Line Removal and Restoration of Handwritten Strokes

Arvind K R, Jayant Kumar and A G Ramakrishnan
MILE Lab, Department of Electrical Engineering
Indian Institute of Science, Bangalore-560012, INDIA

Abstract

In document images, we often find printed lines over-lapping with hand written elements especially in case of signatures. Typical examples of such images are bank cheques and payment slips. Although the detection and removal of the horizontal lines has been addressed, the restoration of the handwritten area after removal of lines, persists to be a problem of interest. In this paper, we propose a method for line removal and restoration of the erased areas of the handwritten elements. Subjective evaluation of the results have been conducted to analyze the effectiveness of the proposed method. The results are promising with an accuracy of 86.33%. The entire process takes less than half a second for completion on a 2.4 GHz 512 MB RAM Pentium IV PC for a document image.

1 Introduction

A typical document analysis system consists of a preprocessing stage that enhances the quality of the document image by removing noise elements such as spurious dots and lines, followed by machine printed and non-printed element extraction and finally the actual printed/handwritten recognition level. Although the preprocessing stage of removal of independent dots and lines seems trivial, in cases where the lines are superimposed over handwritten areas, it becomes a complicated problem. The accuracy of any handwritten recognition system depends on the quality of the handwritten elements fed into system. Hence, it becomes necessary for the lines passing over them to be removed and the erased regions to be restored appropriately.

Yu and Jain [1] have proposed a method for line removal and character restoration using Block Adjacency Graph representation of the input binary image. The horizontal form lines were located by finding long straight lines based on the block adjacency graph. Form line separation and character reconstruction were also implemented from this graph. Jin-Yong Yoo *et. al.* [2] have classified the various types of junction points at the point of contact or crossing over of the characters and the line. After line removal, the junction points are detected and restored based on their classification type. A. L. Koerich and Lee Luan King [3] have detected and removed the lines using horizontal projection profile(HPP). The removed regions are rectified by checking the neighbors for every pixel that could be fitted into the erased line. Based on whether the neighboring pixels satisfy certain condition or not, they decide to leave the pixel on or off.

Xiangyun Ye *et. al.* [4] have used morphological methods for removal of lines and restoration of erased area. Govindaraju and Srihari [5] have proposed a method for removing interfering strokes under the assumption that the thinning algorithm was applied and the curvatures of interfering strokes was very smooth. Global properties of textual word shapes and those of interfering strokes were used to separate them.

In this paper we propose a fast, simple and effective method for line removal and restoration of the handwritten characters. We have been able to achieve an accuracy of almost 86.33%. The timing analysis has proven our method to be very efficient taking less than a second for an image of resolution 2000x2000 on a 2.4 GHz 512 MB RAM Pentium IV PC.

2 System Description

The system is divided into four steps. Initially, the document image is cleaned of all noise elements such as spurious dots and lines. Next, it is segmented into its constituent blocks. This is carried out similar to the block segmentation method given by Arvind *et. al.* [6]. Then the blocks are skew corrected based on their horizontal projection profiles and entropy. In the third level, the rows of the blocks containing the printed lines are detected using the projection profiles. Then we identify the run-lengths within these rows that actually make up the line and remove them. Finally, the regions that belong to the handwritten elements that were removed along with the line are recovered.

2.1 Noise removal

We apply Connected Component Analysis (CCA) and obtain the number of ON pixels and aspect ratios for every component. Then, we find the minimum and maximum values of them. Let them be *minp*, *maxp* and *mina*, *maxa* respectively.

$$TP = \frac{np - minp}{maxp - minp} \tag{1}$$

$$TA = \frac{na - mina}{maxa - mina} \tag{2}$$

where *np* is number of ON pixels in the component and *na* is aspect ratio of the component. If TP or TA is less than 0.002 which is computed emprically, then we remove it.

