FINDING TEXT IN IMAGES VIA LOCAL THRESHOLDING

Julinda Gllavata[1], Ralph Ewerth[1] and Bernd Freisleben[1,2]

[1]SFB/FK 615, University of Siegen, D-57068 Siegen, Germany
[2]Dept. of Math. and Computer Science, University of Marburg, D-35032 Marburg, Germany
{juli, ewerth, freisleb}@informatik.uni-marburg.de

ABSTRACT

Texts appearing in images or videos are not only important sources of information but also significant entities for indexing and retrieval purposes. However, since text is often printed against a shaded or textured background it is often difficult to recognize. In this paper, an approach which can automatically detect, localize and extract horizontally aligned text with different sizes, fonts and languages in images is presented. The basic idea of our approach is to apply an appropriate local thresholding technique to sequences of line histogram differences in order to increase the robustness of text detection with respect to complex backgrounds. The performance of our approach is demonstrated by presenting experimental results for a set of images taken from video sequences.

1. INTRODUCTION

Text in images or videos can help to identify the image information (e.g. somebody's name appearing in an image) or to display information which is independent of the image (e.g. important news during the transmission of a movie). In general, text appearing in images can be classified into two groups: *scene text* and *artificial text* [6]. Scene text is part of the image, and appears accidentally, like in traffic signs etc. whereas artificial text is produced separately from the image and is laid over it in a later stage, such as the name of a journalist during a news program. Artificial text is usually a very good key to index image or video databases. Thus, the main purpose of automatic text localization methods is the detection of artificial text information in images and videos.

However, text localization in complex images is an intricate process due to different backgrounds or different fonts, colors, sizes, and alignment orientations of texts appearing in them. In order to be successfully recognizable by an OCR system, an image having text must fulfill certain conditions like a monochrome text and background where the background-to-text contrast should be high.

In this paper, we present an approach to detect, localize, and extract texts appearing in images and make them ready for a subsequent OCR process. Our proposal is an extension of an algorithm that we proposed in a previous paper [3]. Essentially, this algorithm works as follows: Color images are first converted to grayscale images. Then, an edge image is generated and horizontal line histograms considering the number of high-contrast edges per line are computed. Finally, a global threshold is used to decide whether a certain number of edges represents text or not. However, it is clear that such a threshold limits the detection performance since the number of such edges per line strongly depends on the background complexity.

Consequently, the basic idea of the proposal made in this paper is to apply a local threshold. We have observed some similarity between the problem of detecting text in a down-sampled difference sequence of line histograms and the problem of detecting cuts in videos using frame histogram differences. Thus, we suggest to use an adapted version of a local thresholding technique that has been successfully used in a well known video cut detection algorithm [8]. The performance improvements resulting from the use of local thresholding are illustrated by presenting experimental results for a set of images.

The paper is organized as follows. Section 2 gives a brief overview of related work in the field. Section 3 presents the individual steps of our approach to text detection. Section 4 shows the experimental results obtained for a set of images. Section 5 concludes the paper and outlines areas for future research.

2. RELATED WORK

Several approaches for text localization in images and videos have been proposed in the past. Based on the methods being used, these approaches can be categorized into two main classes: *connected component based methods* and *texture based methods*.

The first class of methods [1], [2], [4], [6] employs connected component analysis, which consists of analyzing the geometrical arrangement of edges or homogeneous color and grayscale components that belong to characters. For example, Lienhart and Effelsberg [6] have proposed an approach which operates directly on

color images using the RGB color space. The character features like monochromacity and contrast with the local environment are used to qualify a pixel as a part of a connected component or not, segmenting each frame into suitable objects in this way. Then, regions are merged using criteria of having similar color. Finally, specific ranges of width, height, width-to-height ratio and compactness of characters are used to filter out character regions.

Garcia and Apostolidis [2] perform an eight-connected component analysis on a binary image, which is obtained as the union of local edge maps that are produced by applying the band Deriche filter to each color.

Jain and Yu [4] first perform a color reduction by bit dropping and color clustering quantization, and afterwards a multi-value image decomposition algorithm is applied to decompose the input image into multiple foreground and background images. Then, connected component analysis is performed on each of them to localize text candidates. This method extracts only horizontal texts of large sizes.

