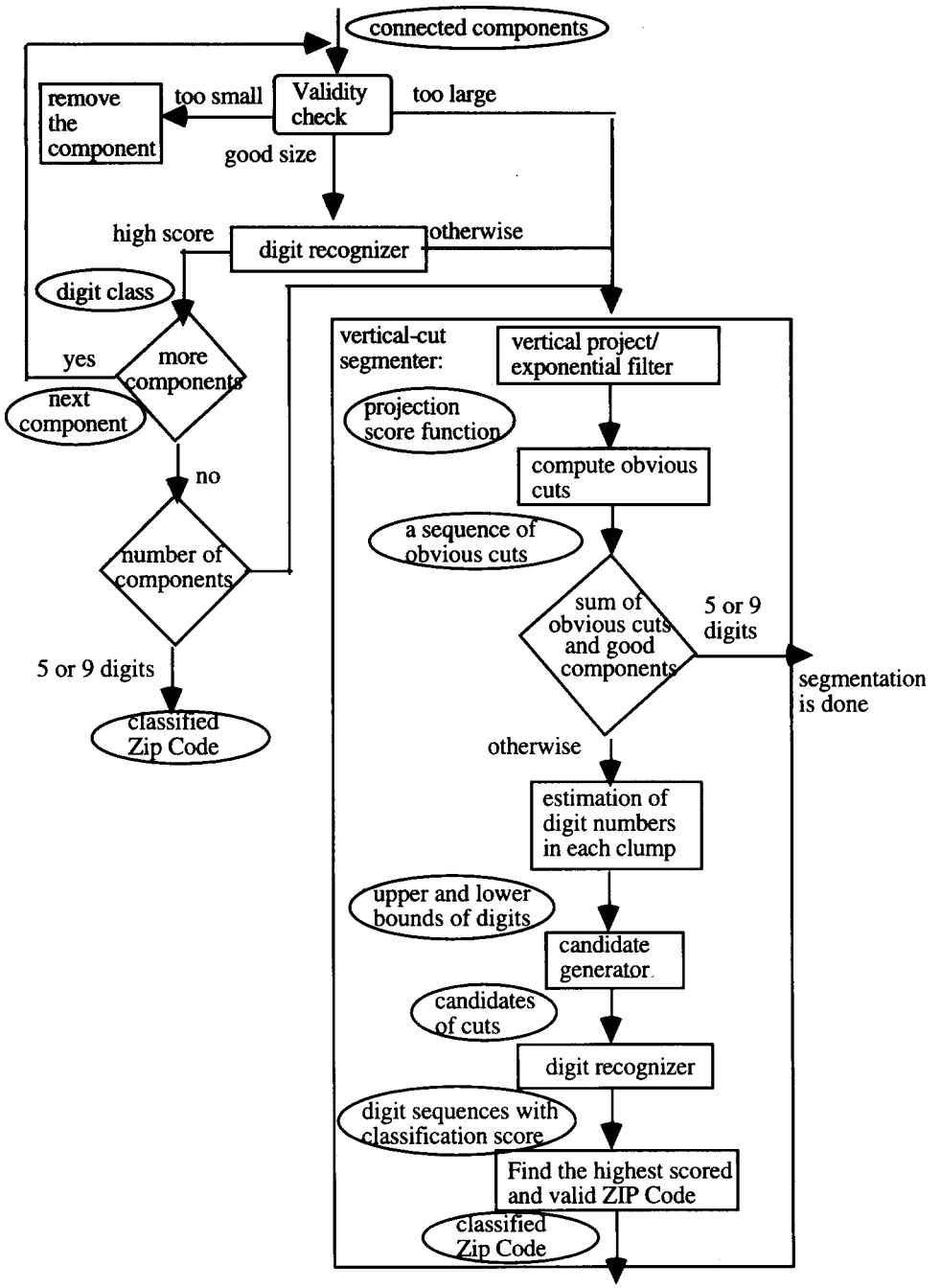


Fig. 11. ZIP code segmentation and recognition.

and lower number of digits contained in each clump was computed based on the estimated pitch. The candidate cuts were generated for each clump based on

(1) the estimated number of digits in the clump,
(2) the good projection scores, and
(3) nearly equidistant distribution.

Each candidate segmentation was sent to the classifier and the resulting probabilities were multiplied to give an overall score for that clump. The system evaluated all combinations of subsegments of the clumps, provided they gave the correct number of digits. The last step in the system was the check of validity of the resulting ZIP code by using a database of valid ZIP codes. If the resulting sequence of digits was not valid, the next-highest-ranking sequence of digits was checked. This process was repeated until a valid ZIP code was found.

The system was tested on 2585 images of five- and nine-digit ZIP codes in a USPS data base. The basic

(a) (b)

Fig. 12. (a) An example of mixed cursive script. (b) An example of pure cursive script.

system had a raw error rate of 26.15%. When 40% of the lowest scoring images were rejected, the error rate for the remaining 60% was 5.47%. Note these results were based on ZIP codes not digits and were obtained without using the ZIP code validation directory.

5. SEGMENTATION OF HANDWRITTEN CURSIVE WORDS

Handwritten cursive words can be either pure cursive script or mixed cursive (see Fig. 12). The text in both categories is difficult to segment and recognize. The difficulty stems from the following factors:

- the letters in cursive writing are often connected,
- the individual letters in a cursive word are often written so as to be unidentifiable as isolated characters, and
- the variance in writing styles.

The approaches presented in this section differ in both the architecture of segmentation-recognition systems and the sets of features employed to represent word images. Some approaches rely on an explicit segmentation of the word image into letters, while others perform segmentation during the recognition process.

5.1. *Segmentation as a preprocess to word recognition*

Algorithms in this category often involve a preprocessing before segmenting characters. The preprocessing includes noise removal and slant correction.[27,28] Cursive word segmentation usually consists two major steps, **presegmentation and segmentation**. There are two approaches in presegmentation. While both approaches try to segment a word image into a sequence of presegments, one assures that each segment contains no more than one character and the other assures that each segment contains at least one or more characters. At the segmentation step, the presegments are further processed to achieve more reliable results, namely, each segment containing exactly one character.

The common characteristic of the algorithms discussed in this subsection is that they all have simple segmentation methods and sophisticated recognition algorithms that are robust enough to tolerate the segment errors. One example of this type of approach is used in the word recognition algorithms based on Hidden Markov Model (HMM). Most HMM-based approaches first segment a word image into a sequence of characters and then use HMM to recognize the characters. Usually the Markov model is robust enough to recover segmentation errors, and therefore the segmentation techniques used in this type of algorithms are simple, such as using the aspect ratio of upper and lower contours of the connected components. However, this type of approach may only work on a small lexicon or vocabulary. The following subsections introduce the most representative algorithms for segmenting cursive words.

