Text Localization in WWW Images

Oleg G. OKUN and Matti K. PIETIKÄINEN
Machine Vision and Intelligent Systems Group
Infotech Oulu and Department of Electrical Engineering
P.O.Box 4500, FIN-90014 University of Oulu, Finland

ABSTRACT

The WWW pages often contain text embedded in pictures. The internet search engines as a rule are not capable of finding it, although this kind of text can be meaningful for a user. Text localization in WWW images is therefore of great importance, but this task is not easy to solve, because characters can have arbitrary size, orientation and color, and they can be superimposed on complex, non-uniform background. In this paper, we consider a simple method based on edge detection for text localization in WWW images. Edge detection is done either in an indexed color image or in a grayscale image. A feature employing the edge density and magnitude, computed for small non-overlapping blocks of nxn pixels, is utilized for block classification into text or non-text. Experimental results obtained on a number of test images containing text of different scripts (Latin, Arabic, Chinese) are promising, encouraging further research in this direction.

Keywords: Text Localization, Edge Detection, WWW Images

1. INTRODUCTION

Huge amount of information is nowadays available on the Web. Various search engines are used for information retrieval, but these programs are mostly intended to search for plain text. However, more and more text is embedded in pictures and such text is not detectable by search engines designed for plain text. Hence, localization and extraction of text from WWW images is of great importance, because it would make the search engines more intelligent and user-friendly.

The images placed on the Web are mainly colored. In color document image analysis, previous works have concentrated on text extraction from video images and page images of technical journals [1, 2, 3, 4, 5, 9, 10, 11, 12, 13, 15]. The literature on text localization or extraction from WWW images is not abundant [6, 8, 16], because interest on this subject arose quite recently.

The WWW images have much in common with the color page and video images, but there are also several distinctions. Firstly, the font style, size, and orientation of characters are more arbitrary: for example, Gothic or other stylized characters or curved text lines are not uncommon. Secondly, color may change within a word or even within a single character, though often this change is rather gradual than abrupt. Thirdly, text is often rendered at a low spatial resolution (72 dpi). Finally, the background may be very complex, which brings additional problems to the text localization task.

Because of the complex nature of color images, independently of their type, many existing methods use a number of thresholds, which are sometimes difficult to tune. The purpose of this paper is twofold. Firstly, we review problems and previous approaches concerning text localization in WWW images. Secondly, we investigate how well text can be localized by a method based on simple edge operators. We understand, however, that text localization relying only on such low-level features will not solve the whole problem. Our approach was inspired by the promising results obtained for complex images of advertisements [7].

This paper has the following structure. Section 2 briefly reviews previous work in the field. Section 3 introduces our main idea, while its implementation is presented in Section 4. Experiments are described in Section 5. Section 6 concludes the paper.

2. PREVIOUS WORK IN THE FIELD

In [6], the initial colors are first reduced by dividing values of R, G, and B elements by 64, followed by spatial merging consisting of detecting connected components with the same color. The found connected components are then divided into major and minor, depending on their population. A component whose population is greater than a ($a=7$) is assumed to be a major one. Minor components are merged to the nearest major component if the color difference between them is below a threshold. Major components with the population more than $3\%$ ($\beta=10$) of the total number of pixels in the image are classified as background and discarded.

After that, strokes are extracted by assuming that characters are composed of strokes, independently of a language or script. Two stroke-based features are employed: 1) a stroke has a relatively fixed width, and 2) the ratio of stroke width to its length is bounded. These features are computed during thinning of the found major components. As a result, the strokes are represented by labeled skeletons.

The obtained strokes are further merged into strings of characters by using the potential field approach, where each stroke is considered to be an object having a potential field proportional its size. Two adjacent potential fields are merged if their attraction forces are larger than a predefined threshold. The potential field approach allows one to detect curved text lines, but only when the distance between adjacent lines is sufficiently larger than that between characters within a line.

Schmidt et al. [8] try to localize text in compressed im-
ages. They argue that the compressed images with text have different size than those without text. The main idea is that adding text to a simple, homogeneous image increases its complexity, which can be measured by the size of a JPEG-compressed image. Authors use a simple heuristic formula linking the area occupied by text and the increase of the image size. However, this method seems to be successful only if the text is superimposed on a uniform background and it may be necessary to know the font size in advance.

