

Text Extraction from Gray Scale Document Images Using Edge Information

Q. Yuan, C. L. Tan

Dept. of Computer Science, School of computing

National University of Singapore

3 Science Drive 2, Singapore 117543

Email: {yuanqing, tancl}@comp.nus.edu.sg

Abstract

In this paper we present a well designed method that makes use of edge information to extract textual blocks from gray scale document images. It aims at detecting textual regions on heavy noise infected newspaper images and separate them from graphical regions. The algorithm traces the feature points in different entities and then groups those edge points of textual regions. From using the technology of line approximation and layout categorization, it can successfully retrieve directional placed text blocks. Finally feature based connected component merging was introduced to gather homogeneous textual regions together within the scope of its bounding rectangles. We can obtain correct page decomposition with efficient computation and reduced memory size by handling line segments instead of small pixels. The proposed method has been tested on a large group of newspaper images with multiple page layouts, promising results approved the effectiveness of our method.

1. Introduction

The stated work is part of the project that facilitates the news article retrieval from the microfilm archives of the Singapore National Library. As the requirement of automatic processing on large sum of scanned newspaper images in microfilm format, we demand an automated image processing system to handle the case. Actually the processing of document image segmentation and classification is an OCR pre-processor, as well as pre-processing for understanding document page layout with structured formats. Through this stage page segmentation will need to accurately partition images into blocks of text, figure, table, frame, and other entities.

Despite the many efforts spent on the subject (see [2] for thorough review) there is still much room for improvement in document segmentation techniques, which is the key factor to improve the overall performance of an automatic reading/processing system. Even very good OCR system can be almost useless when text-extraction is

performed poorly, which is often the case in existing systems for documents with multiple layouts.

Techniques for document segmentation and layout analysis are traditionally subdivided into three main categories: bottom-up, top-down and hybrid techniques. Some other up-to-date methods are introduced by recent progresses in this area, so as to expand the scope of above categorization [15].

Bottom-up techniques [7,8,13] progressively merge evidence at increasing scales to form, e.g., words from characters, lines from words, columns from text lines. They are usually more flexible than top-down methods, but they may suffer from the accumulation of mistakes when going from the small-scale details up to the large-scale features.

Top-down techniques [5,6,9] start by detecting the large-scale features of the image (e.g., columns) and proceed by successive splitting until they reach the smallest-scale features (i.e., individual characters, or text lines). For the procedure to be effective, a priori knowledge about the structure of the page is necessary. These techniques are therefore particularly useful when the layout is constrained, such as is often the case when considering pages from scientific journals.

Most methods do not fit into one of these two categories and are therefore called hybrid. Among these we can find methods based on texture analysis [4] and methods based on background analysis [3]. In methods based on texture analysis the problem of reconstructing the document layout is seen as a problem of texture segmentation. The document page is subdivided into small regions each of which is classified as belonging to one of a few categories (text, drawing, image, etc) according to an analysis of its texture. Once each region in the image has been tentatively classified, a globally consistent segmentation is carried out by the usual techniques of machine vision. Examples of methods using texture analysis are those based on Gabor filtering and mask convolution, fractal signature, and wavelet analysis. All these methods are quite general and flexible but they are also computationally demanding.

After a brief review of previous approaches to document segmentation, the following part of this paper describes a somewhat novel algorithm based on edge detection and line approximation analysis to extract text from document pages. This method is comparatively flexible and fast, and it deals successfully with documents with a relatively complex layout, documents where graphical features and text are intertwined, skewed at some degree and infected with heavy noise in scanning process. In this paper we assume that the character strings on the image are mainly aligned horizontally or vertically, but theoretically it may also be modified and then adapted to other directions as well.

The organisation of this paper is as follows: Section 2 is devoted to outline of our method. In section 3 we present a detailed characterisation of our algorithm. The illustrated experimental examples and results are shown in Section 4. Finally we give some discussion and conclusion in Section 5. Time performance and time complexity are reported.

