

Page segmentation and text extraction from gray scale image in microfilm format

Q. Yuan, C. L. Tan

Department of Computer Science, School of Computing

National University of Singapore

3 Science Drive 2, Singapore 117543

yuanqing.tan@comp.nus.edu.sg

ABSTRACT

The paper deals with a suitably designed system that is being used to separate textual regions from graphics regions and locate textual data from textured background. We presented a method based on edge detection to automatically locate text in some noise infected grayscale newspaper images with microfilm format. The algorithm first finds the appropriate edges of textual region using Canny edge detector, and then by edge merging it makes use of edge features to do block segmentation and classification, afterwards feature aided connected component analysis was used to group homogeneous textual regions together within the scope of its bounding box. We can obtain an efficient block segmentation with reduced memory size by introducing the TLC. The proposed method has been used to locate text in a group of newspaper images with multiple page layout. Initial results are encouraging, we would expand the experiment data to over 300 microfilm images with different layout structures, promising result is anticipated with corresponding modification on the prototype of former algorithm to make it more robust and suitable to different cases.

Keywords: Page segmentation, text extraction, edge detection, gray scale image, microfilm format

1. INTRODUCTION

Segmenting an image into coherent regions is the first step before applying an object recognition method. For example, prior to using an optical character recognizer (OCR), the characters must be extracted from the image. In this paper, we address the problem of automatically finding text in a microfilm formatted grayscale image. The stated work is part of the project that facilitates the news article retrieval from the microfilm archives of the Singapore National Library. These microfilm images contain quite a lot of noise during scanning process. One of them was photocopied from the earliest issue of local English newspaper dated back to 1845. The characters in textual regions cannot be segmented by simple thresholding algorithm, and multiple size, font and orientation of the text made it more difficult to do segmentation process by a general method. As to solve this problem, there exist two classes of methods: texture-based and non-texture-based. The texture-based methods consider a document image as a composite of textures of different classes. With this approach, various well-known texture segmentation techniques can be used but should be modified correspondingly to fit to special cases. Our method belongs to this category. While the non-texture-based methods apply other image processing techniques (for example, to use various splitting-and-merging strategies) to this task and do not treat the image as texture. The segmentation problem can also be classified as Top-down (or model driven), Bottom-up (or data-driven) and Hybrid (Pavlidis and Zhou, 1992) methods. For the bottom-up method, which starts by first segmenting the document into small blocks such as characters, and then merges them into bigger blocks as words and then text lines. In top-down methods, a document is segmented from large components (high-level) to smaller, more detailed, sub-components (lower-level). It starts by segmenting the document into large blocks and then analyses them in order to achieve separation of the characters in the text blocks. Most of the top-down techniques are based on the *run length smoothing* (RLS) algorithm (Wong et al., 1982), also called the *constrained run length* (CRL) algorithm (Wahl et al., 1989).

Generally, for an efficient document image processing, document analysis system should be able to classify a document into several blocks in accordance with block characteristics and its applications. The criteria for the extent of the block classification cannot be defined clearly because it depends on the purpose or the application of the document analysis. But from the scanned newspaper images we can easily find these categories of the blocks of small font text body, big font

headlines, illustrating advertisement figures and some cartoon graphics. Hence, for an efficient document segmentation of these newspaper images, block features, which characterize and discriminate each of these blocks, should be extracted to enable the efficient block classification into at least those 4 block categories. In most of the previous works, the features for block classifications are extracted from the binarized document images.³⁻⁵ But these works have some fundamental difficulties such as the determination of the threshold value for the binarization and the lack of enough information for detailed block classification. So, for an efficient block classification, the feature extraction from gray images is required. In the works of some other researchers, features are extracted from the distribution of the gray value⁶ or the SGLDM (spatial gray level dependency matrix)⁷ of the gray image. But these works are sensitive to the background noise and brightness variation, and post-processing are required for character string extractions from character-containing blocks such as cartoon graphics or advertisement figures.