The method in [16] is intended to analyze GIF images. It uses color quantization based on the Euclidean minimum spanning tree (EMST), where the number of nodes of a graph is no more than 256, because GIF does not support more than 256 colors. The EMST is a graph with colors as nodes. The distance (weight of an edge linking two nodes) between two colors is computed as the Euclidean distance of their R, G, and B elements. Once the EMST is constructed, the average distance is determined, and the edges with the weights larger than the average are deleted from the EMST. As a result, the EMST is divided into several disjoint sub-trees, each of which corresponds to a separate color cluster. For each cluster, the most frequently occurring color is selected as a representative of the cluster.

The connected components belonging to each cluster are then detected. Components are classified into text-like or non-text-like on the basis of shape characteristics such as size, number of holes and branches, stroke width. Finally, projection profiles of bounding boxes eliminate spurious components. This method can only extract horizontally aligned text.

By analyzing the methods listed, we can conclude that either they are designed for a specific image format or/and they require a lot of parameters to be tuned.

In the next section, we explain our approach aiming at reducing the number of parameters and heuristics used.

3. KEY PRINCIPLES OF OUR APPROACH

As mentioned above, the WWW images are usually color. They are typically stored in one of the two graphical file formats: JPEG or GIF. A GIF image can have no more than 256 unique colors and therefore the analysis of such an image is simpler than a JPEG image which may comprise up to 16 million colors. Since there are only two formats, it is natural to derive a common image representation from them for convenience of further analysis.

Although some authors argue that 256 colors are often insufficient for the WWW images, but in fact, one of the first steps of almost every method is to reduce the number of original colors, because many perceptually similar colors can be replaced with one color.

Therefore we limit the number of colors to 256, and it means that the original color image with 24 bits per pixel is transformed into another image with 8 bits per pixel by using a color quantization technique. The problem of color quantization is to design a look-up table (called also color map or palette) and to determine how each index of this table has to be mapped to the RGB colors. This color map is often created by analyzing the color histogram of a given image.

The 8-bit image is called indexed and it is still color, but its image and color data are separated from each other, that is, this image (suppose that it is of M x N pixels) consists of an M x N index array and an M x N color map. Each row of the color map determines the B, G, and R components of a single color. Each pixel of the index array is an index into the color map. For example, the index 5 at the position (i,j) points to the fifth row of the color map, where the color of the pixel (i,j) is stored. Since we restrict ourselves to 8-bit images, it means that m is equal to 256 and indices range from 0 to 255.

We prefer to process the 8-bit indexed images rather than the RGB ones, because we can manipulate them in the similar way as grayscale images by considering the indices as a kind of pixel values (though we remember that they are not true pixel values).

An alternative (or complementary) way to cope with color is to transform a color image from one color space (say, RGB) into another one. The resulting color space should separate well luminance and chrominance information so that we can select one color plane representing the luminance and work with it as with a grayscale image.

The edges detection is applied to the image obtained in either way, and the gradient magnitude of each edge (edge pixel) is determined. The image is partitioned into small non-overlapping blocks of n x n pixels, and an edge-based feature related to the density of edges and their magnitudes within each block is computed. By doing this, we suggest that the edges in the background are generally weaker than those of the characters so that an edge detector is able to suppress (at least, partly) the weak edges. As a result, a properly chosen threshold can sufficiently well separate text and non-text blocks. By sufficiently well, we understand that some non-text blocks may also be classified as text.

4. DESCRIPTION OF THE METHOD

The input and output images are color, but the output image is divided into small n x n pixels blocks classified as non-text or as text. The non-text blocks have all pixel values set to zero, while the text blocks are copied as they are from the original image. The value of n depends on the size of the original image.

The proposed algorithm consists of the following steps:

Step 1. Specify m. Convert the original color image either to an indexed image with m colors or to a grayscale image. Denote the resulting image as I.

Step 2. Set a threshold t₁ for suppressing weak edges or let it be automatically computed during the next step. Set a threshold t₂ for block classification into text or non-text. Choose an appropriate n.

Step 3. Apply an edge operator with t₁ to I. As a result, E (edge) and G (gradient magnitude) images are generated.

Step 4. Divide I into n x n pixels blocks. Create a new image B, where each pixel represents the corresponding block of I.

Step 5. For each pixel of B (and therefore for each block of I), compute a feature R by using E and G obtained in Step 3. If R ≥ t₂, a given block is text, otherwise it is non-text.
Now let us consider these steps in detail. As mentioned in the previous section, there are two alternatives to reduce the amount of data in a color image.

With the first of them, a color image is converted to an indexed one by using the minimum variance quantization technique [14]. This method involves partitioning the RGB color space into smaller boxes (not necessarily cubes) of different sizes, depending on how colors are distributed in the original image. We set the number of such boxes to 256 (that is, m=256) by assuming 256 colors.