5.1.1. *Bozinovic and Srihari's algorithm.* This algorithm assumed nonslanted script and the connectivity of the lower contour of the word.[28] The presegmentation algorithm consisted of two major steps, finding presegmentation points (PSP) and compaction. The presegmentation operation cut the word image horizontally into minimal portions, termed presegment (PS), which were potential characters or partial characters. The presegmentation method was developed based on the following ligature analysis:

- a word image is dissected into three horizontal zones (see Fig. 13),
- local minima along the lower contour of the word image points usually occur on ligatures in the middle zone,
- the ligatures coming off letters with lower loops such as "g", "j" and 'f" correspond to local minima in the lower zone of the letter just preceding them, however,
- those minima may mark middle points in the following letters: "m", "u", "a", "o" "b", "d" and "r".

The computation of PSPs is illustrated in Fig. 14. First the algorithm searched for local minima along the lower contour of the word. For every local minimum, it looked left and right within certain limits to locate zones in which PSPs were eligible to be placed. Each of these zones was characterized by both a continuous sequence of single runs in vertical projection and the density less than a predetermined threshold. If such a zone was found, a PSP was placed in the middle of it, thus creating overall anywhere from zero to two new ones. The limits were, to the left, the immediately preceding (rightmost so far) PSP, and to the right, a heuristic boundary *ms*, the maximum shift parameter. In case of failure to generate anything to either the left or right of a local minimum, the algorithm looked for a point that had the lowest vertical projection value within the limit to the right.

The above process steps could result in the following undesired cases:

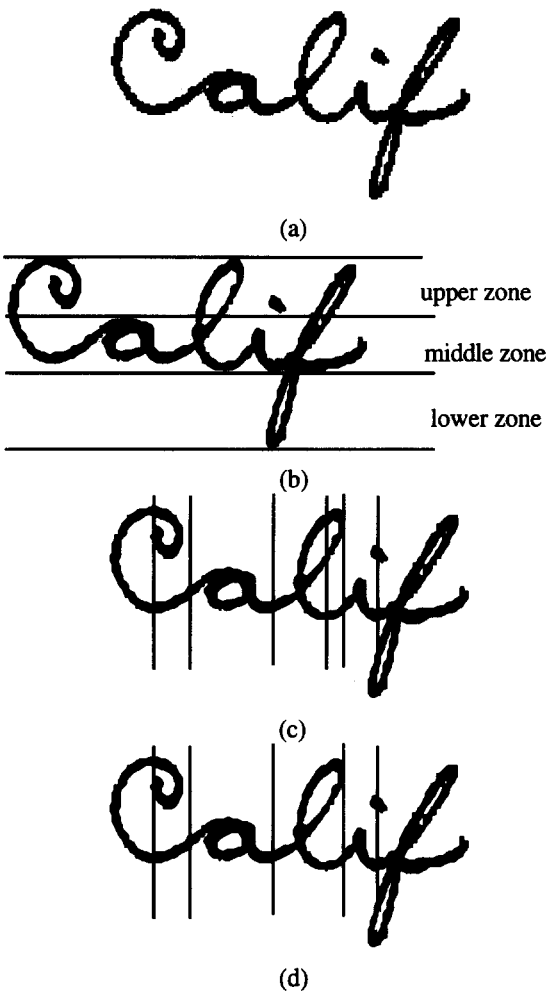


Fig. 13. (a) An image of word "Calif". (b) Illustrates the three zones on the word image. (c) Vertical lines corresponding to PSP's. (d) The PSP's after the compact procedure.

- Multiple PSPs within letters. For example, the "C" in Fig. 13 was cut to partial characters;
- a long ligature was assigned two PSPs. For example, the ligature between the "1" and "i" in Fig. 13.

To remedy the second problem, a compaction procedure was applied to the PSPs. The procedure searched for sequences of PSPs whose maximum distance was less than a threshold. If such a sequence was found, it was replaced by a PSP within the sequence that had the lowest vertical projection value. Figure 13(d) shows the PSPs after the compact procedure.

The presegmentation algorithm could miss segments altogether due to either a lack of surrounding local minima or too-crowded lettering, when actual PSPs were not in a single-run zone and the maximum shift parameter was too large to stop generation of the next PSP at that step. Bozinovic and Srihari's used the pre-segments in a procedure known as Event Detection to compute events in terms of certain topological

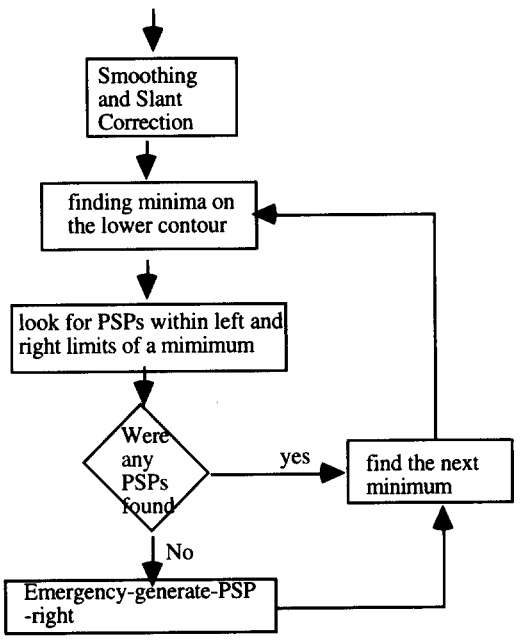


Fig. 14. Segmentation based on minima on lower contour of a word image.

and geometrical features. These events were used further in letter hypotheses and word formation.

The system was tested in six separate experiments. In the first experiment, only one writer was involved for both training and testing. The training set consisted of 66 words which were conformed with the constraints of being horizontal and not slanted. The test set consisted of 64 words. The system correctly recognized 77% of the words, rejected 14% and incorrectly recognized 9%. The second experiment was designed to exercise learning capabilities and the test was run on the same data as in experiment 1. The result was similar to that from experiment 1. The third experiment used a larger lexicon than that in experiment 1 and the results showed some deterioration compared with the case with the small lexicon. In the fourth experiment, the input was from a second writer and no slant restrictions were placed. The results were 54% correct for the first choice and 61% correct for the top-two choice. In the fifth experiment, the input was from the third writer, and the recognition increased to 63% correct for the first choice and 65% for the top-two choice.