In our paper we observed that edges being detected at textual regions are sharply different from those of graphic and figure regions, it may help us to separate textual data from graphics as well as textured background. We propose an edge-based block segmentation and classification algorithm with automatic character string extraction. We exploit edge features of the gradient average, the normalized number of the edge pixels, the horizontal dominant edge contents from the gradient and orientation of the edge pixels. All of these features are insensitive to the background noise and the brightness variations of the document image because they are extracted only from the edge information such as edge features as well as the character and line contents of a block. Using these features, we can successfully separate different regions in the gray level newspaper image from each other.

The rest of paper is organized as follows: In Section 2, we describe in detail our approach for page segmentation and automatic text extraction from gray-scaled images. Experimental results are presented in Section 3. Finally, we conclude the paper in Section 4.

2. DESCRIPTION OF OUR 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 have been used implicitly in literature^{8,9} by considering text regions as having a certain texture with frequency components distinct from those of a gray scale figure. The regular alignment of characters into lines and columns has been used more explicitly in literature⁷, where text lines are assumed to form rectangles with distinct shapes and different lines are separated by white regions with no black pixels. The critical characteristic is that text (represented in particular font) has a dominant fixed stroke width and that this fact, coupled with the ratio mentioned above, may be used to identify text in images.

To extract characters from a more complex image, the connected component approach has been used in². An image is first segmented into candidate character regions using adaptive threshold and connected component analysis. A component can then be accepted as a character (or a part of a character) by applying additional heuristics, or classifying it with an OCR system.

Our approach is to segment the image into text/non-text regions as best as possible, and then let the OCR system do the detailed refinement. In other words, we would like to find all text areas and as few spurious non-text areas as possible.

A high level block-diagram of the algorithm for locating text is drawn in Figure 1.

The first 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. 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 textural areas. Moreover, there are steep edges corresponding to the minimal and maximal row coordinates of small letters, such as 'm', 'u', 'n'. Text lines can then be found by extracting the rows between two sharp edges of the spatial variance - one edge rising and the other falling. 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 block. In our experiments, we have chosen Canny edge detector to do the first step of operation.

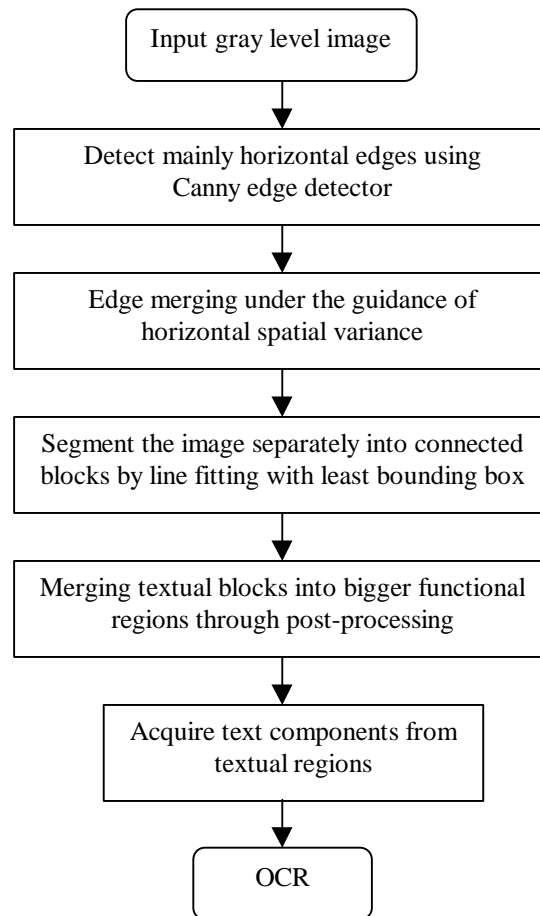


Figure 1. Block diagram of the algorithm for Page segmentation and Text extraction.

2.1. Horizontal Edge Detection

We choose Canny edge detector in our system to detect edges from grayscale images since Canny operator has several advantages: It has low probability of missing edge point, 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.

The Canny edge detector optimizes the product of signal-to-noise ratio and localization. The algorithm is summarized by the following notation. 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,

$$S[i, j] = G[i, j; \mathbf{s}] \times I[i, j] \quad (1)$$

Where s is the spread of the Gaussian and controls the degree of smoothing.