2.2 Block Segmentation

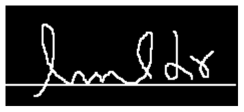
Arvind *et. al.* have segmented blocks by run-length smoothening the image with the parameter selected such that the intra and inter character gaps, up to a paragraph level, are filled. Then, they apply morphological erosion operator to remove thin joints between blocks. Finally CCA is used to segregate the blocks.

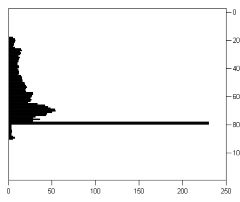
2.3 Skew Detection and Correction

Assuming that the maximum skew would not be greater than 10 degrees, we rotate the image by ±10 degrees and obtain the HPPs along with their corresponding entropy values. The resolution of rotation is one degree. The angle for which the entropy is minimum, is the skew angle of the segmented block. Entropy is given by Eqn. 3.

$$E(i) = \sum_{i} -HPP(i) * log(HPP(i)) \tag{3}$$

where E is Entropy. After obtaining a rough estimate of the skew angle, we go in for a finer estimate of the skew angle around the rough estimate, upto a resolution of 0.1 degree. After obtaining the skew angle, the image is rotated by that angle in the opposite direction using bilinear interpolation. Thus the block is skew corrected. Although the skew detection and correction could be done on the entire document image, it is done at block level since every block with its logos, typing defects etc. could have a different skew.





(a) (b)

Figure 1. (a) Image with printed line (b) Horizontal projection profile of image. The peak appears at the row which contains the printed line

2.4 Line detection and removal

The line detection is fairly simple. We observe that at places where there exists a line, there is a peak in the HPP. Hence, those rows of the image, where there is a peak in the HPP, is marked for a potential machine printed line. We say that there is a peak if the HPP value is greater than k times the mean of the HPP value. The value of k has been selected empirically. An example for line detection is shown in Fig. 1. After potential line containing rows have been detected, we traverse through the rows and obtain the run-lengths within them. If the run-length is greater than some p percent of the width of the image, then we consider it to be a part of the machine printed line. Then, we extract the entire component to which the run-length belongs by applying CCA. Finally, the rows containing the line within this component are detected in the same manner as mentioned previously. All the run-lengths within the rows satisfying the threshold are removed, thus removing the machine printed line.

2.5 Restoration of handwritten elements

The restoration process involves two stages: *Detection of stroke* and *Filling up of erased* area.

2.5.1 Detection of diagonal stroke

After line removal, we calculate the two values x and y as depicted in Fig. 2(a), given by the following equations.

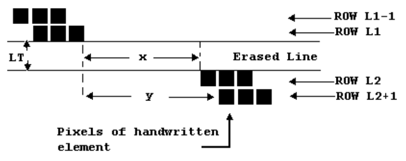
$$x = round\{\frac{3+LT}{0.17}\} \tag{4}$$

$$y = floor\{\frac{(3+LT)x}{2+LT}\} \tag{5}$$

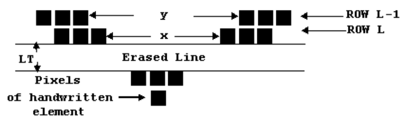
where LT is the thickness of the handwritten stroke. In Eqn. 4, 0.17 indicates the minimum angle of the handwritten stroke with respect to the printed line, that we are checking for, i.e. $Sin^{-1}(0.17) \simeq 10$ degrees is taken to be the minimum angle of the handwritten stroke. If the stroke angle is lesser than this, then it is difficult to distinguish between the printed line and the handwritten stroke. x is the distance of breakage caused by the machine printed line over the handwritten stroke. In Eqn. 5, y is the probable distance at which ON pixels must be found for a given value of x. As shown in Fig. 2(a), $L1$ is the row above the line and $L2$ is the row below the line. The algorithm is explained below.