5.1.2. *Segmentation using singularities and regularities.* A segmentation algorithm based on singularities and regularities was proposed by Chen *et al.*[27] In this algorithm, the singularities and regularities, sometimes referred to as islands and bridges, were extracted from a word image after a sequence of preprocesses, namely, slant normalization, morphological filtering and stroke width estimation (see Fig. 15). The morphological filtering consisted of a closing and an opening operation with a 3×3 disk as the structuring

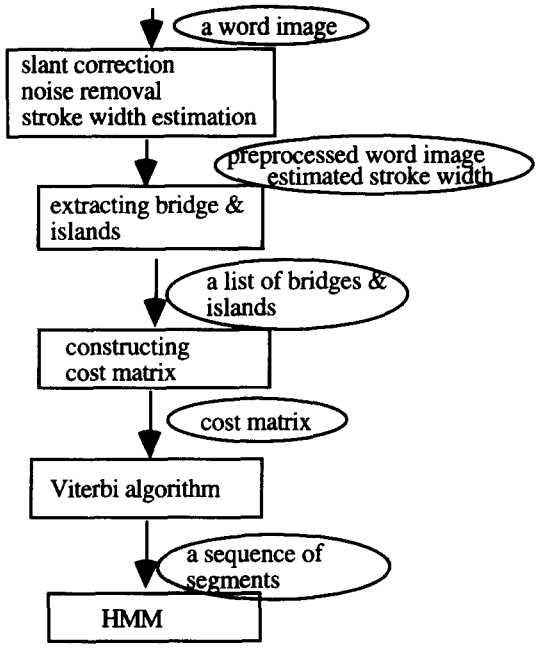


Fig. 15. Word segmentation using bridge and islands.

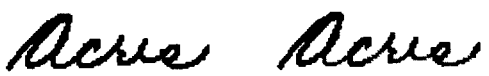


Fig. 16. An example of segmented word image using bridge and islands: (a) original image, (b) islands are in black and bridges are in gray.

element. The objective of the morphological filtering was to remove noise, fill up gaps and smooth contours.

After the morphological filtering, every on-pixel was considered in one of these two classes, islands and bridges. Islands corresponded to vertical strokes and bridges were the points connecting the islands (see Fig. 16). The islands and bridges were extracted through morphological operations. A morphological opening operation with the structuring element of a vertical line, whose size was decided by the estimated stroke width, was applied to obtain the islands and then a closing by a 5 × 5 structuring element of a horizontal line (one pixel wide) was used to group the islands which were close to each other. The bridges, which were the candidates of segmentation points, were found by the complementation of the islands. The segmentation points should exist only at bridges. The segments were found by traveling through the entire word from leftmost island to the rightmost island using the Viterbi algorithm.[29] The cost matrix was defined as:

$$\text{cost}(i,j) = \begin{cases} \textit{motionless}(i) & i = j \\ \textit{bridge}(i,j) & \textit{there is a bridge from island } i \text{ to } j \\ \textit{boat}(i,j) & \textit{otherwise} \end{cases}$$

where

$$\text{bridge}(i,j) = \text{islands} - (j-i),$$

$$\textit{boat}(i,j) = \begin{cases} \textit{islands} \times (j-i) & j > i \\ (\textit{islands})^2 \times (i-j) & i < j \end{cases},$$

$$\textit{motionless} = (\textit{islands})^3$$

islands = total number of islands obtained from the steps above.

This cost matrix gave minimal cost to the paths that contained bridge points. It is clear that each segment produced by this approach contained either a character, a partial character, or two characters. Specifically, the output from the segmentation algorithm satisfied the following criteria:

- each complete character could be segmented into at most three parts, and
- each segment contained at most two complete characters.

The segments of a word were then sent to a HMM for word recognition. The system was tested on a USPS database. There were 1583 word images, and among them 627 were unique words. The writing styles of these words were totally unconstrained. The data set was divided into a training set, which contained 1489 images, and a test set, which contained 94 images. The recognition reached 68% correct for the top one hypothesis and 84% correct for the top five hypotheses.

5.1.3. *Simon's segmentation algorithm.* A typical example of word recognition methods pertaining both global features and local segmentation features was described by Simon.[4] Simon's segmentation approach illustrated in Fig. 17 was also based on regular and singular features. First a chain graph, i.e. skeletons of characters, was obtained from a binary word image by a thinning process. The segmentation of a word was performed by finding the axis and tarsi of the word. The axis, also called regularity, of a cursive word was defined as the shortest path from one extremity (left) to the other (right) that remained in the body of the word. The tarsi were obtained by complementing the axis. The singularities of a word image include forks or crossings, line extremities and loops. Figure 18 shows an example of chain graph, axis and tarsi. The segmentation of a word image could be performed by combining axis and tarsi features. All the features including axis and tarsi, ascenders and descenders, the concavities and convexities of the chain graph, etc., were represented in a symbolic description chain (SDC) for word recognition. The detail of the recognition processes can be found in reference (4).

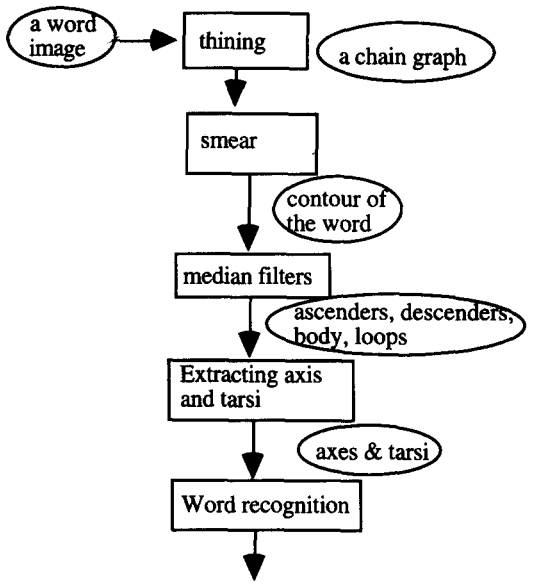


Fig. 17. Axis- and tarsi-based segmentation and recognition.

The experiments were conducted on a data set of 25 words written by eight writers. The correct recognition reached as high as 91%.

