Automated Borders Detection and Adaptive Segmentation for Binary Document Images

Daniel X. Le* , George R. Thoma*, and Harry Wechsler**
*National Library Of Medicine, 8600 Rockville Pike, MS 55, Bethesda, MD 20894
**Department Of Computer Science, George Mason University, Fairfax, VA 22030

Abstract

This paper describes two new and effective algorithms: one for detecting the page borders for documents available as binary images, and the other an adaptive segmentation algorithm using a bottom-up approach for segmenting binary images into blocks. The borders detection algorithm relies upon the classification of blank/textual/non-textual rows and columns, objects' segmentation, and an analysis of projection profiles and crossing counts. Segmentation, done by an adaptive smearing technique, is different from all previous bottom-up approaches because any decisions on merging and/or separating are based on the estimated font information in binary document images.

Index Terms - Bottom-up approach, page borders, segmentation, document processing, smearing algorithm.

1. Introduction and background

Electronic conversion of paper documents is increasingly of importance in automated document delivery. The Lister Hill National Center for Biomedical Communications, a research and development division of the National Library of Medicine (NLM), is conducting research in the area of binary document processing. This paper describes two stages of this ongoing effort at NLM: the detection and removal of page borders, and the block segmentation of page images.

When a page of a book is scanned, text from an adjacent page may also be captured into the current page image. These unwanted regions are called "textual borders." Whenever a scanned page does not completely cover the scanner setup image size, there will usually be "non-textual borders" in an image. Such borders are dominated by non-textual data (graphics, line art, or dithered images). Moreover, gutters - 'the white space formed by the adjoining inside margins of two facing pages as of a book or magazine [1].' - may also be scanned in and result in gutter borders. Fig. 1 shows textual borders, non-textual borders, and gutter borders.

There appear to be very few techniques recommended for page borders detection. An example is the commercial ScanFix™ [2] software that advertises a method to detect and remove page borders. However, there is no document describing the method. For page blocks segmentation, other proposed techniques are based on either a top-down or bottom-up approach. Wahl et al. [3] proposed a segmentation using the constrained run length algorithm while Nagy et al. [4] proposed a top-down segmentation strategy called RXYC. Both methods are fast, but they perform poorly if the document page is skewed. Pavlidis et al. [5] described a top-down segmentation that is based on the strength and variation of correlation of adjacent scanlines. While this technique was reported to perform well on a variety of printed documents, no information was provided on the size of the experiments and performance statistics. Fletcher et al. [6] described a bottom-up approach algorithm based on typesetting knowledge. This algorithm is robust, but it is very slow. Gorman [7] recently proposed a method based on k-nearest-neighbor clustering of page components. This method deals only with the textual parts of a page image. Jain et al. [8] suggested a bottom-up segmentation using Gabor filters. It is not obvious that this method would carry over to binary data, and that the associated costs will remain affordable for real-time document analysis.

All previous bottom-up algorithms assume that font sizes are between 10-point and 12-point. Character leading and word spacing are then calculated based on this assumption for segmentation and merging processes. However, this assumption is often violated because document images may be scaled up/down for reading and quality purposes. Also many document pages have font sizes below 10-point or 12-point. Our algorithms are different from all previous bottom-up approaches because any decisions on merging and/or separating are based on the estimated font information in document images.

These two proposed algorithms are part of a document processing system being proposed by us [9,10,11]. A high-level diagram of our document processing system is shown in Fig. 2 and it consists of two major components:

1015-4651/96 $5.00 © 1996 IEEE
Proceedings of ICPR '96

preprocessing and document analysis. In this paper, we only present the page borders detection and removal process and the page blocks segmentation process. For the other three processes, the interested reader might refer to references [9,10,11] for more information.

The rest of this paper is divided into five sections. Section 2 provides the basic definitions. Sections 3 and 4 discuss in detail our algorithms. Section 5 discusses experimental results. Section 6 contains conclusion.

2. Basic features

A horizontal/vertical projection histogram is the sum of black pixels projected onto the vertical/horizontal axis. A crossing count histogram represents the number of times that pixels in a row/column turn from 0 to 1.

A textual square is defined as an area in which text data is dominant. A non-textual square is an area in which blank or non-textual data is dominant.

A textual row/column is a row/column in which textual data is dominant while a blank row/column consists of majority of white pixels. A non-textual row/column is a row/column in which non-textual data is dominant.