- Calculate x
- Start checking for ON pixel in row $L2$, from the pixel below the right most ON pixel on row $L1$, for x pixels. If an ON pixel is found within a distance of x pixels, then update the value of x.
- Calculate y
- Start checking for On pixel in row $L2 + 1$, from the pixel below the right most ON pixel on row $L1$, for y pixels. If no ON pixel is found within a distance of y pixels and there exists an ON pixel at the y^{th} pixel, then the criteria for diagonal stroke is fulfilled; else do nothing
- If criteria is satisfied then start diagonal filling algorithm

This algorithm is executed on both directions to detect both left and right diagonal strokes.



(a)



(b)

Figure 2. (a) Figure depicting x, y and LT for diagonal stroke (b) Figure depicting x, y and LT for U stroke

2.5.2 Detection of U stroke

Fig. 2(b) depicts the important parameters such as x, y and LT in the detection and restoration of U stroke. y is given by Eqn. 6. The algorithm is explained below.

$$y = floor\{\frac{(2 * LT + 2)x}{2 * LT + 1}\} \tag{6}$$

- Obtain x which is the distance between the strokes of the handwritten element in row L as shown in Fig. 2(b).
- Calculate y as given by Eqn. 6.
- Obtain the distance k between the handwritten strokes on row $L - 1$
- If k = (y±1) then, it satisfies the criteria for U stroke and start the U filling algorithm

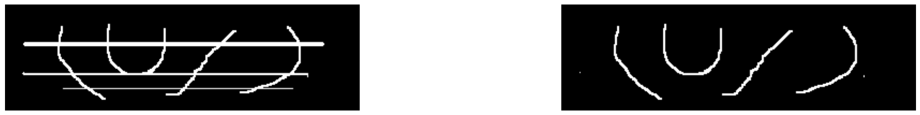


Figure 3. Synthetic image with multiple printed lines having varying thickness and image after line removal and restoration

2.5.3 Filling of diagonal stroke

After detection of diagonal stroke, we simply start joining the handwritten element above and below the line using Bresenham line drawing algorithm [7]. Bresenham's line algorithm is an algorithm that determines which points in an n-dimensional raster should be plotted in order to form a close approximation to a straight line between two given points. It is one of the earliest algorithms developed in the field of computer graphics.

2.5.4 Filling of U stroke

We first determine the minimum between the thickness of the stroke t and the thickness of the machine printed line LT. Let us assume that the minimum value happens to be t. Then, we draw horizontal lines for t number of rows. Although we could draw horizontal lines exactly joining the stems of a U stroke, this would look artificial. Hence, we use a step like formation. The step formation is given by Eqn 7.

$$Newlength = L - (R * K) \tag{7}$$

where L is the actual length between the strokes, R is the row number of the row being currently being filled, K is minimum of t and LT. Fig. 3 shows an image consisting of multiple lines with varying thickness and its restored image.

3 Experimental Results

3.1 Data Description

Our data consists of 300 English document images, scanned at 200 dpi and stored in 1-bit depth monochrome format. These documents contain handwritten elements, signatures, logos and other such things along with free-flowing text paragraphs.

3.2 Timing Analysis

The algorithm has been implemented in ANSI-C language and compiled using GCC compiler. It has been executed on a Pentium IV 2.4 GHz, 512 MB RAM PC. Table I depicts the timing for various stages in the restoration process for 100 block images that were obtained by block segmentation. This timing does not include the skew correction and file writing.

Table 1. The Detection and Correction time for 100 block images

Total Detection Time	10.57s
Total Correction Time	33.05s
Total Time	43.62s
Average Time	0.44s

3.3 Time Complexity

If n is the number of lines removed and l is the length of the longest line, the total number of times this calculation is done will grow proportional to $O(n) * O(l)$. Also, if m is the number of different classes of strokes, which is constant, time-complexity for identification of stroke comes as $O(n) * O(l) * O(1)$. After that, the restoration is achieved by obtaining pair of points from the two broken ends and drawing a Bresenham-line between them. If Tl and Ts are the thickness of line and stroke, the time complexity for line drawing can be approximated to $O[2 * min(Tl, Ts)]$.