5.1.4. *Finding optimal segments using AI search.* As we pointed out earlier, cursive word segmentation usually consists of two steps, presegmentation and segmentation. At the presegmentation step, a word image is cut to presegments. If we consider a complete set of cutting points inside a word as a single hypothesis, then every one of the possible hypotheses can be evaluated by means of a set of functions defining the optimum criterion. Obviously the evaluation of all the possible segmentation could not be realized in a reasonable computing time. Balestri and Masera[30] proposed to use the well-known A* search algorithm to find the best segmentation result. The algorithm is illustrated in Fig. 19.

The key issue in the A* algorithm is to define the two heuristic cost functions $g(n)$ and $h(n)$. Balestri and Masera suggested to consider the set of all possible segmentation as a weighted graph, and then the objective became to find the path with minimum weight or cost. The two heuristic functions $g(n)$ and $h(n)$ at node n were defined by:

$$\hat{g}(\text{start}) = 0$$

$$\hat{g}(n) = \hat{g}(n-1) + \frac{w}{p[\text{width} = w^* p[\text{pos} = s]}$$

$$\hat{h}(\text{start}) = N$$

$$\hat{h}(n) = \hat{h}(n-1) - w$$

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n),$$

where N is the width of the bounding box of the word, w the distance between the last hypothesized cutting point and the next one, i.e. the width of the next presegment, s the position of the last cutting point, $p[\text{width} = w]$ the probability of having a w-wide character and $p[\text{pos} = s]$ the probability of the sth pixel of the word to be a correct segmentation point. The probability $p[\text{pos} = s]$ was estimated by means of a simple set of features, such as the distance between uppermost and lowermost black pixels of the word detected on a vertical line passing through the cutting point, or the distance from these two pixels to the baseline detection. A "score" was used for every column of pixels in the region to be split. $p[\text{width} = w]$ was computed based on the statistics of the width of the handwritten single character.

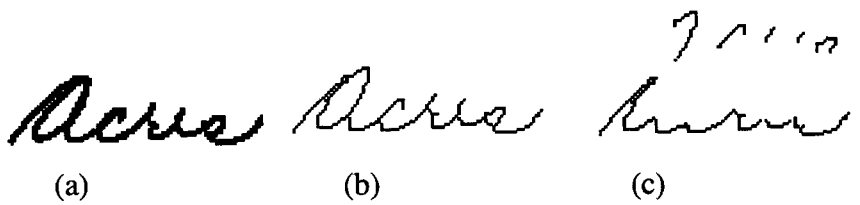
In order to improve the computing time, Balestri and Masera suggested an additional criterion for the elimination of some branches of the tree expanded by the A^* algorithm when the number of open nodes exceeded a predefined value. The additional criterion was

$$\langle \hat{f} \rangle = \frac{1}{N} \sum_{i=1}^{N} \hat{f}(i),$$

where i is a node on the "open list" of A* and N the total number of nodes in the "open list". If node i on the "open list" had $\hat{f}(i)$ greater than $\langle \hat{f} \rangle$, it was discarded. Note that after this an additional criterion was added, the admissibility of the A* algorithm was no longer ensured, however, Balestri and Masera claimed that in practical cases no performance deterioration had been observed. Unfortunately, no experiments were reported in reference (30).

5.2. *Segmentation-recognition constrained to lexicons*

This subsection introduces word segmentation techniques constrained to lexicons. We discuss word seg-



(a) (b) (c)

Fig. 18. An example of word segmentation using axis and tarsi. (a) The original image; (b) the chain graph; (c) shows the regular axis at the bottom and the tarsi at the top.

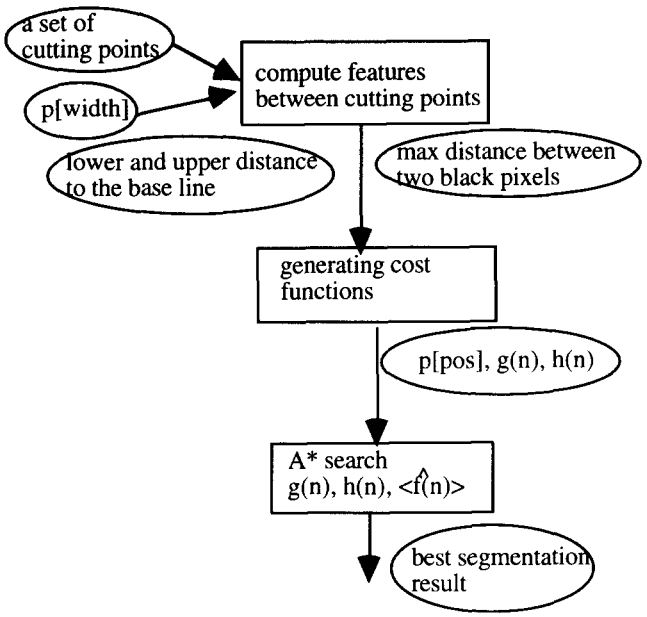


Fig. 19. Word segmentation based on the A* heuristic search.

mentation techniques from three research groups. These techniques are common in the sense that they all split a handwritten word image into presegments that constitute characters, and they all use lexicon knowledge in the segmentation-recognition processes. The techniques differ in the processes that generate presegments and combine presegments into characters, and subsequently, words.

5.2.1. *OuladJ etal.'s algorithm.* This algorithm performed segmentation-recognition through the matching of presegments with words in a lexicon.[11] The segmentation process was embedded in the word recognition process (see Fig. 20). A word image was represented in four-directional codes sequence which was scanned along the writing direction. During the scanning, every occurrence of a transition from a maxima to a minima was considered a possible segment point. The sequence up to the current segmentation point was matched with the words in the database, which were organized in a decision tree. A node in the decision tree represented one letter of the alphabet and the nodes on the path from the root of the tree to a leaf represented a word. Each letter was coded by a pair of primitives (P_1, P_2), where P_1 was s sequence of directional codes defining temporal evolution of a letter using a four-directional coding and P_2 represented *a priori* knowledge on letter shape introduced in the database. P_1 was used for classification. Values assigned to P_2 were equal to 1, 2, 3 or 4 with respective to a letter had: no vertical line such as "e", one ascender such as "h", one ascender and one descender such as "j" or one descender such as "g". If a match was possible, primitive P_1 was detected and a letter associated with

