Hidden Loop Recovery for Handwriting Recognition

David Doermann
Institute of Advanced Computer Studies, University of Maryland, College Park, USA
E-mail: doermann@cfar.umd.edu
Nathan Intrator
School of Computer Science, Tel-Aviv University, Ramat Aviv 69978, Israel
E-mail: nin@math.tau.ac.il
Ehud Rivlin
Department of Computer Science, Technion, Technion City 32000, Israel
E-mail: ehudr@cs.technion.ac.il
Tal Steinherz
School of Computer Science, Tel-Aviv University, Ramat Aviv 69978, Israel
E-mail: talstz@math.tau.ac.il

Abstract

One significant challenge in the recognition of offline handwriting is in the interpretation of loop structures. Although this information is readily available in online representation, close proximity of strokes often merges their centers making them difficult to identify. In this paper a novel approach to the recovery of hidden loops in offline scanned document images is presented. The proposed algorithm seeks blobs that resemble truncated ellipses. We use a sophisticated form analysis method based on mutual distance measurements between the two sides of a symmetric shape. The experimental results are compared with the ground truth of the online representations of each offline word image. More than 86% percent of the meaningful loops are handled correctly.

1. Introduction

One of the major problems in offline cursive word recognition is hidden loops. Loops are usually very dominant during recognition since they are considered as decision support features with high confidence. In some text, a formatted loop is representative of an entire letter, while in other cases the existence of a loop may distinguish one letter from another. In such cases, when a loop is absent, it is very difficult or even impossible to recognize the character it is associated with and misrecognition might be inevitable. Unfortunately, lost loops that are hidden because of interaction between strokes demand unique handling during the preprocessing phase. In this case loops of this kind become blobs. It is well known that even a single pixel can change the topology of a shape, so interpretation at a regional level is essential.

1.1. Previous Work

Several methods have already been proposed to identify suspicious blobs that represent hidden loops. Some techniques such as Doermann et al. ([1]) and Pettier et al. ([2]) require gray-level images. To the best of our knowledge there has been only one method to recover hidden loops in binary images (Abuhaiba et al. in [3]): removing pixels whose distance from the nearest edge exceeds a threshold. This method is usually impractical because the resulting signal to noise ratio is unacceptable.

In this paper a novel approach for hidden loop recovery based on contour dependent blob analysis is described.

1.2. Theory

Lost loops appear as blobs in a binary image. Blobs, however, can also be found at the intersections of strokes, at junctions, at zones where consecutive strokes were close to each other, or where one stroke partially overrides another. Previous work shows that a blob's width cannot distinguish between those blobs that originated from online loops and other kinds of blobs. Furthermore different stroke widths may be observed on the same word based on the dynamics of the writing process (see [4] for a detailed explanation). In this case shape analysis is required.

In cursive handwriting a loop is formed by a continuous stroke that begins and ends on the same position. The resulting shape resembles an ellipse. Usually the principle component of the loop would be perpendicular to the writing direction. Therefore a major part of the loop's edge is located on the same side of the word's contour, i.e. either on the upper or lower contour.

In an offline image the origin of a loop is often blurred so one cannot identify a single pixel junction. Yet an elliptic contour with an aperture of a stroke width remains. It is our view that a piece of a word's contour that looks like a truncated ellipse is unique and is associated with loops solely. Moreover hidden loops share the same pattern and even though there are no background pixels inside, one can still identify a thin elliptic shape.

This algorithm aims to recover hidden loops that are important for offline word recognition systems. Therefore it does not need to recover redundant loops that do not contribute for the recognition process taken in the next steps. We define redundant loops as those who satisfy one of the following criteria:

- Encapsulated within another loop
- Located on the ascending part of a *c* or *t*
- Located on the short tail of an open *p* or *s*

When possible it is recommended to ignore hidden loops of this kind, given that there exists an automatic process that can identify them a-priori.

Therefore the main principles of our algorithm are:

1. Contour examination for blob hypothesizing and shape analysis.
2. Redundant loop selection based on close neighborhood.

In Section 2 we describe the proposed 3-step algorithm. Then experimental results are reported in Section 3. We summarize in Section 1 with a short discussion and some future work plans.

