A Format-Driven Handwritten Word Recognition System

Xia Liu and Zhixin Shi
Center of Excellence for Document Analysis and Recognition
State University of New York at Buffalo, Buffalo, NY 14260, U.S.A.
E-mail: zshi@cedar.buffalo.edu

Abstract

A format-driven word recognition system is proposed for recognition of handwritten words. Unlike most traditional handwritten word recognizers being given a set of target words as lexicon, we assume that our system is given a set of format descriptions other than lexicon words. Applications of the proposed system include recognition of relatively more important keywords such as postal codes, titles or trademarks. The format descriptions are in terms of the lengths of the keywords, the types of the characters in the keywords and positional informations. Due to the important role of the keywords in the applications, the recognition expectations in terms of recognition rate and accuracy are usually higher then lexicon-driven word recognizers.

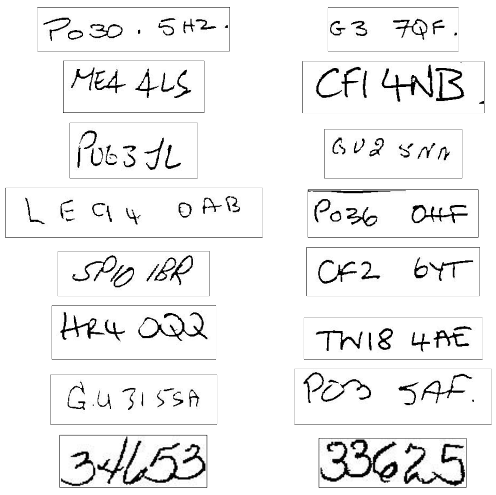


Figure 1. Character string samples

1. Introduction

Most of the conventional handwritten word recognition methods found in the literature are lexicon-driven methods[1, 4] in which one is given a handwritten word image with a list of possible target words - the lexicon. The recognition of the word image is basically a matching process. Each algorithm gives a way of matching the word image with the given text words in the lexicon. The best match gives the recognition result. This type of handwritten word recognition has many applications including recognition of street, city and state words in postal processing. Another example is bank check recognition. In the first example, the lexicon is generated by querying a postal database using other information such as recognized post codes and street numbers. The existence of a lexicon plays a crucial role in lexicon-driven handwritten word recognition algorithms.

To the other extreme, the methods for recognition of a handwritten word without lexicon are call lexicon-free methods. In general, lexicon-free handwritten word recognition is a very difficult task. For applications using lexicon-free approach such as automatic data entry from scanned images, a very strong linguistic post-processing is needed to get the job done.

For our applications in postal address interpretations, we often need to recognize another type of handwritten word images which we classify as images of *key words*. The typical example are the images of post codes(see Figure 1). There is no way to generate lexicons for recognition of these images. But they do have some other kind of assisting informations. In the case of US zip code, the length of zip codes is alway 5 or 9 characters and the type of characters in the zip code is numeral. While in the case of Canadian post codes, the length is fixed 6 but the characters in the post codes are alpha characters and numerals placed alternatively. The most complex post codes are UK post codes in which there are multiple formats specified for their lengths and positions of alpha and numeral characters. The recognition of this kind of handwritten words has gained much attention due to its pivotal role in applications such as the postal address interpretation.

In this paper we propose a format-driven handwritten word recognition system. Unlike lexicon-driven handwritten word recognition systems, we assume that our system is given a set of format descriptions other than lexicon words. The format descriptions are in terms of length of the words, type of the characters in the images and positional informations. The proposed system consists of several document analysis modules, namely, a preprocessing module, a segmentation module and a recognition module. The preprocessing module includes handwritten image quality enhancement, slant normalization, connected component analysis, identifying sub-words with touching characters and estimation of number of characters in the sub-words. The segmentation module is built with a segmentation algorithm based on a thorough stroke analysis using contour representation of the strokes. In the recognition module, a search algorithm incorporates the given format with a high performance character recognizer to identify the required characters specified in the format.

Applications of the proposed system are recognitions of relatively more important keywords such as postal codes, titles or trademarks. Due to the important role of the keywords in the applications, the recognition expectations in terms of recognition rate and accuracy are usually higher then lexicon-driven word recognizers.