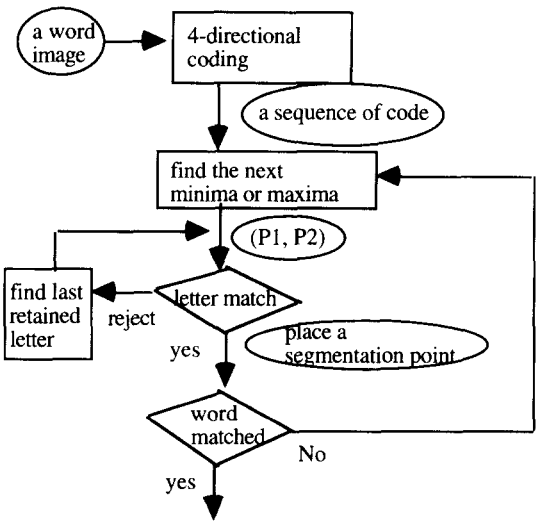


Fig. 20. Word segmentation through matching with lexicon.

pair (P_1, P_2) was predicted. After a letter was identified, a segmentation point was retained, and the next letter in the input word was searched. All the segmentation points were stored so that when a letter was rejected or a letter was found to match a leaf in the dictionary, the process could be backtracked to attempt to predict other letters from the last retained segmentation point. Thus, the algorithm led to a list of candidate words discriminated by a criterion defined as the summation of the recognition weights assigned to each identified letter. An experiment was conducted on a data set produced by five scriptors. A dictionary of 100 words

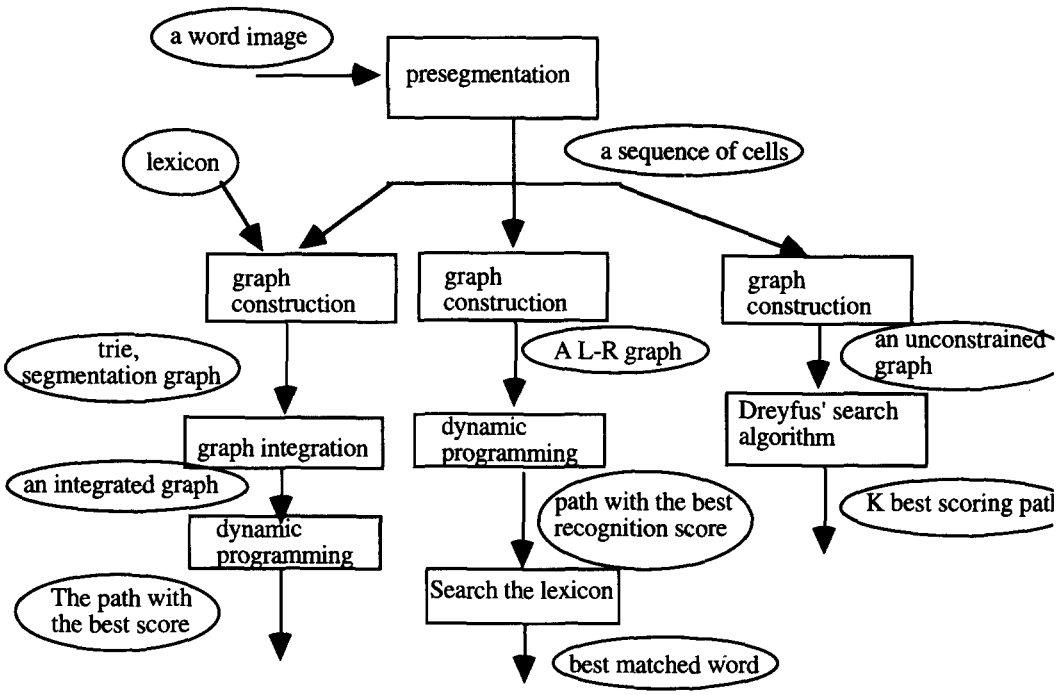


Fig. 21. Graph-search-based word recognition algorithms.

was used in the experiment and the correct recognition rate reached 90%.

5.2.2. *Nohl et al.'s algorithms.* Nahl *et al.*[31] proposed three graph-search-based approaches for segmenting-recognizing a word image (see Fig. 21). Before the graph search, the following preprocessing was applied to a word image. A word image was first divided into a set of nonintersecting cuts referred to as a sequence of cells representing candidates for divisions between successive characters in the word image. The cuts were numbered 1 to N from left to right, where the left-side edge of the word image is numbered as 0 and the right side edge as N + 1. A segmentation of a word image was a sequence of segments each consisting of one or more adjacent cells, and in general a number of segmentation can be made by combining the cuts (see Fig. 22).

The three methods for finding the correct segmentation paths were:

(1) finding the best-scoring segmentation for each lexicon entry;

(2) combining the lexicon and segmentation alternatives to form a single optimal path problem, in which there existed a one to one correspondence between paths through the graph and legal segmentation of lexicon entries; and

(3) finding the K best segmentation/interpretations independent of lexicon and then perform post-processes to eliminate interpretations not found in the lexicon.

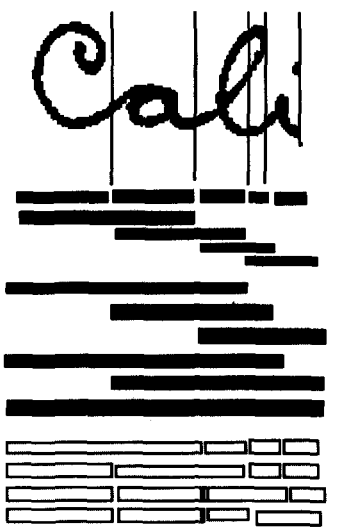


Fig. 22. All possible segmentation from cells by Noel, etc. The black bars represent the possible segments which are combinations of cells. The hollow bars in each line represent a segmentation into four characters.

The first method employed the Viterbi dynamic programming algorithm to find the best segmentation and the word interpretation was constrained to match a particular lexicon entry. This process was performed once for each word in the lexicon and produced a recognition score for the word. Since a segment consists of at least one cell, the segmentation process was to find the best combination of the cells. A segment of the

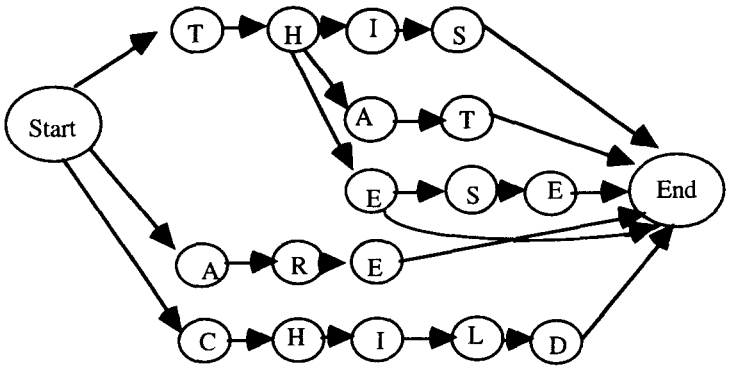


Fig. 23. A lexicon trie.

