Locating Text in Complex Color Images

Yu Zhong, Kalle Karu and Anil K. Jain
Department of Computer Science
Michigan State University
East Lansing, MI 48824, USA
zhongyu@cps.msu.edu, karukall@cps.msu.edu, jain@cps.msu.edu

Abstract

There is a substantial interest in retrieving images from a large database using the textual information contained in the images. An algorithm which will automatically locate the textual regions in the input image will facilitate this task; the optical character recognizer can then be applied to only those regions of the image which contain text. We present a method for automatically locating text in complex color images. The algorithm first finds the approximate locations of text lines using horizontal spatial variance, and then extracts text components in these boxes using color segmentation. The proposed method has been used to locate text in compact disc (CD) and book cover images, as well as in the images of traffic scenes captured by a video camera. Initial results are encouraging and suggest that these algorithms can be used in image retrieval applications.

1 Introduction

Image segmentation 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 complex color image. By a complex image we mean that the characters cannot be segmented by simple thresholding, and the color, size, font and orientation of the text are unknown. Applications of locating and recognizing text include address localization on envelopes, scanning and understanding a printed document, identifying products (such as books, CDs, canned food, etc.) by reading the text on their covers or labels. Another application area of a text understanding system is in image database retrieval, where the capabilities of usual textual database systems are extended to image databases [2]. Considerable work has been done with image databases of trademarks [1, 5], where it is important to check if a new trademark is similar to any of the thousands of existing trademarks.

Previous research in recognizing text has focused mainly on character recognition. Text is often assumed to be printed in black on a white background, so that it can be extracted by thresholding the gray-scale image. Considerable research has been done in the area of page layout segmentation, where the main problem is to discriminate text areas from figures (half-tones), both of them being relatively easy to extract from a white background. The difference between text and figure is obvious to a human reader, but difficult to formalize for an automatic system. A number of heuristics have been used to locate text. Some of them are summarized below.

The most intuitive characteristics of text is its regularity. Printed text consists of characters with approximately the same size and line thickness which are located at a regular distance from each other. Such regularities have been used implicitly in [3, 4] 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 [7], where text lines are assumed to form rectangles with distinct shapes, and different lines are separated by white regions with no black pixels. These methods were successfully applied to obtain page layout of technical journals.

To extract characters from a more complex scene, the connected component approach has been used in [6]. An image is first segmented into candidate character regions using adaptive thresholding and connected component analysis. A component can then be accepted as a character (or a part of a character) by applying additional heuristics, or simply classifying it with an OCR system which has a class for 'non-character' reject option.

We note that, in a general setting, it is impossible to extract text or characters from a complex scene without first recognizing it. However, feeding an OCR

0-8186-7128-9/95 $4.00

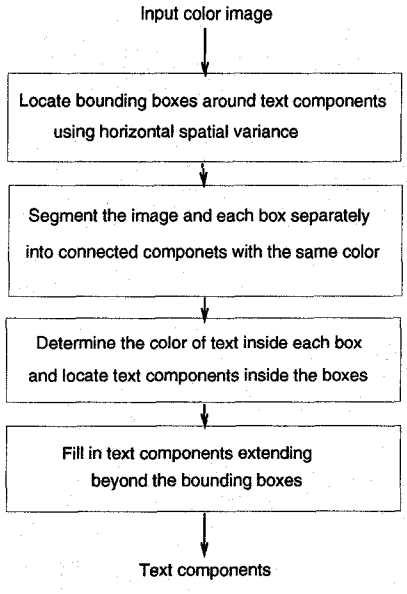

Figure 1: Block diagram of the algorithm for extracting text.

system with all possible components in an input image is not reasonable because a typical image may have thousands of components. Our approach is to segment the image into text/no-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, without actually classifying the characters.

In the next section, we describe the algorithm for extracting text. We use a combination of spatial variance and connected component techniques to place a bounding box around candidate text regions and to locate character components inside these boxes. Section 3 describes experiments done with various color images. In Section 4 we summarize the paper and list the limitations of the proposed method.

2 Locating Text in an Image

A high-level block-diagram of the algorithm for locating text is drawn in Figure 1. The first step of the algorithm finds the approximate locations of text using the higher (horizontal) spatial variance of the image intensity on text lines. The output of the first step is a set of bounding boxes around predicted text lines. The second step performs color segmentation of the image. Individual characters are assumed to have uniform color, so that color segmentation followed by connected component analysis segments the image into components, among which character components can be extracted using various heuristics.