2. Our approach

There are several approaches to handwritten word recognition and segmentation [10]. In *segmentation-based* approach, the word image is segmented into isolated entities each is a complete image of an individual character. Then each entity is sent to a character recognizer for a recognition result. Methods using segmentation-based approach separate the segmentation from the recognition. The segmentation for getting each individual character is usually based on heuristics and done before the recognition of each characters. The previous works related to our research are those on numeral string recognitions. Most numeral string recognition methods are segmentation-based approaches. There is no lexicon as that in lexicon-driven word recognitions. Numeral string recognitions belong to format-driven word recognitions though their format informations are as simple as given lengths of the strings and the type of the characters in the strings is numeral only.

Another approach is called *segmentation-free* in the sense that multiple segmentation candidates are validated by an isolated character recognizer. This approach, sometime also called *oversegmentation* is more rational than segmentation-based approach in the case when heuristics prevail, at the price of a higher computational complexity. Works using *segmentation-free* approach are found in [2, 7].

In this paper we apply both segmentation-based and segmentation-free methods. Works using this approach are found in [6, 3, 4].

We build a preprocessing module starting with an image quality enhancement algorithm to correct as much as possible the problems such as holes and broken stroke pieces due to imperfection of the binarization process. Then we apply a slant normalization algorithm to correct slantness of the characters in the image. We use the segmentation-based approach on a higher level. We do not try to segment the handwritten word to entities each representing a *single character*. Rather we try to cut the long handwritten word down to smaller substrings – blobs which may be of a single character or several characters. The main efforts are put in getting *all the fragments of one character* into the same blob. In most cases a character is in one connected component. In the cases when there are broken strokes caused by binarization process or multiple stroke characters, heuristic rules are used to group them together. Since the grouping is done on this higher level, the heuristic rules can be built with very high reliability. Hence we do not have to use recognition to help in this stage. Special punctuation marks and delimiters are also identified in the preprocessing module.

For each segmented blob, the number of characters in the blob is estimated by a horizontal intersection method together with the blob size information. The blobs with number of characters greater than one will be sent to the segmentation module.

The segmentation module is built with a segmentation algorithm based on a thorough stroke analysis using contour representation. The segmentation of each connected or touching characters is done by segmenting the characters with splitting the contour representations. For each blob, it first detects the special contour points as splitting points. Then the split contour pieces are grouped by a grouping algorithm consisting of a set of heuristic rules. The grouping algorithm takes the information including the number of characters in the blob, slant of the strokes, locations of each split contour pieces and the type of the required characters.

When we estimate the number of characters in the preprocessing module, we try to identify the blobs with single character and send them to the recognizer directly. We take a very conservative approach to get the identification with high success rate. In the segmentation module we go with an approach similar to the segmentation-free methods. But instead of going through a complicated search algorithm as presented in [2, 3] and [6], we just vary the grouping of the split contour pieces to get the characters in the blob by varying the number of characters around the estimated number. The optimal result is validated by a high accuracy character recognition. This design approach is based on the following facts. The accurate rate in estimation of the number of characters in a blob is high so that the real number of characters is within at most one away from the estimated number. The character recognizer, GSC used in the system, is not only highly accurate to accept valid characters but also being able to reject with low confidence the invalid characters. Therefore the number of calls to the recognizer is minimized to within at most three passes for each blob. The optimized recognitions will be the final results.

In section 3 we shall describe preprocessing module, including image enhancement and slant normalization, connected component analysis, blob generation, punctuation marks and delimiters filtering and estimation of number of characters in a string. In section 4 we present a novel approach to the problem of segmentation of touching characters. In section 5 we discuss the use of recognition to the segmentation in a loose fashion as segmentation-free approach.

3. Preprocessing

3.1. Image enhancement and slant normalization

To solve the over fragmentation problem caused by binarization process, we apply two types of image processing algorithms. The first algorithm that we developed in [8] is called region growing method. It apples a moving window on the center of the detected strokes of handwritten characters to fill up the holes and gaps. The conservative filling process is done carefully by adding some foreground pixels to the image based on the geographic properties of the local area around a character stroke. The filling process is also designed for an efficient implementation.