word image was identified by specifying the pair L-R, where L is the index of the cut at the left side of the segment and R the index of the cut at the right side of the segment. To assist the computation, a graph was constructed by using the following two rules:

Allowed Nodes:

Start
End
(L-R, C) where $1 \leq$ R-L ≤ 4

Allowed Arcs:

(L-R, C)$\rightarrow$(L′-R′, C′), where L′ = R,

where C represented a particular character within the word, and the arrowhead arc indicated that character C′ in the segment L-R may be followed by character C′ in the segment L′-R′. Note that in general two nodes (L-R, C) and (L-R, C′) could occur with C′ $\neq$ C. The "Start" node was the origin of arcs terminating on all nodes corresponding to the first character of an interpretation; and the "End" node was the termination of arcs from all nodes corresponding to the last character of an interpretation. Each node was associated with a score P representing the prpbability that segment L-R should be interpreted as character C. Thus, every legal segmentation of the word image had a corresponding path in the graph. The score for any particular segmentation was the product of the score $\{P_i\}$ associated with the nodes along the corresponding path, and the best segmentation was computed using the Viterbi algorithm to maximize $\prod_i P_i$. The word recognition system evaluated each word in the lexicon and produced a score for each word and the lexicon word with the highest score was taken as the most likely interpretation. The advantage of this approach is that it provides a score for each word in the lexicon, and the graph constructed using the dynamic programming is relatively small. The primary disadvantage is the computational inefficiency in solving a dynamic programming problem for each word in a potentially large lexicon.

The second method integrated the constraints imposed by the lexicon into the dynamic programming problem so that the highest scoring segmentation and interpretation matching the lexicon were determined simultaneously. It began by representing the lexicon as a special graph called a "trie," in which each word was mapped to a path beginning at "Start" and ending at "End," and words having common initial substrings shared the corresponding nodes in the trie. Figure 23 shows an example of Lexicon Trie. The second graph used in this method was called the integrated graph. The integrated graph was the "product" of the lexicon trie with the segmentation graph described above. The nodes were labeled by a pair (segment, lexicon-trie node). For each rank R of the integrated graph, nodes were created by taking all combinations of segments from rank R of the segmentation graph and character interpretations from rank R of the lexicon trie. Arcs were filled in between the parent and the children in both the segmentation graph and the lexicon trie. The integrated graph contained all allowed segmentation of every lexicon word and any path from "start" to "end" corresponded to an allowed segmentation of a word in the lexicon. Each node in the graph was associated with a number equal to the recognition score for the corresponding image segment and class. The path with the largest score was the preferred segmentation and interpretation. This method produces only the top-choice segmentation and interpretation, however, it may significantly improve real time performance when the character recognition algorithm is reliable.

The third method computed the unconstrained K best paths. It began by constructing an unconstrained segmentation-recognition graph, in which each node represented a possible character interpretation and was associated with a recognition score. There was a one to one correspondence between paths in the graph and allowed segmentation and interpretations of the word image. A search algorithm was used to look for the K best scoring paths in the unconstrained graph. The computation may be terminated at any desired value of K. This method is most attractive for applications where segmentation and recogni-

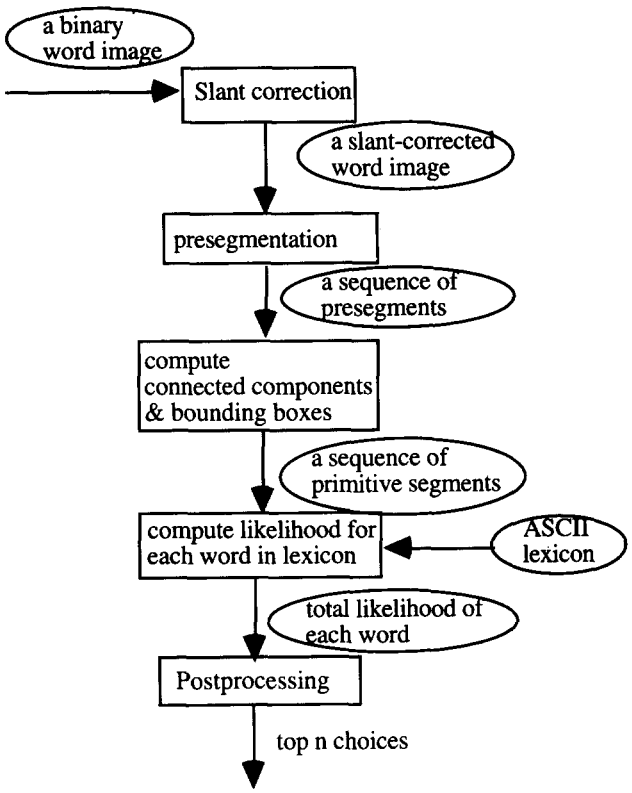


Fig. 24. Handwritten word segmentation-based total likelihood.

tion are relatively reliable, or where lexicons are large and dense in the space of possible interpretations. No experiments were reported in reference (31).

5.2.3. *Kimura and Shridhar's algorithm.* This algorithm used a lexicon to direct word segmentation and recognition.[32,33,34] The computational steps are illustrated in Fig. 24.

A binary word image was first being processed by slant correlation. The character segmentation consisted of presegmentation and segmentation. During the presegmentation, the algorithm first detected the minimal points of the upper contour of a given input word image. The presegment points were local minima that were deep from the adjacent local maxima. If a minimal point was not open vertically upward, the algorithm shifted to the right or the left of the minimal point to look for an optimum point with regard to an evaluation function of total run length and the counts, and took the optimum point as the presegment point. The word image was then physically split by vertical lines passing through the pre-segmentation points. The connected components and their bounding boxes in the split word image were computed. Each connected component, referred to as a primitive segment, was part of a character. The bounding boxes, which were usually disjoint and did not include parts of other connected components, were sorted from left to right according to the location of their centroids. If two or more boxes showed the same x coordinates as the centroids, they were sorted top to bottom. For each word in a lexicon, the primitive segments of the input word were merged and matched against the characters in the lexicon word so that the average character likelihood was maximized using the dynamic programming. The segmentation problem became a simple Markov process which was an assignment of boxes to letters. To calculate the total likelihood of character recognition, a high speed feature extraction routine was applied to each set of primitive segments belonging to a character. A feature vector was constructed by utilizing the bounding boxes of the primitives and the intermediate word feature extracted preceding the segmentation recognition process. Due to the height normalization for the likelihood calculation, a separate consistency check of relative character height was used in a postprocessing to the result of segmentation-recognition. This segmentation-recognition and the post-processing steps were repeated for all lexicon words and the words were sorted according to the linear combination of the average character likelihood and the height consistency measure.