Page orientation is defined here as the printing direction of text lines. Therefore the page orientation can be in either portrait mode (horizontal printing) or landscape mode (vertical printing).

Smearing algorithm replaces 0's by 1's if the number of adjacent 0's is less than or equal to a given constraint C.

Font information includes character point size, word spacing, character leading, ascenders, descenders, baseline, and middle line. Character point size is equivalent to a character height (also known as an "em"). Word spacing is the white space between consecutive words. An extra white space between adjacent lines of text lines is called the character leading. An ascender is 'the part of a lowercase letter that exceeds x height letter (as in "b" or "t") [1].' A descender is 'the part of a lowercase letter (as q, p, y) that is lower than the lowest part of an x-height letter (as o, a, e) [1].' A bottom line of non-descender characters is called the baseline while a top line of non-ascender characters is the middle line. Fig. 3 explains font information.

3. Page borders detection and removal process

The page borders detection and removal process is based on classification of blank/textual/non-textual rows and columns, locations of border objects, and an analysis of projection profiles and crossing counts of textual squares. In most cases, borders are very close to edges of a binary document image and they are separated from page contents by white areas. Spacings between pixels of non-textual borders are much closer than those between characters. Consequently, using the estimated font size, the page contents and page borders can be roughly separated and preliminary calculations of the areas of textual and non-textual borders can also be made. Objects within these preliminary areas are analyzed to adjust borders locations. The page borders detection and removal process consists of four steps and each step will be discussed in detail in the following subsections.

3.1 Font size estimation

The binary image is divided into squares whose size is dependent on image resolution (for 200 dpi, the size is about 80 pixels). Each square is then classified as a textual square or non-textual square. Any square that satisfies the following empirical condition is a textual square.

In a square, a top-half square, a bottom-half square, a left-half square, or a right-half square

BlankRATIO < (Total black pixels/Total pixels) < GraphicsRATIO

The empirical values of BlankRATIO and GraphicsRATIO are 0.045 and 0.444 respectively. Font information can then be estimated by analyzing shapes of projection histograms of all textual squares. Horizontal histograms are used for portrait images and vertical histograms are for landscape images. The projection histogram of a textual square usually consists of a series of black/white stripe patterns. These patterns must satisfy the following two empirical conditions:

(a) White stripe length ≥ 1 and

(b) MinFONTSIZE $\leq$ Black stripe length $\leq$ MaxFONTSIZE

In this paper, we used 3-point for MinFONTSIZE and 18-point for MaxFONTSIZE. Let H_{top} represent the distance from the character baseline to the ascender top, H_{bot} represent the distance from the character baseline to the ascender top of the character underneath, and let H_{mid} be the distance from the baseline to the middle line of a character. Fig. 3 shows all distances defined in the above. The font information can be estimated as follows:

(1) Calculate H_{top} and H_{bot} using black and white stripe lengths of all textual squares and calculate B_{max}, the maximum black stripe length among all textual squares,

(2) For each textual square, build another horizontal or vertical black/white stripe histogram where black stripe lengths, in addition to satisfying the above empirical condition (b), must be greater than or equal to one-half of B_{max}, and calculate H_{mid}, the black length whose total numbers within all black stripes are maximum,

(3) Ascender height = Descender height = $H_{top} - H_{mid}$.
Character height = H_{mid} + Ascender height + Descender height.
Character leading = H_{bot} - Descender height.
Word spacing = (1/4 to 1/2) of Character height.

3.2 Areas of non-textual borders estimation

Areas of non-textual borders can be roughly located by identifying non-textual rows/columns around image edges, and checking distances from these non-textual rows/columns to textual and blank rows/columns against word spacing and character leading. The word spacing used in this process is chosen to be one-quarter of a character height will be. Let W_x be the word spacing and W_y be one half of the character leading. The following shows how to estimate areas of non-textual borders:

(1) Calculate the horizontal and vertical projection histograms of areas around image edges.

(2) Classify rows and columns as follows:

Non-textual: if (Total black pixels/Total pixels) > GraphicsRATIO
Textual : otherwise

(3) Locate left/right/top/bottom non-textual borders as follows:

The left/right border starts from the left/right image edge and stops when the number of consecutive textual or blank columns is greater than W_x (for portrait image), or W_y (for landscape image).