Agnihotri and Dimitrova [1] have presented an algorithm which operates directly on the red frame of the RGB color, with the aim to obtain high contrast edges for the frequent text colors. By means of the convolution process with different masks, they first enhance the image and then detect edges. This edge image is further processed by grouping neighboring edge pixels to single connected components structures. Finally, the candidates undergo another treatment in order to be ready for OCR.

The second class of approaches [5], [7] regards texts as regions with distinct textural properties, such as character components that contrast the background and at the same time exhibit a periodic horizontal intensity variation due to the horizontal alignment of characters. Methods of texture analysis like Gabor filtering and spatial variance are used to automatically locate text regions. Such approaches do not perform well with different character font sizes, and furthermore they are computationally intensive.

For example, Li and Doerman [5] typically use a small window of 16x16 pixels to scan the image and classify it as text or non-text using a three-layer neural network. For a successful detection of various text sizes they use a three-level pyramid approach. A projection profile is used to extract text elements from text blocks.

Wu et al. [7] have proposed an automatic text extraction system which at first uses distinctive characteristics of texts, such as for instance the fact that text possesses certain frequency and orientation information or that text shows spatial cohesion (characters of the same text string are of similar heights, orientation and spacing) to identify the possible text regions in an image. As a second step, bottom-up methods are applied to extract connected components. A simple histogram-based algorithm is proposed to automatically find the threshold value for each text region, making the text cleaning process more efficient.

3. PROPOSED TEXT DETECTION AND SEGMENTATION METHOD

In this section, we present the four basic processing steps of the text detection and segmentation method described in [3] in more detail and motivate the new idea of adapting a local thresholding technique, which is introduced in step 3. The proposed system design is based on the following assumptions: (a) only texts with a horizontal alignment can be detected, and (b) texts that are smaller than a certain (small) font size will not be detected. Our approach to text detection in images consists of the following steps:

Step 1: Image preprocessing.

If the image data is not represented in YUV color space, it is converted to YUV color space by means of an appropriate transformation. In contrast to the systems presented in [1], [2], [6] we use only the luminance data (Y channel of YUV) during further processing since it is sufficient to differentiate between the possible text regions and the rest of the image.

Step 2: Edge detection and horizontal projection.

The pixels representing the text contours usually have a high contrast to their neighbor pixels. Thus, an edge image is generated where the value of each pixel is equal to the biggest of the differences between it and its three neighbors (the left, the upper and the left upper neighbor pixel). This process is followed by removing all the pixels whose contrast with their local background is not sufficiently high. As a result, most of the text edges as well as some non-text edges are included in the edge image. Next, the technique of horizontal projection analysis is applied directly on the edge image. The histogram $H(I)$ is computed, where $H(I)$ is the number of pixels in line I of the edge image exceeding a given value.

Step 3: Text detection via local thresholding.

In order to reduce noise effects the histogram values that were produced in step 3 undergo a down-sampling process. As the result of this process, the beginning and the end of a text line will cause very high peaks on the down-sampled histogram. Additionally, down-sampling allows to deal with anti-aliased texts as well as with texts that are not aligned perfectly horizontally. The down-sampling process is included in the calculation of the two histogram difference sequences D and D':

$$D(H, I/N) = H(I+N) - H(I), \quad \text{where } I \text{ is the line number and, for all } I\text{: } I \bmod N = 0.$$

$$D'(H, I/N) = \mathrm{ABS}(D(H, I/N)).$$

The idea is to detect large single or double peaks in D' (an example for a sequence D' can be seen in figure 1) that represent either the beginning or the end of a text line.

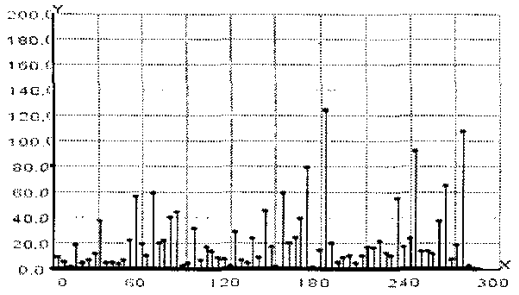

Figure 1: Example for the sequence of absolute histogram differences for an image from the test set.