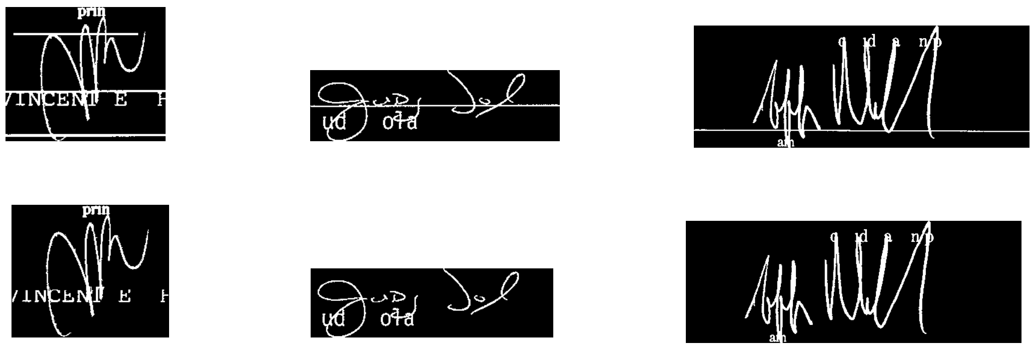


Figure 4. Images with printed lines are shown in first row and corresponding images after line removal and restoration is shown in second row

3.4 Accuracy Calculation

Since there is no absolute accuracy for restoration process, we had to go for a subjective evaluation. We ran the algorithm on 300 block images and distributed the results among three people. We asked them to rate the results as good, moderate and bad with points of 1, 0.5, 0 respectively. Table 2 displays the results. We have obtained an average accuracy of 86.33%. Fig. 4 shows some of the results that we have obtained.

4 Conclusion

In this paper, we have presented a novel method for restoration of handwritten characters after removal of machine printed lines passing over them. The method is extremely fast, simple and can handle restoration of handwritten elements with multiple lines passing over

Table 2. The Accuracies for 100 block images per person

Person	Accuracy
A	88%
B	85%
C	86%
Average	86.33%

them with varying thickness. Additionally, we divide the document image into blocks and skew correct them individually. Hence, we can handle documents with blocks having multiple skew. As future work, we are intending to extend the algorithm to restoration of printed characters as well.

References

[1] B. Yu, A. K. Jain, "A generic system for form dropout", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(11):1127-1132, 1996

[2] J. Yoo, M. Kim, S. Y. Han, Y. B. Kwon, "Line removal and restoration of handwritten characters on the form documents", *Proc. 4th Int. Conf. on Document Analysis and Recognition*, pp. 128-131, 1997

[3] A. L. Koerich L. L. Ling , "A system for automatic extraction of the user-entered data from bank checks", *Proc. of Int. Symposium on Computer Graphics, Image processing and Vision*, pp. 270-278, 1998.

[4] X. Ye, M. Cheriet, and C. Y. Suen, "A generic method of cleaning and enhancing handwritten data from business forms", *Int. J. on Document Analysis and Recognition*, 4:84-96, 2001.

[5] V. Govindaraju and S. N. Srihari, "Separating Handwritten Text from Interfering Strokes", *From Pixels to Features III: Frontiers in Handwriting Recognition, Elsevier Science Publisher*, pp. 1728, 1992.

[6] Arvind K. R., Peeta Basa Pati, A. G. Ramakrishnan, "Automatic text block seperation in document images", *Proceedings of 4th International Conference on Intelligent Sensing and Information Processing*, 2006.

[7] J. D. Foley, A. V. Dam, S. K. Feiner, J. F. Hughes, *"Computer Graphics: Principles and Practice in C ", 2nd Edition, Addison-Wesley, Pearson Education*