The second algorithm is a straightforward process which we rather call it "blind grouping". The motivation of using it is from an observation of our postal address images showing horizontal background stripes. These stripes as gaps break the handwritten characters into fragments aligned vertically. We blur the input image of a handwritten word to get a masked image. The blurring is done by prolonging each vertical foreground runlength by one pixel. Therefore in the vertically blurred image, up to two pixel gaps in the original image will be glued together. We do not use horizontal blurring as it may cause close connected components to touch each other.

One of the typical variations of writing styles showing in handwritten words is slant - strokes being rotated clockwise or count-clockwise with a small angle. Slant normalization algorithms are designed to straight the strokes up. Experiments have shown clear improvement for recognition of the words. The slant normalization algorithm we apply in our system is modified from the one in [4]. It takes a chaincode representation of the character strokes. The slant angle is calculated from tracing the strokes aligned within an angle around vertical direction. The calculation takes in its consideration all the related informations including the sizes of the strokes. Using the detected angle, the image is corrected by a sheer transform on the chaincode representation. One other benefit from using chaincode representation is that the connected components is built in it. Small pepper noises can be removed easily.

3.2. Grouping sub-words

Sub-word grouping is one of the key features to make our method different from others. We divide a handwritten word to sub-words – blobs each containing complete character or characters from touching. The special punctuation marks and delimiters are also detected and put into separate blobs. Since we do not require each blob contain only one character, i.e. there is no splitting is needed, the only difficulty will be grouping of fragments from broken characters or multiple stroke characters.

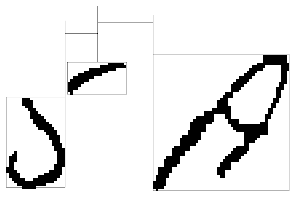


Figure 2. The detected small pieces are grouped to their neighboring good big pieces according to there locations.

First the connected components are generated and sorted according to their horizontal positions. Average height of the components is computed. The baseline of the components is estimated with skew information included. We also compute an average width of characters in the component string using the average height as observation standard, i.e. only the components with horizontal size less than the average height are taken into the computation of the average width.

For each connected component, based on its size, shape and position relative to the baseline, we classify it to be one of the types as *regular big* and *regular small*. The small components far away from the other components and long slim components are treated as candidates of noise which may be removed by a heuristic rule. The punctuation marks and delimiters are also detected and the corresponding connected components are stored into blobs. Average distance of gaps between components is estimated as well.

Then a simple grouping algorithm is designed to group *regular small* components to their closer neighboring *regular big* components. The distances are measured from the center horizontal position of the *regular small* components to their left neighboring *regular big* component's right end and to their right neighboring *regular big* component's left end as shown in Figure 2. The grouped components are built into blobs for characters.

3.3. Estimation of number of characters

For each character blob the number of characters in the blob is estimated with a horizontal intersection method. For the blobs containing single characters, the images are rebuilt from the components in the blobs to be images covered by the components in the original image. And the images each for a single character are sent to the recognition.

The blobs with estimated number of characters greater than one are sent to the segmentation module for further processing.

The estimation algorithm is designed with a very conservative strategy in that the blobs estimated to be of single characters with high accurate rate. In this way, the images sent to the recognition from preprocessing are surely of single characters. However the blobs estimated to be of more than one characters may contain single character. These characters will be further identified in the segmentation module.

4. Segmentation algorithm

Most of the segmentation algorithms in lexicon-driven handwritten word recognition systems are designed for cursive handwritten words. Oversegmentation allows uneven size characters to appear. This is practical in dealing with cursive handwritten words often containing un-recognizable characters. But in applications using format-driven recognition algorithms, each character in its position may play an important role for the keyword. There may no un-recognizable character be allowed. The types of segmentations needed are for splitting touching characters due to imperfect binarization or characters being too close to each other. Connections from one character to another happen but less often than those in handwritten cursive words. Due to th pivotal role of the keywords, the writing quality of the word images in applications using format-driven recognition algorithms is generally hight than that in applications using lexicon-driven recognition algorithms. But for the same reason, the recognition expectations are higher.