2. An overview of the method

Our system takes advantage of the distinctive characteristics of text that make it stand out from other image material. For example, by looking at the comic page of a newspaper a few feet away, one can probably tell quickly where the text is without actually recognizing individual characters. Intuitively, text has the following distinguishing characteristics:

- 1) Text possesses certain frequency and orientation information;
- 2) Text shows spatial cohesion—characters of the same text string (a word, or words in the same line) are of similar heights, orientation, and spacing.

Therefore, the most intuitive characteristics of text are its regularity. Printed text consists of characters with approximately the same size and line thickness that are located at a regular distance from each other. Such regularities can also be observed from edges being detected on textual boundaries, we may easily find the elongation tendency when take run-length smearing operation on textual areas. The regular alignment of characters into lines and columns let it be classified from other functional regions distinctly. As a result, our approach tried to find critical characteristics of text and then make use of its edge features to segment the image into text/non-text regions as best as possible, accordingly higher accuracy can be achieved when OCR system do recognition focusing on identified regions. In short, our method includes several major steps such as Preprocessing for noise reduction; Edge detection to form

directional edge plane; Edge merging and bounding line approximation; Region classification by spatial grouping. The detailed description is stated in next section, we omit the pre-processing stage that just use standard filters to sharpen the intensity of source image with noise reduction.

3. Description of algorithm steps

One most important step of the algorithm is to find approximate locations of text lines in a gray-scale image. The main idea of this algorithm can be explained by considering a printed page with Manhattan layout. If we compute the spatial variance along each horizontal line over the whole image, we see that regions with high variance correspond to text lines, and regions with low variance correspond to background or other textures. Moreover, through run-length smearing operation we can find that edges from textual regions may elongate to form the approximate encircling rectangles. Text lines can then be found by extracting the rows between two parallel edges of the spatial variance – in specific distance of average character height. This heuristic can be applied to a more complex image, assuming that the spatial variance in the background is lower than in the text. Since we need to locate both the row coordinates of a text line and the column coordinates of its beginning and end, the spatial variance must be computed for each pixel over a local neighborhood in the horizontal direction. This results in an image of horizontal spatial variances with the same size as the input image. From this image we need to find significant horizontal edges and then pair the edges with opposite directions into lower and upper boundaries of a text line.

3.1. Directional edge plane acquisition

We choose Canny edge detector in our system to detect edges from grayscale images. Canny operator has several advantages: It has low probability of missing an edge, at the same time it has some resistance to the affection by the presence of noise.

Since we are currently interested only in horizontal text lines, we select those edges that have horizontal direction.

Following notations summarized the algorithm. Let $I[i, j]$ denotes the image. The result from convolving the image with a Gaussian smoothing filter using separable filtering is an array of smoothed data,

$$S[i, j] = G[i, j; \sigma] \times I[i, j] \quad (1)$$

$$M[i, j] = \sqrt{P[i, j]^2 + Q[i, j]^2} \quad (2)$$

$$O[i, j] = \arctan(Q[i, j], P[i, j]) \quad (3)$$

Where σ is the spread of the Gaussian and controls the degree of smoothing, P and Q are approximated partial derivatives on x and y direction.

The algorithm passes a 3×3 neighborhood across the magnitude array $M[i, j]$. The values for the height of the ridge are retained in the nonmaxima-suppressed magnitude. We use $O[i, j]$ to control the filtration of major directional edge pixels.

Our updated steps of directional edge detection algorithm is outlined as follows:

1. Smooth the image with a Gaussian filter.
2. Compute the gradient magnitude and orientation using finite difference approximations for the partial derivatives.
3. Apply non-maxima suppression to the gradient magnitude.
4. Use the double thresholding algorithm to detect and link edges.