2. Contour Based Hidden Loop Recovery

In this section a short description of the proposed algorithm is given. There are three phases that are involved in the process:

1. Contour partitioning - locating contour segments that may represent a loop .
2. Selection - removing illegal candidates that a-priori do not resemble an elliptic blob or surround a visible loop.
3. Shape analysis - choosing blobs that satisfy the elliptic shape criterion.

Figure 1 demonstrates the algorithm flow (from top left to bottom right): (a) The original word, (b) the suspicious contour segments, (c) the selected contour segments, and (d) the hidden loop that was discovered.

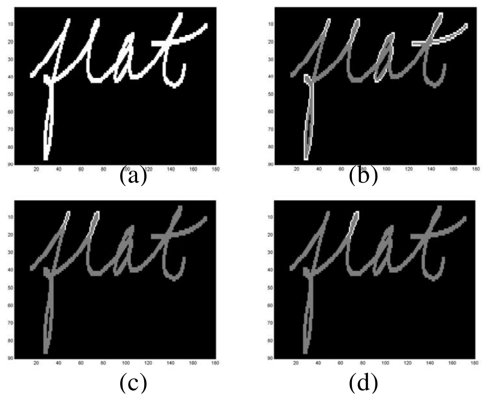

Figure 1 - A demonstrative algorithm flow:

(a) The original word; (b) The suspicious contour segments; (c) The selected contour segments; (d) The discovered hidden loop;

In the following sections, we expand on each step.

2.1. Contour Partitioning

Let the external contour be the surrounding path of the word when tracing the edge on the inside and clockwise. Without loss of generality one can assume that the contour starts (and ends) on the original starting point chosen by the writer. In this case the upper contour would be the prefix of the contour until the original ending point and the lower contour would be the suffix of the contour from this point and on.

Let D_1 be the function of the distances between every pixel on the upper contour and its nearest neighbor on the lower contour, and let D_2 be the function of the distances between every pixel on the lower contour and its nearest neighbor on the upper contour respectively. Given a pixel on one contour, one can use a BFS (Breadth First Search) algorithm to find a pixel on the other contour that minimizes the Geodesic distance between the first pixel and all pixels on the other contour. We define the second pixel as the nearest neighbor of the first one. In this case the image is considered as a graph where each pixel is a node and 8-neighboring pixels are connected with edges.

Next both the upper and lower contours are divided into regular and singular sections. Let *p* be a pixel on the upper contour, then:

$$p \in Regular \text{ iff } D_1(P) \leq threshold$$

$$p \in Singular \text{ otherwise}$$

Where *threshold* is the averaged stroke width. It is our view that the most frequent value of both D_1 and D_2 is a fine estimation of the averaged stroke width.

The lower contour is partitioned similarly using D_2.

One can observe that *Regular* sections belong to axial strokes while *Singular* sections are associated with either one of tarsi or axial loops.

2.2. Selection

At this stage singular sections that are associated with redundant loops are filtered out. The criterion that was used to disqualify loops of this kind was the number of extremity points found in the interval. Only singular sections of the upper (lower) contour that contain a single local maximum (minimum) point are selected.

Additional singular sections are filtered out if they appear on the surrounding of a visible loop.

2.3. Shape Analysis

Given a set of the selected singular contour sections, each section is tested for being elliptic. The most significant property of the ellipse is the narrow waists that are accepted in the origin of the blob. However it is possible that there will be only 1 pixel difference between the maximum and minimum widths of the blob. In this case no smoothing is allowed because it may wipe out the most important pixel. Therefore one cannot utilize turning angle analysis methods, because angles are not well defined on curves that were not polygonized first.

Nevertheless the width of a blob at each point can be computed, given that the blob is divided into 2 sides. In this case one can use the above mentioned nearest neighbor scheme for this purpose. Moreover since there is only one local maximum (minimum) point, it is reasonable to make it the symmetry center for further analysis.

Therefore in the first step each section is divided into two subsections denoting the left and right substrokes of the presumed loop respectively. Then each substroke is traced until the first local minimum (maximum) is reached.