We assume that in case of a text line beginning the sign of D(H, I/N) is positive since the number of edges increases when a text appears, and D(H, I/N) is negative in case of text line end. A sliding window technique is used in order to consider local image properties, such as the variation of background in case of text detection. For each text candidate at line N*J a sliding window of width 2M+1 (D'(H, J-M), D'(H, J-M+1),..., D'(H, J+M)) is considered, where M is a positive integer. The value D'(H, J) within a sliding window of size M is accepted as the beginning of a text line, if the following conditions are satisfied:

1. D' (H, J) IS THE MAXIMUM WITHIN THE WINDOW, AND EITHER
   2. D'(H, J) > R * D'(H, S), WHERE D'(H, S) IS THE 2ND LARGEST VALUE IN THE WINDOW, AND R > 0,
   OR
   3. THE 2ND LARGEST VALUE D'(H, S) IS A NEIGHBOR OF D'(H, J), AND D'(H, S) > R * D'(H, T), WHERE D'(H, T) IS THE 3RD LARGEST VALUE AND SIGN(D(H, J)) = SIGN(D(H, S)) , AND R > 0,
4. D(H, J) > 0.

The end of a text line is considered to be found if the same conditions are fulfilled except for the last one, which is replaced by: D(H, J) < 0. The third condition is incorporated to deal with anti-aliased texts as well as with text not perfectly aligned horizontally. To discard some possible false alarms, each of candidates undergoes a geometrical analysis. It should be remarked that the parameters M and N limit the detectable text height to a minimum of 2*M*N lines. As mentioned above, this principle has been proven to be very successful for cut detection purposes [8] where a cut is represented by a single large peak in a sequence of image histogram differences. We use this technique since we have observed a similar data pattern in case of text detection.

Step 4: Text segmentation.

The remaining text regions are processed in this step, preparing them to be fed to an OCR. First, a gap filling process takes place. If there is a gap between two pixels in horizontal, vertical or diagonal direction in the binary edge image, then this gap is filled with the color of its neighbor pixels. In the second step, this gap image is used to simplify the background of the text candidate regions which have been extracted from the gray image. The output of this step is a so called text image, where the detected text appears on a simplified background.

4. EXPERIMENTAL RESULTS

We have tested the proposed approach performing a series of experiments on a test set, which consists of 175 images of various types both with and without text. The first part consists of frames extracted from MPEG-1 video sequences which were kindly provided to us by Lienhart [6]. These sequences mainly consist of the credits title sequences at the end of a movie with a lot of text lines scrolling downwards. We have extracted a frame each second if either the background or the text being displayed changed significantly within that second. Otherwise, we have checked the next frame one second later. The second part of images covers a wider range of image types and complexities since we have used images from our image database. This database was created during a former media research project on the presentation of politicians in selected TV evening news broadcasts between 1950 and 2000. Since the quality of these images is quite poor, the conditions for text localization are clearly more challenging. Third, commercials images have been used. All images have been selected due to background complexity and have a resolution of 384*288 pixels. Overall, the 175 images contained 784 text lines.

During the experiments, the following parameter values were used: N=4, M=1, R=2. The choice of N and M was made in order to detect texts of even small size (the minimum detectable text height is 8 lines), while parameter value R is based on experiences in cut detection experiments. We have checked the output for each image and measured the number of correctly detected text lines as well as the number of falsely detected text lines. A detected text line was accepted as correct if from our point of view the probability is high that an OCR system could recognize the text characters. Examples of the detection results are presented in figure 2 and 3 for an image from our test set for both with and without local thresholding. It is evident from figure 3 that with the local thresholding technique the first three text lines are located correctly, while in case of not using a local threshold (see figure 2), the upper image part has been marked as text which should be considered as false alarm. The corresponding down-sampled sequence of absolute histogram differences for this image can be seen in figure 1. The results of the experiments are presented in table 1 where recall and precision are listed for both the basic implementation and

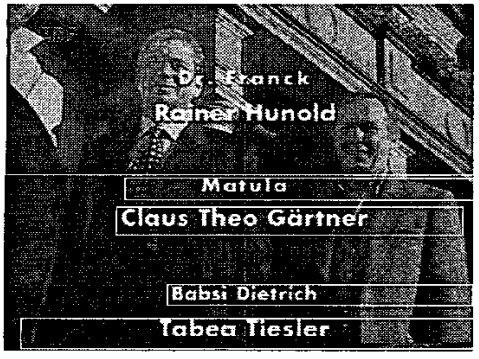

Figure 2: The detection result without the local thresholding technique.