The gradient of the smoothed array $S[i, j]$ can be computed using the 2×2 first-difference approximations to produce two arrays $P[i, j]$ and $Q[i, j]$ for the x and y partial derivatives:

$$P[i, j] = (S[i, j+1] - S[i, j] + S[i+1, j+1] - S[i+1, j]) / 2 \quad (2)$$

$$Q[i, j] = (S[i, j] - S[i+1, j] + S[i, j+1] - S[i+1, j+1]) / 2 \quad (3)$$

The magnitude and orientation of the gradient can be computed from the standard formulas for rectangular-to-polar conversion:

$$M[i, j] = \sqrt{P[i, j]^2 + Q[i, j]^2} \quad (4)$$

$$O[i, j] = \arctan(Q[i, j], P[i, j]) \quad (5)$$

Since the problem of finding locations in the image array where there is rapid change has merely been transformed into the problem of finding locations in the magnitude array $M[i, j]$ that are local maxima. To identify edges, the broad ridges in the magnitude array must be thinned so that only the magnitudes at the points of greatest local change remain. This process is called non-maxima suppression, which in this case results in thinned edges. Non-maxima suppression thins the ridges of gradient magnitude in $M[i, j]$ by suppressing all values along the line of the gradient that are not peak values of a ridge. The algorithm begins by reducing the angle of the gradient $O[i, j]$ to be mainly around horizontal direction.

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Figure 2. Masks used to compute gradient components

The algorithm passes a 3×3 neighborhood across the magnitude array $M[i, j]$. At each point, the center element $M[i, j]$ of the neighborhood is compared with its two neighbors along the line of the gradient given by the sector value $I[i, j]$ at the center of the neighborhood. If the magnitude array value $M[i, j]$ at the center is not greater than both of the neighbor magnitudes along the gradient line, then $M[i, j]$ is set to zero. This process thins the broad ridges of gradient magnitude in $M[i, j]$ into ridges that are only one pixel wide. The values for the height of the ridge are retained in the nonmaxima-suppressed magnitude.

Let $N[i, j] = nms(M[i, j], L[i, j])$ denote the process of nonmaxima suppression. The nonzero values in $N[i, j]$ correspond to the amount of contrast at a step change in the image intensity. In spite of the smoothing performed as the first step in edge detection, the nonmaxima-suppressed magnitude image $N[i, j]$ will contain many false edge fragments caused by noise and fine texture. The contrast of the false edge fragment is small.

To reduce the number of false edge fragments in the non-maxima suppressed gradient magnitude is to apply a threshold to $N[i, j]$. All values below the threshold are changed to zero. The result of applying a threshold to the nonmaxima-suppressed magnitude is an array of the edges detected in the image $I[i, j]$. A more effective thresholding scheme uses two thresholds.

The double thresholding algorithm takes the nonmaxima-suppressed image, $N[i, j]$, and applies two thresholds t_1 and t_2 , with $t_2 = 2t_1$, to produce two thresholded edge images $T_1[i, j]$ and $T_2[i, j]$. Since image T_2 was formed with a higher threshold, it will contain fewer false edges; But T_2 may have gaps in the contours (too many false negatives). The double thresholding algorithm links the edges in T_2 into contours. When it reaches the end of a contour, the algorithm looks in T_1 at the locations of the 8-neighbors for edges that can be linked to the edges in T_2 into contours. When it reaches the end of a contour, the algorithm looks in T_1 at the locations of 8-neighbors for edges that can be linked to the contour. The algorithm continues to gather edges from T_1 until the gap has been bridged to an edge in T_2 .

The steps of Canny 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 nonmaxima suppression to the gradient magnitude.
4. Use the double thresholding algorithm to detect and link edges.

2.2. Edge merging

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 edge pixels. Components that have a large vertical spread are not good candidates for upper and lower boundaries of a text line, and then can be removed. Among the remaining lines, we find the ones that have almost the same row coordinates, 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 chose a relatively wide range of threshold to do merging from detected edges. The method in our implementation tries to link edge points within the threshold, retain those positions which exceed the given threshold, as a result we can cluster the linked edges to form roughly rows of text lines.