Since in the preprocessing we have already grouped the characters in blobs. The segmentation is done on the lower level for each sub-word in the blobs. Our segmentation algorithm uses chaincode for the contour representation of character strokes. Segmentation is done by splitting and grouping the contours.

4.1. Contour classification and segmentation

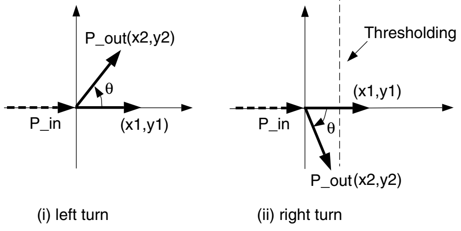
Chaincode of contours is built of arrays of the contour points by tracing the boundary of the character strokes consecutively along counterclockwise direction. The first boundary point encountered in another trace along the norm direction of a contour starting from a boundary point `P` is taken to be the opposite point `OP`. Stroke width is also estimated by averaging all the distances from each boundary points to their opposite boundary points.

A close look at the touching points and ligatures between two characters reveals that at each touching point, the contour makes a significant right turn. We use the significant right turning points to segment the contour to be classes of contour pieces.

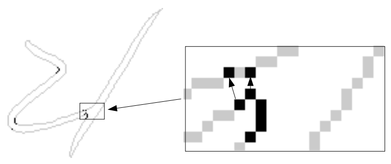
To determine the *significant* right turning contour points, we first compute vectors `P_in` leading in to a contour point `P` from its several previous neighboring contour points and `P_out` going out of `P` to its next several contour points. These vectors are normalized and placed in a Cartesian coordinate system with `P_in` along the x-axis (Figure 3 (a)). A threshold `T` is then selected such that any significant right turn satisfies the conditions:

$$x_1y_2 - x_2y_1 < 0 \quad (1)$$

$$x_1y_1 + x_2y_2 < \mathtt{T} \quad (2)$$



(a)



(b)

Figure 3. (a) The distance between the thresholding line and the y-axis gives a threshold for determining a significant right turn. (b) The significant right turning points and their opposite contour points are marked.

where (1) indicates that the turn is to the right and (2) indicates that the turn is significant. Since the threshold `T` is the x-coordinate of the thresholding line in Figure 3 (a), it can be determined experimentally to be a number close to zero. This ensures that the angle θ made by `P_in` and `P_out` is close to 90°. Significant right turning points together with their opposite contour points located earlier divide the contour into contour pieces(Figure 3 (b)). Contour pieces may now be classified as belonging to one of the two connected characters.

4.2. Grouping contour pieces

Based on informations including sizes and types of characters given in the format, we have multiple ways of grouping contour pieces to make character images in a blob. Final decision is made by a dynamic programming implementation. Contour piece grouping is done as follows. We assume that the number of characters in the blob is known. Then the simplest way of grouping the contour pieces to each group for a character is by using the vertical lines dividing the horizontal extend to regions each is for one character. A contour piece is classified as part of a character if the *center of mass* of the piece is within a particular region for containing the character. In the ideal case, the characters have similar widths and the touching strokes lie close to the middle of the two characters. But this is not always the case. The method described fails, for example, for an image containing touching digits 5 and 1.

For general cases we first estimate a vertical slant using the

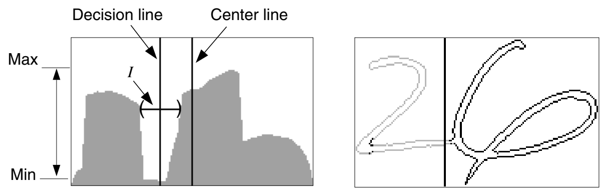


Figure 4. Histogram of vertical extents is used to find decision line, and the segmentation result.

consecutive contour structure. then we compute the histogram of extents along the slant direction where the extent of the image along each slant line is defined as the distance between the uppermost black pixel and the lower most black pixel in that slant line. The touching point may be expected to lie near the valley of the histogram. Valley areas are located to be intervals(Figure 4).