The color of the text inside each box is determined as the color whose proportion drops most sharply when the box size is increased 2 times. Using the computed text color and the previous segmentation result, we can find character components inside each box. Since some text components may be cut by box boundaries, we extend text components inside a box to the component in the entire image. If the new component does not extend far from the box, it is classified as a text component. The final post-processing step restores characters that are left out of the bounding boxes. For each component in the entire image that has a proper size, we check if its row-centroid is between the upper and the lower boundaries of some bounding box in which the text color is the same as the color of the component. If it is, we add this component to the text in the box by increasing the box size appropriately.

The method described above can be used to locate horizontally aligned text only. In principle, it can be extended to find text in any orientation, provided that the text is aligned between two parallel lines. Such a generalized algorithm, however, is expected to give more spurious non-text regions.

2.1 Locating Text by Spatial Variance

The first step of the algorithm depicted in Figure 1 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 white regions between the text lines. Moreover, there are steep edges corresponding to the minimal and maximal row coordinates of small letters, such as 'a', 'o', and 'm'. 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 line. In principle, any edge detector can be used to locate the edges. In our exper-

iments, we have used Canny edge detector. Since we are currently interested only in horizontal text lines, we select those edges which have horizontal direction ($\pm$ a small constant).

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 which have a large vertical spread are not good candidates for upper and lower boundaries of a text line, and can be removed. Among the remaining lines, we find the ones which have almost the same row coordinates, and merge them into a single line. The merged edges, of course, must have the same orientations.

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:

1. A significant portion of the two lines must overlap when they are projected vertically.
2. A rising edge must be above the falling edge.
3. There must be no other lines between the two edges in a pair.
4. 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 lines, it is a simple matter to construct the smallest rectangle which 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.

2.2 Locating Text Components Using Color Segmentation

The second step of the algorithm segments an input image into components with uniform color. If we require that the color of an individual component be slowly varying, with sharp edges at its boundaries, then the basic algorithm for finding connected components can be applied. In that case, two neighboring pixels are in the same component if the difference between their colors is less than a threshold. This simple algorithm, however, does not work with most natural images where the characters have blurred boundaries, and are, therefore, connected with the background. A more discriminative method is to divide the image into components of constant color. The image segmentation can now be done by taking one color (a triple of the R-G-B values), and finding all connected components with this color.

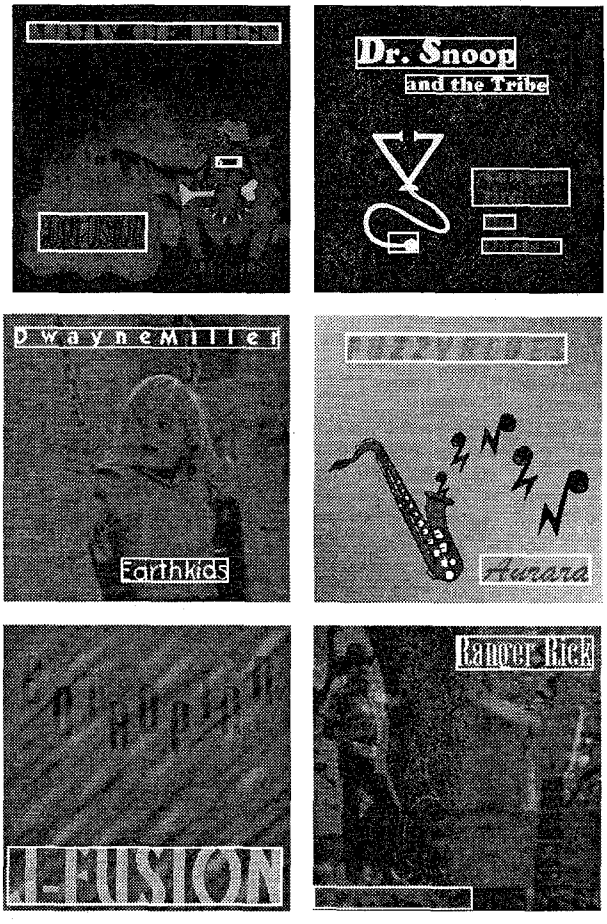

Figure 2: Locating text in CD cover images.