2.3. Noise reduction by Thin Line Coding (TLC)

After edge linking and merging process, there would still exist quite a lot of noise components such as spurious lines in the edge image, we need to find a highly efficient method to reduce these noises on relatively low cost. As always, the critical step in performing noise reduction is to distinguish noise and signal. In our case, noisy lines might be isolated lines, spurs, and loops that are shorter than given thresholds, or lines may have gaps. One approach to reduce this type of noise in line images is to use matched line filters. This involves convolution of the image with filters matching one-, two-, or three-pixel lines and marking them for elimination. But it is impractical for longer line features, given that the number of possible matched filter masks greatly increases with the line length. We chose a more versatile approach based on a hierarchical representation called the Thin Line Coding (TLC) provided by O’Gorman. TLC operates on chains and line features instead of pixels. The lowest level of TLC is PCC (Pixel Chain Code); higher levels are line segments, line composites (made up of multiple, joined line segments), and complete line structures (containing all joined lines) at the top level. 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. In addition, the hierarchical representation permits more complex operations such as isolation and retention of only long, short, or mid-length lines, analogous to band-pass filtering.

TLC is used in the same manner as filters are used. First, features of interest are selected and range of feature dimensions, or of noise dimensions, is set. This is done with knowledge of the characteristics of desired signal or undesired noise. 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 there are many fewer TLC structures and features than there are pixels. Take the example of image in Figure 4. The document image contains 2550×3000 , or 8.4 million pixels. However, after thinning is performed on the pixel image, followed by PCC coding, the resulting representation typically requires only ~1% of the memory required to store the original image. TLC encoding leads to further compression: for the example of the last image, there are around 500 to 1000 line structures including characters,

dots, etc., and hence only ~0.01% of 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.

2.4. Strait-Line fitting

We can approximate the chain of edge points with a straight line running through most of them, finding the optimal straight-line fit to a given group of edge points is our final goal. These edge points were extracted from the image by edge detection and contour linking analysis, to be connected in a curve. In our system we want to find the location of text lines that is approximately encircled by pairs of linked edge lines. There exist two methods to obtain optimal fitting of straight lines to a chain of points. First, the least-squares method that is widely used in statistical analysis when one variable, y , is dependent on the other, x . Second, another 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 the latter method is more suitable for our system.

2.4.1. Line Segment Fitting by Evaluation of Eigenvectors

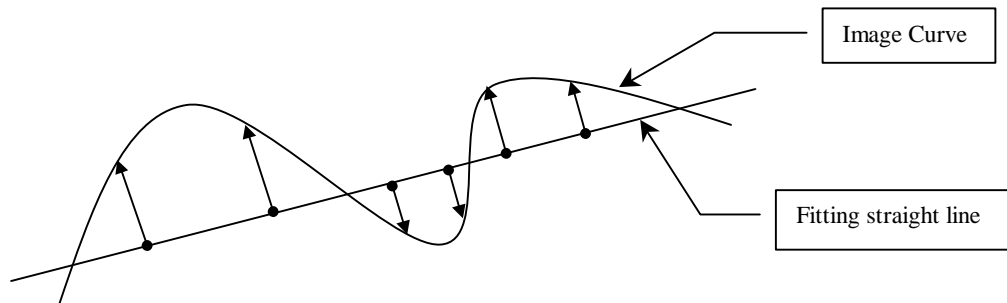


Figure 3. Image curve and its corresponding line fit using eigenvector method, in which the y -axis error is measured perpendicular to the orientation of the resultant line fit.

The eigenvector strategy is to minimize error in the direction perpendicular to the fitted line. This approach avoids the dependence of E_x and E_y on the slope of the optimal fit, with the worst cases at close-to-vertical or close-to-horizontal lines, as described above. An example of eigenvector line fitting is shown in Figure 3.