With the second alternative, we transform the input image from its color space into another color space, where the luminance and chrominance components are separated from each other. We chose the NTSC color space, though other spaces such as HSI could be used as well. The Y-plane of NTSC representing luminance is expressed as follows if the original space was RGB (don’t confuse G and B in Eq. 1 with those used in the algorithm description).

\[ Y = 0.299R + 0.587G + 0.114B. \]  

There are various operators for edge detection. We selected the Sobel operator because of its simplicity and wide use. The Sobel operator is applied to the obtained (either indexed or grayscale) image in Step 1 in order to detect edges, followed by thresholding and non-maximum suppression. To be technically correct, the Sobel operator is not intended for indexed images. But both grayscale and indexed images have the same range of values so that it is interesting to compare edge detection results in both cases.

A pixel is considered to belong to an edge if 1) its gradient magnitude is larger than the magnitudes of its 4-connected neighbors and 2) its gradient magnitude is larger than \( t_1 \), so that weak edges are cut off. The value of \( t_1 \) can be set by a user or automatically computed as the root-mean-square estimate of noise:

\[ t_1 = \sqrt{\frac{1}{4}\sum_{i=1}^{4}\sum_{j=1}^{4}(G_x(i,j) + G_y(i,j))^2}{(h-1)(w-1)}, \]

where \( h \) and \( w \) are the height and width of the image and \( G_x(i,j) \) and \( G_y(i,j) \) are \( x \) and \( y \)-components of the gradient magnitude \( G(i,j) = \sqrt{G_x^2(i,j) + G_y^2(i,j)} \) for a pixel at \( (i,j) \).

After edge detection we have two images: a binary edge image \( E \) (1-edge, 0-non-edge) and a grayscale gradient magnitude image \( G \). Using both images, we classify blocks of the original image into text or non-text. The image \( B \) represents a block partitioning of the original image, where the pixel \((i,j)\) in \( B \) corresponds to the block of pixels between coordinates \( \text{isstart} \) and \( \text{icend} \), \( \text{jstart} \) and \( \text{jend} \), where \( \text{isstart} = (i-1)n + 1 \), \( \text{jend} = jn \) (we assume that \( i \) and \( j \) start from 1). If the sizes of \( I \) are \( w \) pixels, then those of \( B \) are \( (h/n)x(w/n) \) pixels.

For each pixel \((i,j)\) of \( B \) (and therefore each block of \( I \)), a special feature \( \mathcal{R} \) is computed with one of the following equations:

\[ \mathcal{R} = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{H}(G(i,j) - t_1) \mathcal{H}(E(i,j) - 1), \]  

\[ \mathcal{R} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{G(i,j) \mathcal{H}(G(i,j) - t_1) \mathcal{H}(E(i,j) - 1)}{n^2}, \]  

where \( \mathcal{H} \) is the step function:

\[ \mathcal{H}(x-a) = \begin{cases} 1 & \text{if } x \geq a, \\ 0 & \text{if } x < a. \end{cases} \]

\( \mathcal{R} \) defined by Eq. 3 is the number of edge pixels per block. \( \mathcal{R} \) in Eq. 4 is the average gradient magnitude per pixel and \( \mathcal{R} \) in Eq. 5 is the average gradient magnitude per edge pixel. Such definitions of \( \mathcal{R} \) reflect the fact that the number of edge pixels and their gradient magnitudes are higher in many cases for text than for non-text blocks.

Using \( \mathcal{R} \), every block of \( I \) is labeled either as text or as non-text based on the threshold \( t_2 \) set by a user and determined from the range of \( \mathcal{R} \) computed over all blocks. Unfortunately, the choice of an appropriate value is not easy. On the one hand, the larger \( t_2 \) is, the more probable that we can miss some text. On the other hand, a small \( t_2 \) naturally leads to a high false positive rate when a lot of non-text blocks are labeled as text. As a result, we should compromise when selecting \( t_2 \). Currently, we narrow the original interval \([\text{min}, \text{max}] \) (excluding \( \mathcal{R} = 0 \)) to the subinterval \([\text{min}, k(\text{max} - \text{min}) + \text{min}] \), where \( k \in [0.1, 0.3] \) and select \( t_2 \) from it. It means that any value from this subinterval can be used as \( t_2 \).

5. EXPERIMENTS

We collected 40 WWW images (both GIF and JPEG) from the Web for testing of our method. The images contained text in English, Arabic, and Chinese. The background was both uniform and complex.

Finally, we would like to demonstrate that the color-to-indexed conversion does not significantly degrade the visual quality of the image (compare Figs. 1 and 2). These images are printed as grayscale, but their color originals showed almost no observable difference. This indicates that we can use indexed images with 256 colors instead of the corresponding true color images with many more colors.