The top/bottom border starts from the top/bottom image edge and stops when the number of consecutive textual or blank rows is greater than W_y (for portrait image) or W_x (for landscape image).

Line 1-1 in Fig. 1 shows an estimated area of a bottom non-textual border.

3.3 Areas of textual borders estimation

This step is similar to the previous step 3.2 except that the areas of non-textual borders are excluded from all of its calculation. Crossing count histograms are also created to enhance the classification of rows and columns. Distances from textual and non-textual rows/columns to blank rows/columns are compared to word spacing and character leading to locate areas of textual borders. The following shows how to estimate areas of textual borders:

(1) Update the horizontal and vertical projection histograms of areas around image edges excluding all areas of non-textual borders,

(2) Smear horizontally with constraint W_x (for portrait image) or W_y (for landscape image). Smear vertically with a constraint W_y (for portrait image) or W_x (for landscape image),

(3) Create the horizontal and vertical crossing count histograms of areas around image edges excluding all areas of non-textual borders,

(4) Classify rows or columns as follows.

Blank: if (Total black pixels/Total pixels) < BlankRATIO and if Total crossing counts < BlankXCOUNTS
Non-textual: if (Total black pixels/Total pixels) > GraphicsRATIO
Textual: otherwise

(5) Locate left, right, top, or bottom textual borders as follows:

The left/right border starts from the left/right image edge (excluding non-textual left/right borders) and stops when the number of consecutive blank columns is greater than W_x (for portrait image) or W_y (for landscape image).

The top/bottom border starts from the top/bottom image edge (excluding non-textual top/bottom borders) and stops when the number of consecutive blank rows is greater than W_y (for portrait image) or W_x (for landscape image).

The empirical value of BlankXCOUNTS is 0.010. Line 2-2 in Fig. 1 shows an area of a left textual border.

3.4 Page borders determination and removal

This step readjusts page borders to minimize the existence of any gutter borders or any border remnants within an image content area (the desired area of the page containing information). By segmenting objects around edges of the content area and checking their distances against image edges, gutter borders and border remnants are located and removed. The following procedure describes step-by-step how to determine page borders:

(1) Smear horizontally with constraint W_x (for portrait image) or W_y (for landscape image) and smear vertically with constraint W_y (for portrait image) or W_x (for landscape image),

(2) For the top and bottom edge areas of the image content, perform objects' segmentation, calculate objects' coordinates, and discard any objects whose sizes are less than or equal to MinFONTSIZE.

(3) Adjust the top/bottom border by (a) locating objects that are not connected to the top/bottom border and (b) setting the new top/bottom border as the minimum/maximum top/bottom row among these objects and as the new image content area.

(4) For the left and right edge areas of the new image content, perform objects' segmentation, calculate objects' coordinates, and discard any objects whose sizes are not greater than MinFONTSIZE.

(5) Adjust the left/right border by a) identifying objects that are not connected to the left/right border, the new top border, and the new bottom border and b) setting the new left/right border as the minimum/maximum left/right column among these objects.

(6) Remove page borders of an image using the new borders.

Lines 1-1, 2-2, and 3-3 in Fig.1 shows all page borders.

4. Page blocks segmentation process

The page blocks segmentation process segments binary document images into blocks using an adaptive smearing algorithm in which any decisions on merging and/or separating are based on the estimated font information of binary document images. As mentioned previously, conventional typesetting has gaps (leading) between adjacent text lines and spacings between words are proportional to character height. Font information and page orientation are used to derive the constraints for the smearing algorithm. The page blocks segmentation process is an adaptive bottom-up approach and it consists of three steps and each step will be discussed in detail in the following subsections.

4.1 Font size estimation

If a binary image is not modified (scaled up or down) after going through the page borders detection process, the results obtained from step 3.1 can be used for this step.

4.2 Smearing constraints calculation

Normally, the word spacing depend on the character height and the recommended normal word spacing value is one-third of the character height. The horizontal and vertical smearing distances S_x and S_y, respectively, should be selected to minimize the following two conditions: (1) words belonging to different columns are not connected and (2) words belonging to different rows are not connected. For a portrait image, S_x is chosen to be the normal word spacing, while S_y is three-quarters of the character leading. For a landscape document image, the chosen values of smearing distances are reversed.