For a set of points $\{(x_i, y_i), 0 \leq i \leq N-1\}$, the objective in eigenvector line fitting is to find the equation of a line that minimizes the error E_Δ , defined in terms of the perpendicular distance d_i between the point with x coordinate x_i and the optimal line:

$$E_\Delta \equiv \sum_{i=0}^{N-1} d_i^2 \tag{6}$$

The steps to obtain the desired optimal fit is stated as follows:

1. Calculate averages of x and y coordinates of the given set of points:

$$\bar{x} = \frac{1}{N} \sum x_i \tag{7}$$

$$\bar{y} = \frac{1}{N} \sum y_i \tag{8}$$

2. Standardize each data point by subtracting the averages just calculated:

$$\hat{x}_i = x_i - \bar{x}, \tag{9}$$

$$\hat{y}_i = y_i - \bar{y} \tag{10}$$

3. Determine the symmetric matrix A , defined in terms of the sum of vectors to each point $v_i = (x_i, y_i)$; with

$$a = \hat{x}_i^2, b = \hat{x}_i \hat{y}_i, c = \hat{y}_i^2: \quad (11)$$

$$A = \sum v_i^\perp v_i = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \quad (12)$$

4. Determine the smaller eigenvalue, \mathbf{I} , and the associated eigenvector, e_1 , of A :

$$\mathbf{I} = [(a + c) - \sqrt{(a + c)^2 - 4ac + 4bb}] / 2 \quad (13)$$

$$e_1 = [b / (\mathbf{I} - a), 1]. \quad (14)$$

5. Find the slope of the desired optimal line as that perpendicular to the slope of the eigenvector:

$$m = -b / (\mathbf{I} - a), \quad (15)$$

If $a = \mathbf{I}$, the slope is the vertical.

6. Find the intercept y_0 of the desired optimal line from the average of the data set:

$$y_0 = \bar{y} - m\bar{x}. \quad (16)$$

2.5. Region merging and post-processing

The final step is to group together pairs of lines with opposite orientations. The huge number of all possible groupings is reduced by using the following heuristics which is a bit similar to the one provided by Jain. ²:

- The distance between a pair of edge lines must be in the scope of suitable position.
- There must be no other lines between the two fitted edge lines in a pair.
- A significant portion of the two straight lines must overlap when they are projected vertically.
- The distance between the upper and the lower edge should not be very small or very large. This heuristic constrains the character size.

From a pair of straight lines, it is a simple matter to construct the smallest rectangle that contains the two 'paired' lines in its upper and lower edges. All such rectangles form the output of this step of the algorithm - bounding boxes around candidate text regions. As we cannot acquire 100% correct textual blocks from above steps, a post-processing is needed for region grouping. We try to find relations between different regions with Geometrical and Textural properties of them. In our method we find two major features to separate textual regions from graphic regions, which help us to merge text blocks together accurately.

The first feature F_a , edge gradient average, is defined as:

$$F_a = \frac{\sum_{i=T_h}^{G_m} H(i) \cdot i}{\sum_{i=T_h}^{G_m} H(i)} \quad (17)$$

Where T_h is the threshold value, G_m is the maximum value of the gradient, and $H(i)$ is the histogram of the edge gradient. The edge gradient of the figure block, which has small gray level difference between the object and the background, is smaller than that of the text block. Therefore, we can discriminate figure block from the text block using F_a .

The second feature F_n , normalized number of edge pixels, is defined as:

$$F_n = \frac{\sum_{i=Th}^{G_m} H(i)}{N_x \cdot N_y} \quad (18)$$

Where N_x , N_y are the width and the height of the block, respectively. This feature represents the density of the edge pixels in a block. In general, the graphic blocks, which contain much blank space, has the lower density of edge pixels in contrast to the text blocks. So, we can use F_n to discriminate graphic block from the text block.

With enough statistical data acquired in our experiment we can easily group textual regions into larger text blocks. Connected component analysis enhanced by the above two features was used in grouping process to filter out some irrelevant noise components.

3. EXPERIMENT AND RESULTS

The proposed method was tested on 10 microfilm formatted newspaper images. Image size varied from 512×512 pixels to 2550×3072 pixels. Most of text blocks may be successfully separated from graphics regions as well as the gray textured background. The algorithm for locating text is relatively fast. The spatial variance method and edge detection performance require around two scans over the image to compute variances, and the same number of scans to perform the connected component analysis of the linked edge lines. Additional computations for grouping homogeneous components depend on how effective data structure can be designed and implemented in our program.