3.2. Edge merging by horizontal smoothing

Before finding pairs of edges with opposite directions, small horizontal edge components need to be merged into longer lines. The edge merging is performed by first finding the connected components of the short edge segments. Components that have a large vertical spread are not good candidates for upper and lower boundaries of a text line, thus can be removed. Among the remaining segments, we find the ones that have similar row coordinates within a threshold, and merge them into a single line. The merged edges, of course, must have the same orientations. In case some skew angles may exist at those scanned microfilm images, we can also expand the scanning angles to be around ± 5 degrees. The method in our implementation try to find length of projection on axis for linked edge lines, retain those positions which approximate the centroid of text lines. As a result we can cluster the linked edges to form rows of text lines. The final step is to group together pairs of lines with same function as top and bottom line. The huge number of all possible groupings is reduced by using the following heuristics which are a bit similar to those provided by Jain [1]:

1. There must be no other lines between the two compelling edge lines in a pair.
2. A significant portion of the two lines must overlap when they are projected vertically.
3. The distance between the upper and the lower line should not be very small or very large. It can be adapted with guidance of character size.

3.3. Use Thin Line Coding (TLC) to generate line segments at less memory cost

As always, there would exist some spurious segments from operation of last step. We need to filter out these noise-like false segments. In our case, noisy lines might be isolated lines, spurs, and loops that are shorter than given thresholds.

We describe a versatile approach based on a hierarchical representation called as Thin Line Code (TLC) provided by O’Gorman. TLC operates on chains and line features instead of pixels. When this encoding scheme is used, noise reduction can benefit from the use of contextual information to aid the analysis and can so remove features whose size or type would make them difficult to identify by a matched filter.

The TLC features marked for elimination are short isolated lines and spurs. The contour image is coded in PCC and subjected to filtering on the basis of structure size. In addition to the advantage of feature-based and contextual filtering, TLC also provides a computational advantage over pixel-based processing because here are many fewer TLC structures and features than there are pixels. Take the example of image in *Figure 1*, the document image contains 2550×3000 , or 8.4 million pixels. However, after TLC coding, the resultant representation typically requires only $\approx 0.01\%$ of the memory that is required to store the original image. Searching for a line type or feature in this representation is of course much faster than doing so on the original image.

3.4. Strait-Line fitting to achieve bounding rectangles

We can approximate the chain of edge segments with a straight line running through most of the edge points, finding the optimal straight-line fit to a given group of edge points is our final goal. The most direct strategy to fit a straight line to a set of points is simply to connect end points. However, more sophisticated methods are required to obtain the best fit using all points in the given chain, but not only those end points. The least-squares method is widely used in statistical analysis when one variable, y , is dependent on the other, x . Above mentioned method that is generally more appropriate to most image analysis problems, which involve x and y not as statistical variables but as locations of points in an image. From experiment we found latter method is more suitable for our system.

At this moment we need to approximate line segments that are compelling on horizontal and vertical directions, as a result to filter out promising candidates by characteristic shapes. In the last stage we group parallel straight-line segments to find its bounding rectangles.

Paragraphs could be categorized for later page layout analysis and content recognition.

4. Illustrative examples and results

In this section we illustrate the experimental results by using our segmentation method. The test base includes a large group of images at resolution ranging from 1888x2520 to 2552x3000 pixels. All test results come out at average running time of 3.83 ± 0.45 s per image on a desktop PC with a Pentium-II CPU (400 MHZ), excluding I/O operations. We chose Microsoft Visual C++ 5.0 as the development platform and compiler. Optimisation was introduced in programming stage for efficient memory allocation and suitable data structures. *Figure 1 and 2*