Next, a mutual distance computation between pixels on each side and their nearest neighbors on the other side takes place. A BFS algorithm is used for this purpose. Finally the level of convexity of each subsection is evaluated exclusively according to the following criteria:

Let S_1, S_2 be a pair of matching subsections, and let D_1 be the function of mutual distances between pixels of S_1 and the closest neighbor of each one of them on S_2. Let MN and MX be the set of local minimum and local maximum points on D_1 respectively. S_1 Would appear convex iff

$$\begin{aligned} &\exists mn \in MN \wedge mx \in MX \mid \\ &mn > mx \wedge D_1(mn) < D_1(mx) \wedge \\ &(mn - mx > 2 \vee D_1(mx) - D_1(mn) > 1) \end{aligned}$$

Meaning that there will be local minimum that is more than 1 pixel deep or more than 2 pixels wide. A section is labeled elliptic iff at least one of its subsections is convex.

Figure 2 illustrates the steps in shape evaluation (from top left to bottom right): (a) A selected singular contour section (note the missing white pixel on top at the local maximum point), (b) the trailed left substroke (S_1), (c) the trailed right substroke(S_2), (d) the mutual distance functions (D_1 and D_2) above and below respectively.

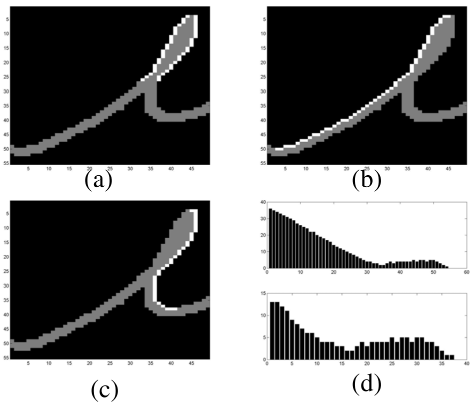

Figure 2 - Steps in shape evaluation:

(a) A selected singular contour section (the missing white pixel on top is the local maximum point); (b) The trailed left substroke (S_1); (c) The trailed right substroke(S_2); (d) The mutual distance functions (D_1 and D_2) above and below respectively

3. Experimental Results

The target of our algorithm is to recover as many hidden loops as possible under the constraint of minimum signal to noise ratio. In this case noise refer to misdetection, i.e. blobs that are mistakenly identified as hidden loops though they stand for normal thick strokes. At this stage it is beyond the scope of our work to test the influence of the proposed algorithm on the following word recognition systems.

Our method was tested on a database of more than 1270 words. There were 5 different writers who wrote about 220 words each, and another writer who wrote 170 words. These writers and words were selected for using pure cursive handwriting.

See Table 1 for database statistics.

For each word the database provides both the online and offline samplings. Since both recordings were made simultaneously, one can refer to the online representation as ground truth. In this case we define a hidden loop as a loop that is identifiable in the online representation of a word but is not traceable in the offline image. Several examples are given in Figure 3.

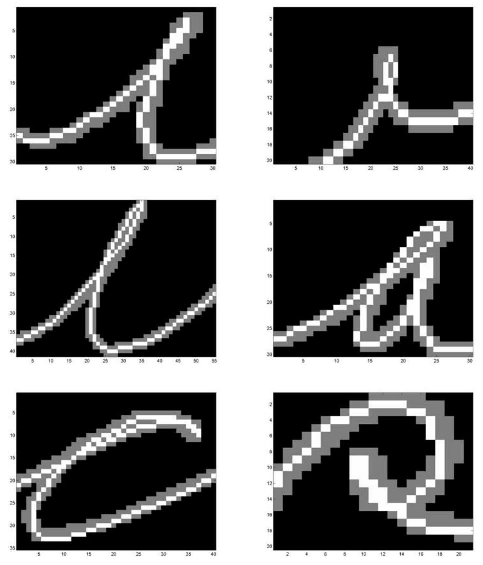

Figure 3 – Examples of hidden loops

One should be aware of the fact that it is very difficult to define a meaningful online loop. For this purpose we define a meaningful online loop as an online loop is either *real* or *large* enough, where:

- *Real* loops are loops whose area is larger than their perimeter, and
- *Large* loops are loops that are not real but their perimeter is more than a *threshold* long.

In Figure 4 there are examples of the two types of meaningful online loops. The gray pixels denote the offline image while the white pixels denote the online trajectory. The left loop is *real* and the right loop is *large* (for a thresholds of 6 pixels and higher). One can see that both loops are hidden in the offline image.