The system was tested on 612 city name images from the USPS mail pieces. The number of cities was 27 and the lexicon had 55 words including the variation and the abbreviation of the city names. The data was

divided into two sets, training and test. The recognition on the training set reached as high as 93.1% and on the test set 85.3%.

6. CONCLUSION

We have presented an overview of important techniques in segmenting handprinted words, handwritten numerals and cursive words. The handprinted word segmentation problem is relatively easier and many techniques for segmenting machine-printed characters[1] may be applicable to this problem. In fact the two algorithms we presented in this paper were extensions of algorithms for segmenting machine-printed characters. Segmentation and recognition of handwritten numerals has important applications such as postal ZIP-code recognition, bank cheque reading, and credit card slip processing. The algorithms we introduce in this paper were developed for the first application and the segmentation processes were driven by domain-specific knowledge. The cursive word segmentation and recognition is the most difficult problem and has just begun to attract research attention.[35-46] In the recognition of cursive words, the segmentation process is often closely coupled with recognition. The control between a segmentation process and a recognition process is important since we know that in order to recognize a letter one must know where it begins and where it ends, whereas to isolate a letter one must recognize it first.

We believe that because of the problem complexity, a good handwritten word recognition system should employ both global and segmentation-recognition strategies. Therefore, handwritten character segmentation techniques will continue to play an important role in the handwritten word recognition process. In fact we are seeing more and more algorithms employing a combination of global and segmentation-recognition approaches in handwritten word recognition.[4,47] We are also seeing more and more algorithms employing domain-specific knowledge to achieve high performance. However, most such systems have domain specific knowledge buried in the programs and knowledge is used for tuning parameters and thresholds. We believe that in order to have a robust and reliable system, it is important to represent domain-specific knowledge explicitly using well-known AI techniques.

Acknowledgement—The work is supported in partial by the research grant and fellowship from The University of Michigan Rackham Graduate School.

REFERENCES

1. Yi Lu, Machine-printed character segmentation, *J. Pattern Recognition* **28**(1), 67–80 (1995).
2. S. Mori, C. Suen and K. Yamamoto, Historical review of OCR research and development, *Proc. IEEE*, **80**(7), (July 1992).
3. C. Y. Suen, M. Berthod and S. Mori, Automatic recognition of handprinted characters—the state of the art, *Proc. IEEE*, **68**(4), 469–487 (April 1980).
4. J.-C. Simon, Off-line cursive word recognition, *Proc. IEEE* **80**(7), 1150–1161 (1992).
5. C. C. Tappert, C. Y. Suen and T. Wakahara, The state of the art in one-line handwriting recognition, *IEEE Trans. PAMI* **12**, 787–808 (August 1990).
6. T. Wakahara, H. Murase and D. Odaka, On-line handwriting recognition, *Proc. IEEE* **80**(7), 1181–1194 (July 1992).
7. J. M. Bertille and M. El Yacoubi, Global cursive postal code recognition using hidden Markov models, *Proc. 1st Europ. Conf. Postal Technol.* 129–138 (1993).
8. M. Brown, Cursive script recognition, Ph.D. dissertation, University Michigan, Ann Arbor, Michigan (1981).
9. R. Farag, Word-level recognition of cursive script, *IEEE Trans. Comput.* **C-28**, 172–175 (1979).
10. T. K. Ho, J. J. Hull and S. N. Srihari, Combination of structural classifiers, in *Pre-proc. SSPR-90*, 123–136 (1990).
12. L. S. Frishopf and L. D. Harmon, Machine reading of cursive script, *Information Theory, Symposium on Information Theory–London* C. Cherry, ed., pp. 300–316. Butterworths, Washington (1961).
13. P. Mermelstein and M. Eden, Experiment on computer recognition of connected handwritten words, *Inf. Control* **7**, 255–270 (June 1964).
14. R. W. Ehrich and K. J. Koehler, Experiment in the contextuel recognition of cursive script, *IEEE Trans. Comput.* **C24**, 182–194 (February (1975).
15. A. S. Fox and C. C. Tappert, On-line external word segmentation for handwriting recognition *Proc. Third Int. Symp. Handwriting Comput. Appl.* pp. 53–55. Montreal, Canada (July 1987).
16. K. Hayes, Reading handwritten words using hierarchical relaxation, Technical Report, University of Maryland, TR-783 (1979).
17. J. J. Hull, Word shape analysis in a knowledge-based system from reading text, *Second IEEE Conf. Artif. Intell. Appl.* pp. 114–119. Miami Beach, Florida (1985).
18. R. G. Cassey and J. van Horne, Segmentation of touching characters in postal addresses, *U.S. Postal Service 5th Adv. Technol. Conf.* Vol. 3, pp. 743–754, Washington DC, (November 1992).
19. E. Lecolinet and J. V. Moreau, A new system for automatic segmentation and recognition of unconstrained handwritten ZIP codes, *Proc. Sixth Scandinavian Conf. Image Analysis* **1**, 585–592 (1989).
20. P. Gader, M. Mohamed and J.-H Chiang, Segmentation-based handwritten word recognition, *U.S. Postal Service 5th Adv. Technol. Conf.* Vol. 3, pp. 215–236. Washington DC (November 1992).
21. R. Fenrich and K. Krishnamoorthy, Segmenting diverse quality handwritten digit strings in near real-time, *Proc. 4th Adv. Technol. Conf.* 523–537, (1990).
22. N. W. Strathy, C. Y. Suen and A. Krzyzak, Segmentation of handwritten digits using contour features, *Second Int. Conf. Document Anal. Recognition* 577–580 (October 1993).
23. M. Cheriet, Y. S. Huang and C. Y. Suen, Background region-based algorithm for the segmentation of connected digits, *Proc. 11th Int. Conf. Pattern Recognition* 619–622 (1992).
24. F. Kimura and M. Shridhar, Recognition of connected numeral strings, *Proc. 1st Int. Conf. Document Anal. Recognition* 731–739 (1991).
25. F. Kimura and M. Shridhar, Segmentation-recognition algorithm for Zip code field recognition, *Mach. Vis. Appl.* **5**, 199–210 (1992).
26. O. Matan, H. S. Baird, J. Bromley, C. Burges, J. Denker, L. Jackel, Y. Le Cun, E. Pednault, W. Satterfield, C. Stenard and T. Thompson, Reading handwritten digits:

A ZIP code recognition system, *IEEE Comput.* 59–62 (July 1992).
27. M. Y. Chen, A. Kundu, J. Zhou and S. N. Srihari, Off-line handwritten word recognition using Hidden Markov Model, *U.S. Postal Service 5th Adv. Technol. Conf.* pp. 563–577 Washington, DC (November 1992).
28. R. M. Bozinovic and S. N. Srihari, Off-line cursive script word recognition, *IEEE Trans. PAMI*, **11**, 68–83 (1989).
29. G. D. Forney Jr, The Viterbi algorithm, *IEEE Proc.* **61**, 268–278 (March 1973).
30. M. Balestri and L. Masera, A system for isolating characters in cursive script, *Signal Processing IV: Theories and Applications*, Lacoume *et al.* eds, pp. 845–846. Elsevier Science Publishers, B. V. North-Holland (1988).
31. C. R. Nohl, C. J. C. Burges and J. I. Ben, Character-based handwritten address word recognition with lexicon, *U.S. Postal Service 5th Adv. Technol. Conf.* Vol. 3, pp. 167–182. Washington, DC (November 1992).
32. F. Kimura, M. Shridhar and Z. Chen, "Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words, *Second Int. Conf. Document Anal. Recog.* 18–22 (1993).
33. F. Kimura, M. Shridhar and N. Narasimhamurthi, Lexicon directed segmentation-recognition procedure for unconstrained handwritten words, *Proc. Third Int. Workshop Frontiers Handwriting Recognition*, Buffalo (1993).
34. F. Kimura, S. Tsuruoka, M. Shridhar and Z. Chen, Context directed handwritten word recognition for postal service applications, *Proc. Fifth Adv. Technol. Conf.* Washington, DC (1992).
35. J. T. Favata and S. N. Srihari, Off-line recognition of handwritten cursive words, *Proc. SPIE Symp. Electron. Imaging Sci. Technol.* San Jose, California (1992).
36. R. Fenrich, Segmentation of automatically located handwritten words, *Proc. Int. Workshop Handwriting Recognition* 33–44. Bonas, France (1991).
37. A. M. Gillies, Cursive word recognition using hidden Markov Models, *U.S. Postal Service 5th Adv. Technol. Conf.* Vol. 3, pp. 557–562. Washington, DC (November 1992).
38. M. Gilloux and M. Leroux, Recognition of cursive script amounts on postal cheques, *Proc. 5th USPS Adv. Technol. Conf.* 545–556 (1992).
39. M. Gilloux, M. Leroux and J.-M. Brittle, Strategies for handwritten words recognition using hidden Markov models, *Second Int. Conf. Document Anal. Recognition*, 299–304 (1993).
40. D. Guillevic and C. Y. Suen, Cursive script recognition: A fast reader scheme, *Second Int. Conf. Document Anal. Recognition* 311–314 (1993).
41. A. Kaltenmeier, T. Caesar, J. M. Gloger and E. Mandler, Sophisticated topology of hidden Markov models for cursive script recognition, *Second Int. Conf. Document Anal. Recognition* 139–142 (1993).
42. A. Kundu, Y. He and P. Bahl, Recognition of handwritten word: first and second order hidden Markov model based approach, *Pattern Recognition* **22**, 283–297 (1989).
43. E. Lecolinet and J. V. Moreau, Off-line recognition of handwritten cursive script for the automatic reading of city names on real mail, *Proc. 10th ICPR* Vol. 1, pp. 674–675. Atlantic City, New Jersey, 16–21 June (1990).
44. M. Leroux, J. C. Salome and J. Badard, Recognition of cursive scripts words in a small lexicon, *Proc. 1st Int. Conf. Document Anal. Recognition*, 774–782 (1991).
45. P. Morasso and S. Pagliano, A neural architecture for the recognition of cursive handwriting, *Fourth Italian Workshop Parallel Architectures Neural Networks* 250–254 (1991).
46. C. C. Tappert, Cursive script recognition by elastic matching, *IBM J. Research develop.* **26**(6) (November 1982).
47. E. Lecolinet and J. P. Crettez, A grapheme-based segmentation technique for cursive script recognition, *Proc. 1st ICDAR* 740–748 (1991).

About the Author—YI LU graduated with a diploma majored in mathematics from Shanghai Teacher's College, Shanghai, China, in 1980. She received a M.S. degree in Computer Science from Wayne State University, Detroit, Michigan, in 1983 and a Ph.D. degree in Computer, Information, Control Engineering from the University of Michigan, Ann Arbor, Michigan, in 1989. From 1989 to 1992, she was a research scientist at the Environmental Research Institute of Michigan, Ann Arbor, Michigan. Currently she is an assistant professor at the Department of Electrical and Computer Engineering at the University of Michigan–Dearborn. Her research interests include computer vision, pattern recognition and artificial intelligence. She has publications in the areas of low-level vision processes, knowledge-based computer vision, handwritten digit recognition, machine-printed character segmentation and recognition, medical image analysis, and knowledge integration. Currently she is an associate editor for Pattern Recognition.

About the Author—M. SHRIDHAR received his MS in Electrical Engineering in 1967 from the Polytechnic University of New York and his Ph.D. in Electrical Engineering in 1970 from the University of Aston in Birmingham, U.K. From 1969 to 1985 he was a faculty member in the Electrical Engineering Department of the University of Windsor, Canada. Since 1985, he has been with the University of Michigan–Dearborn, where he is currently a professor and chairman of the Electrical and Computer Engineering Department. Dr Shridhar's research interests are in the areas of digital signal processing with applications to control systems, image processing, computer vision and pattern recognition. Dr Shridhar has served as a consultant to Ford Motor Co. (dynamic models and expert systems for control of glass furnaces), General Motors Corporation (Control of six-axis robot for assembly-line part retrieval), CGA-Alcatel, France (digit recognition in revenue documents), and Diffracto Ltd, Canada (providing services in the areas of machine vision inspection, robotic vision, modeling and control). He is currently on contract with USPS for development of handwritten address recognition systems and with Royal Design and Manufacturing, Michigan, for development of tool wear analysis with machine vision. He has received grants from NSF and NSERC, Canada, for support of his research activities.