A divide and concur scheme is used to find the N decision lines for $N + 1$ characters in the image. That is, the $N/2$-th decision line is found first and the rest decision lines will be found in the two sub-fields from the image divided by the decision line.

For each decision line we find the intervals closest to the vertical bisecting line (the $N/2$-th vertical line in the histogram) and find the minimal point within the valleys. The one which is closest to the bisecting line is taken as the position of the decision line(Figure 4).

We use the decision line as a guide line to output the contour segments as described earlier. Our segmentation algorithm allows us to classify entire strokes as belong to one or the other neighboring characters, and results in a "cleaner" segmentation compared to traditional segmentation methods.

Images of segmented characters may be recovered from the contour segments by drawing line segments from each boundary point and two neighboring boundary points to their opposite contour points, or the modified opposite boundary points which are built by using the estimated stroke width if any line segments to the opposite points are too long.

In fact, image recovery may be unnecessary since a contour based recognizer can directly use the separated contour segments for recognition.

5. Experiment

The system is implemented as a sub-system in our handwritten interpretation system for UK addresses. A high performance character recognizer GSC developed in CEDAR is used in the system. We tested the system on a set of zipcode images from CEDAR CD-ROM and US Postal Database and a set of post code images from UK addresses. In the first set there are 3074 zipcode images for which a 86% field recognition correct rate and 95% segmentation correct rate are achieved. On the second set of 2556 UK post code images, we got 84.2% field recognition correct rate. The errors are from incorrect grouping, incorrect splitting and incorrect recognition for characters in touching strings. For both tests, zero rejection is assumed.

6. Conclusion

A format-driven handwritten word recognition system is proposed. The major difference of the system from any other systems is its unique segmentation algorithm and its design logic of using multiple levels of views of a handwritten word. Heuristic grouping is done on a high level to get sub-words and segmentation is done on a lower level for identifying the touching or connected characters. Preliminary experiments on real data from US and UK postal zipcode showed very promising results in terms of segmentation and recognition rates when compared with other published systems. Moreover the proposed system works also very well for segmentation and recognition of hand-printed characters with few modifications.

References

[1] J. Park, V. Govindaraju and S. N. Srihari. Efficient Word Segmentation Driven by Unconstrained Handwritten Phrase Recognition, *Proceedings of Fifth International Conference on Document Analysis and Recognition*, (ICDAR 99) Bangalore, India, 1999

[2] S. Seshadri and D. Sivakumar, A technique for segmenting handwritten digits, *Pre-Proc. of the Int. Workshop on Frontiers in Handwriting Recognition III*, Buffalo, New York, USA, May 25-27, 443-448(1993).

[3] R. Fenrich, Segmentation of automatically located handwritten numeric strings, *From Pixels to Features III: Frontiers in Handwriting Recognition*, edited by S. Impedovo and J. C. Simon, Elsevier Science Publishers B. V. 47-59(1992).

[4] G. Kim and V. Govindaraju, Handwritten Word Recognition for Real-Time Applications. *Proc. of the Int. Conference on Document Analysis and Recognition*, Montréal, Canada, 24-27(1995).

[5] Z. Shi and V. Govindaraju, "Segmentation and Recognition of Connected Handwritten Numeral Strings" Journal of Pattern Recognition, Pergamon Press, Vol. 30, No. 9, pp.1501-1504, 1997.

[6] T. M. Ha, D. Niggeler and H. Bunke, A system for segmenting and recognizing totally unconstrained handwritten numeral strings, *Proc. of the Int. Conference on Document Analysis and Recognition*, Montréal, Canada, 1003-1009(1995).

[7] H. Nishida and S. Mori, A model-based split-and-merge method for recognition and segmentation of character strings, *Advances in Structural and Syntactic Pattern Recognition*, H. Bunke(Ed.), World Scientific, 300-309(1992).

[8] Z. Shi and V. Govindaraju, "Character Image Enhancement Using Selective Region Growing", Pattern Recognition Letters, Elsevier Science Publishers, Vol. 17, pp. 523-527, 1996.