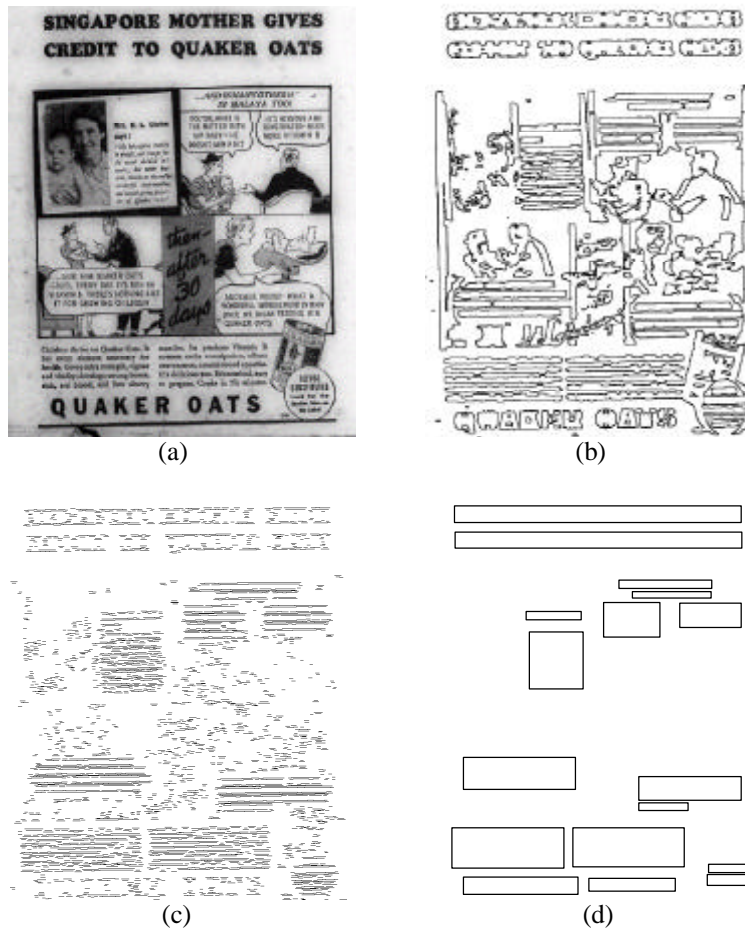


Figure 4. Experimental results. (a) Portion of original gray level image which has cartoon graphics; (b) Resultant image after edge detection and edge merging in horizontal direction; (c) Pairs of line segments being retrieved by eigenvector line fitting process; (d) Output image from connected component analysis for finding the bounding rectangles.

One shortcoming of our algorithm is that it fails to locate text blocks that are not well separated from the background, and characters that are not aligned. Thus, we still need to update our algorithm to make it more robust to the variations in font, size and shape of the text. Actually we are still working on correcting the observed errors in the process of edge linking. Higher rate of accuracy could be anticipated after whole modifications have been made.