4.3 Horizontal and vertical smearing

A binary document image can be segmented into blocks using the smearing algorithm. Images are segmented by sequentially smearing in both horizontal and vertical directions using smearing distances S_x and S_y selected previously.

(1) For a portrait image, smear horizontally with S_x and then smear vertically with S_y to produce the segmented image.

(2) For a landscape image, smear vertically with S_y and then smear horizontally with S_x to produce the segmented image.

The segmented binary image obtained by applying the smearing algorithm with constraints $S_x \sim 9$ pixels and $S_y \sim 4$ pixels to the image in Fig. 1 is shown in Fig. 4

5. Experiment results

All 497 binary images used in this experiment are 8.5 x 11 inches and were scanned at 200 dpi resolution. These images were selected from several different medical journals and represent a wide range of font sizes. For all of these images, the estimated font sizes are within ±2 pixels of their actual sizes. For the page borders algorithm, all textual and non-textual borders existing in 495 images are removed, including gutter borders. As mentioned above, the assumption is that borders are very close to edges of images and borders are separated from image contents by a white space. Therefore, the algorithm may not be successful if the borders overlap the entire edges of an image content area. However, this case is rare, happening only twice in our test set. As a result, our algorithm successfully cleaned up page borders at an accuracy rate of 99.6%. For the page blocks segmentation algorithm, all images were correctly segmented into blocks. For images having more than one font type, words of smaller font in different rows sometimes are joined together. However, these rows could be separated easily by forming a projection histogram on the original image using blocks' coordinates and dividing blocks using the white space between text lines.

6. Conclusion

We have presented algorithms for detecting the page borders of binary document images and for segmenting binary document images into blocks. Our algorithms can handle both textual and non-textual data and are different from all previous bottom-up approaches because any decisions on merging and/or separating are based on estimated font information of document images. As a result, our algorithms are able to handle effectively a wide variety of document images having different font sizes. Evaluation results show that our algorithms successfully cleaned up page borders at an accuracy rate of 99.6% and correctly segmented binary document images into blocks.

References

1. Webster's 3rd New International Dictionary, G & C Merriam, Springfield, MA (1968).
2. Sequoia Data Corp., ScanFix Image Optimizer for MS/DOS.
3. F. M. Wahl, K. Y. Wong and R. G. Casey, "Block Segmentation and Text Extraction in Mixed Text/Image Documents," CGIP 20: 375-390 (1982).
4. G. Nagy, S. Seth, and M. Viswanathan, "A Prototype Document Image Analysis System for Technical Journals," IEEE Computer, pp. 10-22 (1992).
5. T. Pavlidis and J. Zhou, "Page Segmentation and Classification," CVGIP 54(6): 484-496 (1992).
6. L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," IEEE Trans. on PAMI 10: 910-918 (1988).
7. L. O'Gorman, "The Document Spectrum for Page Layout Analysis," IEEE Trans. on PAMI 5(11): 1162-1173 (1993).
8. A. Jain and S. Bhattacharjee, "Text Segmentation Using Gabor Filters for Automatic Document Processing," Machine Vision and Applications 5: 169-184 (1992).
9. D. X. Le, G. R. Thoma, and H. Wechsler, "Automated Page Orientation and Skew Angle Detection for Binary Document Images,", Pattern Recognition 27(10): 1325-1344 (1994).
10. D. X. Le, G. R. Thoma, and H. Wechsler, "Document Image Analysis using Integrated Image and Neural Processing,", Proc. IEEE Third ICDAR, Vol. I, pp.327-330 (1995).
11. D. X. Le, G. R. Thoma, and H. Wechsler, "Classification of Binary Document Images into Textual or Non-Textual Data Blocks Using Neural Network Models,", Machine Vision, Vol 8, pp. 289-304 (1995).