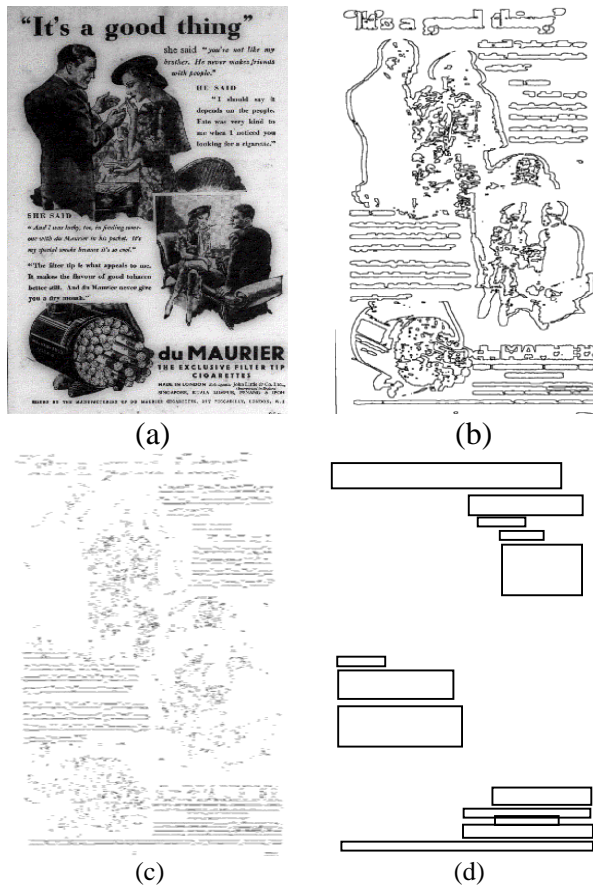


Figure 1: Experimental results. (a) Portion of original gray level image which has mingled graphics and text blocks; (b) Achieved image after edge detection and edge merging in horizontal direction; (c) Pairs of line segments being retrieved by line fitting process; (d) Output image from connected component analysis for finding the bounding rectangles.

present two typical images with the bounding rectangles around extracted textual blocks according to the results of page segmentation. *Table 1* shows the running time for major steps when handling these two typical images.

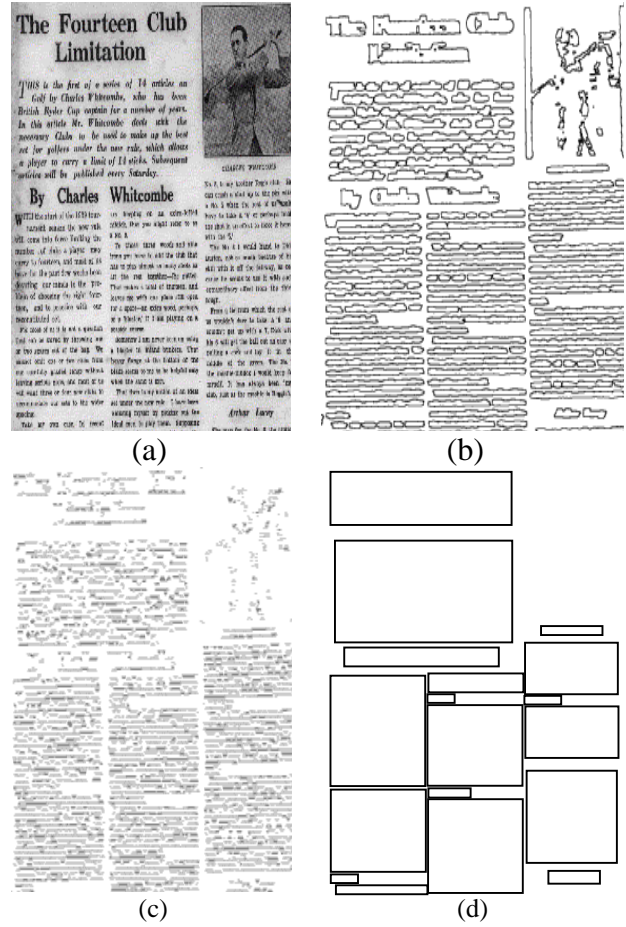


Figure 2: Another experimental results. (a) Portion of original gray level image which has standard newspaper layout; (b) Edges being detected and merged from horizontal dominated direction; (c) Pair of bounding line segments achieved by line approximation; (d) Output image from connected component analysis to locate textual area in their bounding boxes.