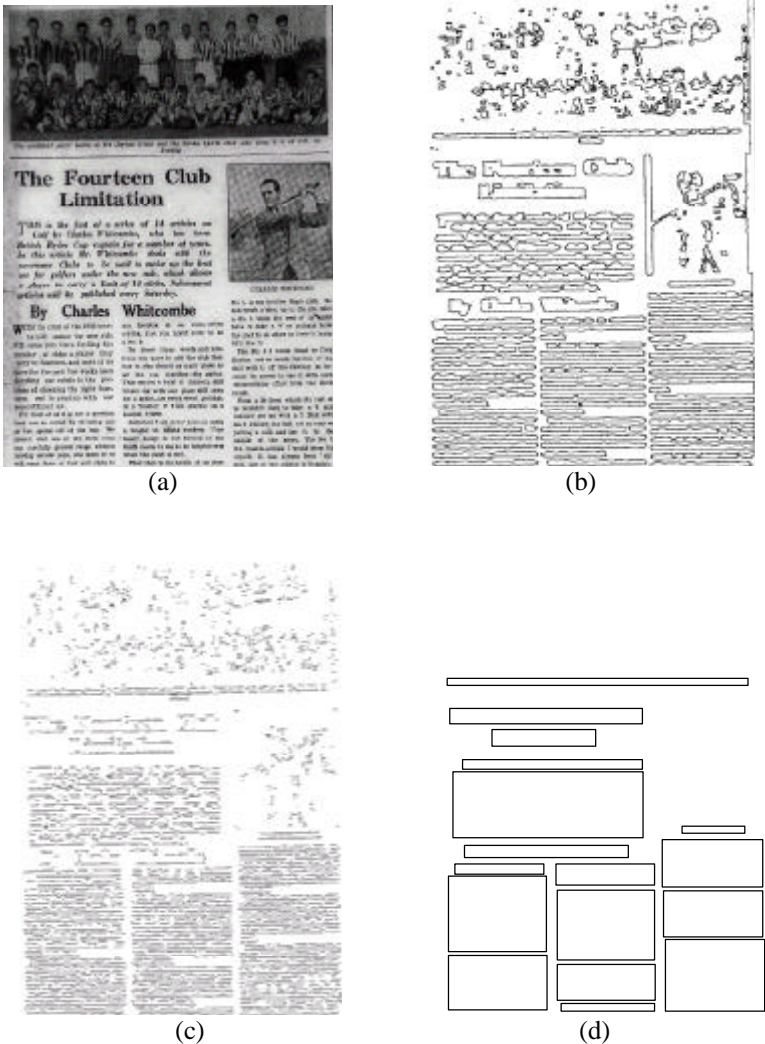


Figure 5. Another experimental result set. (a) Portion of original gray level image which has standard newspaper layout; (b) Edges being detected and merged from horizontal direction; (c) Line segments achieved by eigenvector approximation method; (d) Output image from connected component analysis to locate textual area in their bounding boxes.

4. CONCLUSIONS

Segmentation is an important issue in the automated document analysis research area. Text extraction plays a significant role in document retrieval and storage systems. This paper proposes a relative new segmentation method that clusters the regions of a mixed-type document image into text and non-text areas. The method uses feature extraction to detect edges of

original image. It consists of the following main stages: extraction of edges, edge linking and merging, line fitting approximation, filtering of spurious blocks according to their features, creation of chains of connected blocks, surrounding of chains by bounding rectangles, and finally, clustering the regions of the bounding rectangles as text or non-text areas.

In comparison with other techniques, this method has two advantages. First, we used edge features to do the segmentation process. It is relative noise resistant and then robust to different cases. Second, TLC was introduced into the line fitting process and it brings a fast algorithm as well as saves a lot of memory usage.

The proposed segmentation method was extensively tested on many microfilm based newspaper images. It can be used to locate horizontally aligned text only from current description. In principle, it can be extended to find text in any orientation, provided that the text is aligned between two parallel lines. At the same time the method is independent of the size and type of the characters and the position of the text in the document, and it is insensitive to small skews. The results were promising, almost all the text lines could be retrieved from graphic and figure regions under normal scanning conditions. The average processing time of a page is about 20 second, on a Pentium-II-400 computer.

ACKNOWLEDGMENTS

We would like to thank Singapore Press Holding group to provide all the microfilm formatted newspaper images.

REFERENCE

1. Jiangying Zhou, Daniel Lopresti, "Finding Text in Color Images", *Proc. SPIE'98, Document Recognition V*, pp. 130-140, 1998.
2. 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.
3. G. Cortelazzo, G. A. Mian, G. Vezzi and P. Zamperoni, "Trademark shapes description by string matching techniques", *Pattern Recognition*, Vol. 27, No. 8, pp. 1005-1018, 1994
4. C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic and W. Equitz, "Efficient and Effective Querying by Image Content". *Journal of Intelligent Information Systems*, 3, pp. 231-262, 1994.
5. 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.
6. 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.
7. 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.
8. T. Pavlidis and J. Zhou, "Page Segmentation and Classification," *Computer Vision Graphics Image Processing*, 54(6), pp.484-496, November 1992.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, New York, 1992.