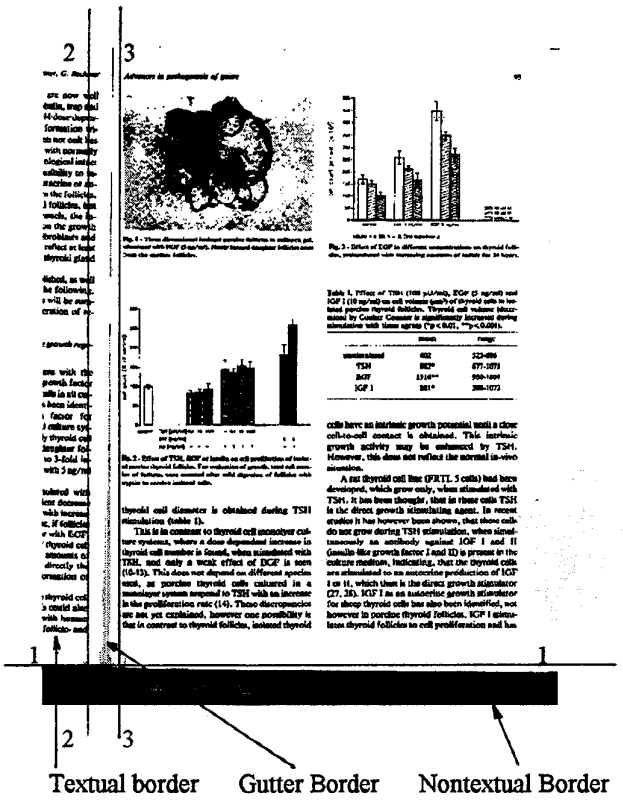

Figure 1

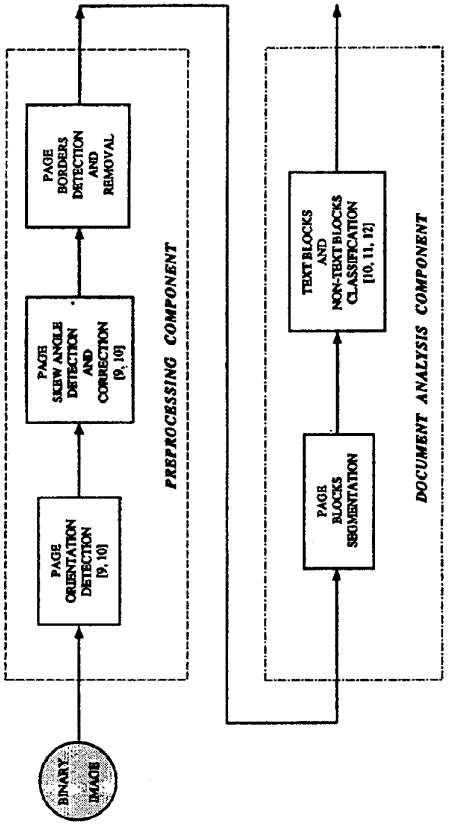

Figure 2

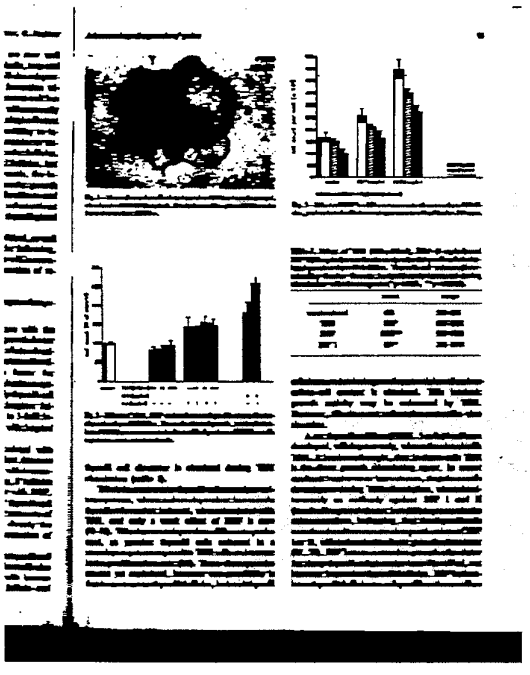

Figure 4

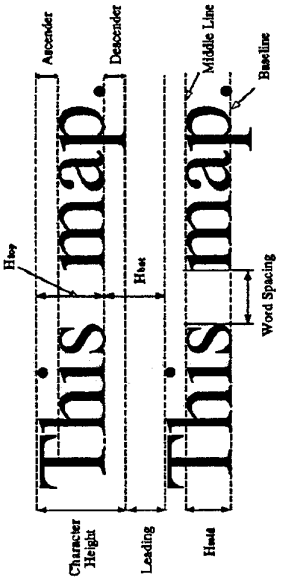

Figure 3