5. Discussion and Conclusion

The main benefits of our method falls on its efficiency for less memory usage and fast running speed, as we can see from the statistical data from last section. The algorithm is also flexible with local adaptive thresholds, which made it somewhat robust to skew being introduced at less than 5 degrees. At the same time the

presence of skew does not require extra pre-processing steps.

One of major weakness of our algorithm reside on its assumption that all text is oriented in the same direction, which is by default horizontal. This makes the algorithm not suitable to deal with documents with multiple layout styles, which means modification is still needed to cope with more sophisticated cases. From experiments we found the ratio of successful detection dropped down sharply under cases when textual paragraphs intertwined heavily with irregular graphical blocks. Other than that, there also exist highly demands to increase the accuracy ratio on handling textual block extraction with complex background. Compared with other methods, our algorithm relied more on adaptability of predefined criteria, it is also one of our future aims to enhance self-learning ability for more robust detection mechanism.

Table 1: Running time of major steps

	Sample Image 1	Sample Image 2
Resolution (pixels)	1936 × 2864	1944 × 2872
Overall Processing (s)	3.96	3.72
Edge Detection & Linking (s)	0.29	0.27
Horizontal Smoothing (s)	0.81	0.74
Straight Line Approximation (s)	1.47	1.53
Bounding Rectangle Finding (s)	1.39	1.18

References

1. Anil K.Jain and BinYu, "Automatic Text Location in Images and Video Frames", *Pattern Recognition, Pattern Recognition*, Vol.31, No. 12, pp. 2055-2076, 1998.
2. L. O’Gorman, R. Kasturi.: Document Image Analysis. IEEE Computer Society, 1995
3. W.S. Baird, S. E. Jones, S. J. Fortune.: Image segmentation by shape directed covers. Proc. Of ICPR, pp. 820-825, 1990
4. A. K. Jain and S. Bhattacharjee, "Text Segmentation Using Gabor Filters for Automatic Document Processing," *Machine Vision and Applications*, 5(3), pp. 169-184, 1992.
5. D. X. Le and G. R. Thoma, "Document classification using connectionist models," *Proc. of IEEE International Conference on Neural Networks*, Orlando, Florida, vol. 5, pp. 3009-3014, June 1994.
6. J. Ohya, A. Shio and S. Akamatsu, "Recognizing Characters in Scene Images", *IEEE Transaction on PAMI*, Volume 16, No. 2, pp. 214-224, February 1994.
7. T. Pavlidis and J. Zhou, "Page Segmentation and Classification," *Computer Vision Graphics Image Processing*, 54(6), pp.484-496, November 1992.
8. D. Wang and S. N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics, and Image Processing*, vol. 47, pp. 327-352, Jan. 1989.
9. F. M. Wahi, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Graphics and Image Processing*, vol. 22, pp. 375-390, Feb. 1982.
10. Y. K. Ham, I. K. Kim, H.K. Chung, R. H. Park, C. B. Lee, S. J. Kim, and B. N. Yoon, "A study on the recognition of mixed documents consisting of texts and graphic images," *Journal of KITE*, vol. 31, no. 7, pp. 76-89, July 1994.
11. S. Imade, S. Tatsuta, and T. Wada, "Segmentation and classification for text/image documents using neural network," *Proceeding of the Second International Conference on Document Analysis and Recognition*, Tsukuba, Japan, pp.930-934, Oct. 1993.
12. J. S. Kim, J. C. Shim, J. H. Lee, and H. M. Choi, "Classification of document image blocks based on textual features and BP," *ISPACS '94*, Seoul, Korea, pp. 104-108, Oct.1994.
13. L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 910-918, Nov. 1988.
14. Y. Hirayama, "A block segmentation method for document images with complicated column structures," *Proc. of the second International Conference on Document Analysis and Recognition*, pp. 91-94, Tsukuba, Japan, Oct. 1993.
15. C. H. Chan, L. F. Pau and P. S. P. Wang, "Handbook of Pattern Recognition & Computer Vision", (2nd Edition), 1999.