Figure 1: RGB image with Arabic text

Figure 2: Indexed image obtained from the RGB image in Fig. 1

The second observation concerns a comparison of edge detection in the indexed and grayscale images. The edge images with automatic computation of \( t_1 \) for the indexed (Fig. 2) and grayscale (Fig. 3) images are shown in Figs. 4 and 5, respectively. As one can notice, the edges in the lower text line were detected in Fig. 5, but not in Fig. 4.
Of course, we could remedy this problem by manually setting $t_1$, but this solution is not good, because it involves a manual parameter setting. We conclude that it is better to detect edges in a grayscale image, whereas an indexed image will be useful in color analysis.

Figure 3: Grayscale image obtained from the RGB image in Fig. 1

Figure 4: Edge image for the image in Fig. 2

Figure 5: Edge image for the image in Fig. 3

Text localization when the grayscale image in Fig. 3 is used as an input, is shown in Fig. 6. In Fig. 6 and other figures below, displaying text localization results, black color corresponds to non-text blocks, while various gray tints point to text blocks copied as they are from the original image.

Figure 6: Text localization for the image in Fig. 3 ($\mathcal{R}$ was computed by Eq. 5, $n=12$, $t_1=0.282$, $t_2=0.085$)

Another text image containing English text is given in Fig. 7. Unlike Fig. 3, its background is non-uniform (for example, sequences of zeros and ones of a varying contrast are seen, which do not carry valuable information). There is also a picture of a city near to the bottom of the image. Text localization results are shown in Fig. 8. Almost all text was correctly identified, though small fragments of non-text were extracted as well.

An image containing Chinese characters overlaid on a picture is shown in Fig. 9. The characters are visually detectable from the background, but the difference in gray levels was not very high so that some edges of the characters were missing. This resulted to missing of small fragments of the characters in Fig. 10. On the other hand, a significant number of non-text blocks were classified as text, because they contained enough edges whose magnitude was large.

The last two examples demonstrate the performance of our method in the case of large-sized characters (see Figs. 11 and 12). Because the thickness of the character strokes in Fig. 11 was larger than $n$ set by us, parts of the characters were not extracted in Fig. 12. In Fig. 14, characters did not have very thick strokes, and the large-sized text was successfully localized.

Figure 7: Grayscale image with English text

When testing our method, we did experiments with all three ways of computing $\mathcal{R}$. The values of $\mathcal{R}$ computed by Eqs. 4 and 5 led to slightly better results than those determined by Eq. 3. This is not surprising because Eqs. 4 and 5 employ gradient magnitudes, while Eq. 3 only exploits the edge density. However, in general, it is difficult to say about the superiority of one definition of $\mathcal{R}$ over the others.

6. CONCLUSION

Text localization in WWW images is very important for information retrieval from the Web, because this kind of text is often related to the content of a Web page. Various methods for text localization have been proposed, but they rely on many heuristics and therefore require many parameters to be tuned. In this paper, we investigated the usefulness of a simple edge-based method for this task.

Our method requires to set or to tune three parameters: $n$, $t_1$, $t_2$. Currently, we compute $t_1$ automatically so that its tuning is not a problem. The value of $n$ depends on the image size and it is well bounded. The third parameter, $t_2$, mostly determines the final results. Unfortunately, we have no precise procedure how to automatically select its best value. A more intelligent method utilizing higher-level information is definitely needed to accomplish the task.
because the low-level features, alone, are not sufficient.

Despite this fact, our method showed encouraging results. It has a number of advantages, among them are: 1) independence of the image file format (both GIF and JPEG images can be processed), 2) good tolerance to texts of various scripts and languages, 3) small number of parameters to be predefined, and 4) moderate computational cost.

In the future, we plan to use complementary color information to rectify the text localization results.

ACKNOWLEDGMENT

This work has been partly supported by the Academy of Finland. We are also thankful to Mr. Topi Mäenpää for his comments.

References


Figure 10: Text localization for the image in Fig. 9 (was computed by Eq. 4, $n=4$, $t_1=0.157$, $t_2=0.004$)

Figure 11: Grayscale image with large characters

Figure 12: Text localization for the image in Fig. 11 (was computed by Eq. 5, $n=12$, $t_1=0.225$, $t_2=0.008$)

Figure 13: Grayscale image with large characters

Figure 14: Text localization for the image in Fig. 13 (was computed by Eq. 3, $n=12$, $t_1=0.204$, $t_2=6.323$)