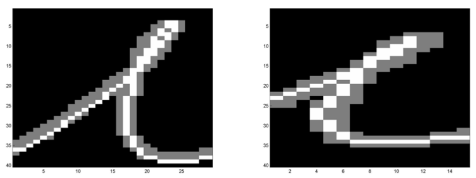

Figure 4 - Examples of *real* (left) and *large* (right) loops respectively

Table2 summarizes the recovery results of *real* loops. Recovery rate is calculated with respect to the number of online loops.

Table 3 and Table 4 summarize the recovery results of *large* loops with thresholds of 8 and 6 pixels respectively.

One can see that more than 95% of the *real* loops and 77%-84% of the *large* loops are handled correctly. In addition it is proven that hidden loops are very common in cursive words – more than 1.5 meaningful hidden loops per word on average.

Misdetections were also counted. In this case a misdetection is either one of a *small* loop or a meaningless blob that were mistakenly recovered. S*mall* loops are online loops that are neither *real* nor *large*.

Figure 5 illustrates an erroneous classification of a blob that was labeled as a hidden loop although it is not associated with an online loop of any kind.

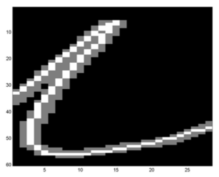

Figure 5 - A misdetection - this blob was mistakenly identified as a hidden loop

Table 5 summarizes the number of misdetections for the different thresholds.

4. Discussion

This article describes a novel approach to recover hidden loops. The results that were presented show that this is a reliable method for the recovery of most of the meaningful online loops that were blotted and covered.

The algorithm utilizes a shape analysis scheme that can overcome the problems that arise when smoothing is used. This scheme is applicable to other domains as well. The embedded loop classification process that filters out irrelevant loop also seems very promising. The module was tested randomly and proved to be very accurate. Thus it may be used to support character recognition as well.

We found it very difficult to determine what is the effective ground truth of such a problem. We believe that our choice of criteria and thresholds is good, given that algorithms of this kind are preprocessing in a word recognition system. In this case it is even complicate for a human being to classify the practical events.

Our experiments have proven that meaningful hidden loops are very common (more than 1.5 per word on average). The proposed algorithm handled 95% of the *real* loops and 77%-84% of the *large* loops correctly.

Future work will concentrate on several improvements of the algorithm. There are several parameters in the shape analysis module that require fine tuning. Additional criteria for classification may also contribute to improve the overall recovery results.

5. References

[1] D. S. Doerman and A. Rosenfeld. The interpretation and reconstruction of inferring strokes. In *Proceedings International Workshop on Frontiers in Handwriting Recognition,* pages 41-50, 1993.

[2] J. C. Pettier and J. Camillerapp. Script representation by a generalized skeleton. In *Proceedings International Conference on Document Analysis and Recognition*, pages 850-853, 1993.

[3] I. S. I. Abuhaiba and M. J. J. Holt and S. Datta. Processing of binary images of handwritten text documents. *Pattern Recognition*, 29(7): 1161-1177, 1996.

[4] D. Doermann. Document image understanding: integrating recovery and interpretation. PhD thesis. University of Maryland, 1993.

Table 1

	Writer #1	Writer #2	Writer #3	Writer #4	Writer #5	Writer #6	Total
Number of words	223	219	223	170	215	223	1273
Number of characters	1130	1113	1130	835	1083	1130	6421
Number of Loops (all kinds)	1039	1272	1013	745	1332	1146	6547

Table 2

Real Loops	**Online Loops**	**Offline Loops**			
		Encapsulated	Disqualified	Found	Total
Number	1006	259	186	519	964
Rate	100%	25.7%	18.5%	51.6%	95.8%

Table 3

Large Loops (>8)	**Online Loops**	**Offline Loops**			
		Encapsulated	Disqualified	Found	Total
Number	856	233	147	341	721
Rate	100%	27.2%	17.2%	39.8%	84.2%

Table 4

Large Loops (>6)	**Online Loops**	**Offline Loops**			
		Encapsulated	Disqualified	Found	Total
Number	1105	288	177	390	855
Rate	100%	26.1%	16.0%	35.3%	77.4%

Table 5

Threshold	*Small* Loops	No Loops	Total
8	180	209	389
6	131	209	340