Due to the large number of possible colors, we first quantize the color space into a few prototype colors. The prototypes are found as local maxima in a smoothed color histogram of the input image, all other colors are then assigned to the nearest prototype. In most images used in our experiments, this results in about 5 – 500 peaks in the histogram, and the components that seem different to the human eye are quantized into different classes. Image segmentation can then be done with the basic connected component algorithm.

3 Experimental Results

The proposed method was tested on CD cover images, book cover images and video frames. Image sizes varied from 256×256 for video frames to 700×500 for book covers. Figure 2 shows input CD cover images together with the computed bounding boxes around text regions. It can be seen that the text which has a uniform color can be located accurately. A few spuri-

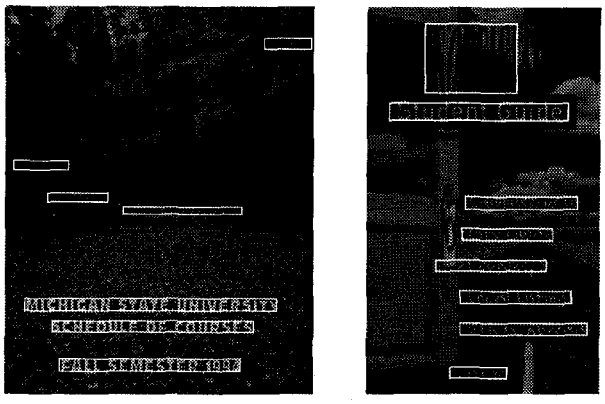

Figure 3: Locating text on book covers.

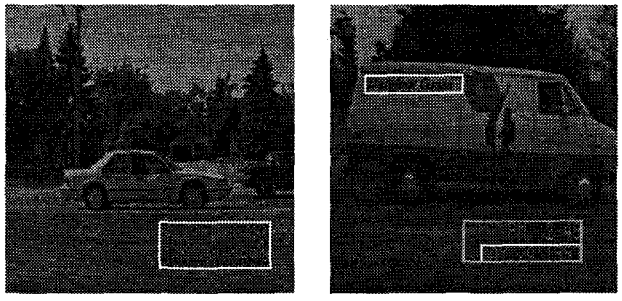

Figure 4: Locating text in video frames of traffic scenes.

ous boxes produce some non-text components, which can be easily removed by an OCR system. Figures 3 shows two images of book covers and the results of locating text in these image. Generally, the text components have been located correctly, except some text with very low contrast. Finally, Figure 4 shows images captured with a video camera. Again, bounding boxes are put around text located in these images.

The algorithm for locating text is relatively fast. The spatial variance method, in addition to performing edge detection, requires no more than two scans over the image to compute variances, and the same number of scans to perform the connected component analysis of the edge pixels. Color segmentation requires one scan over the whole image to construct the color histogram, and two other scans to find connected components. Additional computations for selecting and restoring candidate components depend on how effective data structures can be designed, so that each component is compared with only its nearby neighbors. On a typical 256 × 256 image, the entire processing takes approximately 10 seconds on a Sun Sparc station 20.

4 Conclusion

A method for extracting text in a color image is described. The approach is based on certain heuristics, and the algorithm can, therefore, not be expected to find text in all possible situations. The most serious shortcomings of our algorithm are in failing to locate text which is not well separated from the background (this means that the color of the text and background is similar, or that the contrast of the text is low), and characters which are not aligned, or which have differently colored components. Nevertheless, the method gives good results on a variety of test images. The algorithm is relatively robust to variations in font, color, and size of the text.

Acknowledgement

We want to thank IBM Almaden Research Center for providing us the CD cover images.

References

[1] 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.

[2] 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.

[3] 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.

[4] A.K. Jain and Y. Zhong, Page Layout Segmentation based on Texture Analysis. Under review.

[5] B.M. Mehtre, M.S. Kankanhalli, A.D. Narasimhalu, G.C. Man, Color Matching for Image Retrieval. *Pattern Recognition Letters,* Vol. 16, pp. 325-331, March 1995.

[6] J. Ohya, A. Shio and S. Akamatsu, Recognizing Characters in Scene Images. *IEEE Trans. on PAMI,* Vol 16, No. 2, pp. 214-224, February 1994.

[7] T. Pavlidis and J. Zhou, Page Segmentation and Classification. *Comput. Vision Graphics Image Process.,* 54(6): 484-496, Nov 1992.