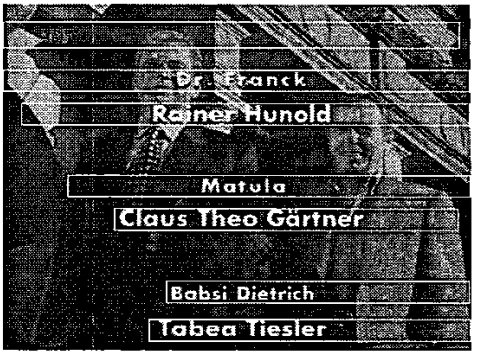

Figure 3: The detection result with the local thresholding technique.

Implementation	Basic impl.	Local threshold
Recall	71.93 %	84.94 %
Precision	84.30 %	85.62 %

Table 1: The experimental results for the test set for the basic implementation and its variant with the local thresholding extension.

its extended version with local thresholding. Recall is defined as the number of correctly detected text lines divided by the number of all text lines, whereas precision is defined as the number of correctly detected text lines divided by the number of all detected text lines, including false alarms. As can be seen in table 1, the proposed technique leads to a noticeable increase of over 12% in the number of detected text lines while the precision increases slightly. However, a fair comparison with other relevant approaches is difficult since researchers have not used a standard image test set so far.

5. CONCLUSIONS

We have proposed an enhancement of an algorithm to localize and extract text from images reported in a previous paper [3]. The algorithm has been extended with a local thresholding technique that led to a noticeable improvement in detection performance. This technique has been adapted from the domain of video cut detection since we observed a similar data pattern analyzing our text detection algorithm. The performance of our approach has been demonstrated by presenting experimental results for a test set consisting of 175 images.

There are several areas for future work. First, the possibilities of adaptive parameter estimation for R, M and N will be investigated. Second, the approach should be extended to also work for video sequences instead of images, considering motion information in compressed video format. Finally, we plan to extend our system with an OCR engine to check the recognition rate of our proposed text detection and segmentation algorithm.

6. ACKNOWLEDGEMENT

This work is financially supported by the Deutsche Forschungsgemeinschaft (SFB/FK 615, Teilprojekt MT). The authors would like to thank M. Gollnick, M. Grauer, F. Mansouri, E. Papalilo, R. Sennert and J. Wagner for their valuable support.

7. REFERENCES

[1] L. Agnihotri and N. Dimitrova, "Text Detection for Video Analysis", in *Proc. of the Int'l Conference on Multimedia Computing and Systems*, pp. 109-113, Florence, 1999.

[2] C. Garcia and X. Apostolidis, "Text Detection and Segmentation in Complex Color Images", in *Proc. of International Conference on Acoustics, Speech and Signal Processing* (ICASSP2000), Istanbul, Vol. IV, pp. 2326-2330, 2000.

[3] J. Gllavata, R. Ewerth and B. Freisleben, "A Robust Algorithm for Text Detection in Images", in *Proc.of 3rd Int'l Symposium on Image and Signal Processing and Analysis* (ISPA '03), pp. 611-616, Rome, 2003.

[4] A. K. Jain and B. Yu, "Automatic Text Location in Images and Video Frames", in *Proc.of International Conference of Pattern Recognition*, pp. 1497-1499, 1998.

[5] L. H. Li and D. Doermann, "Automatic Text Tracking In Digital Videos", in *Proc. of IEEE 1998 Workshop on Multimedia Signal Processing*, Redondo Beach, CA, USA, pp. 21-26, 1998.

[6] R. Lienhart and W. Effelsberg, "Automatic Text Segmentation and Text Recognition for Video Indexing", in *Multimedia Systems*, Vol 8, pp. 69-81, 2000.

[7] V. Wu, R. Manmatha and E. M. Riseman, "Finding Text in Images", in *Proc. of Second ACM International Conference on Digital Libraries*, Philadelphia, PA, pp. 23-26, July 1997.

[8] B. Yeo and B. Liu, "Rapid Scene Analysis on Compressed Video", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, No. 6, pp. 533-544, 1995.