A Robust Algorithm for Text Detection in Images

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. & Computer Science, University of Marburg, D-35032 Marburg, Germany
{juli, ewerth, freisleb}@informatik.uni-marburg.de

Abstract

Text detection in images or videos is an important step to achieve multimedia content retrieval. In this paper, an efficient algorithm which can automatically detect, localize and extract horizontally aligned text in images (and digital videos) with complex backgrounds is presented. The proposed approach is based on the application of a color reduction technique, a method for edge detection, and the localization of text regions using projection profile analyses and geometrical properties. The output of the algorithm are text boxes with a simplified background, ready to be fed into an OCR engine for subsequent character recognition. Our proposal is robust with respect to different font sizes, font colors, languages and background complexities. The performance of the approach is demonstrated by presenting promising experimental results for a set of images taken from different types of video sequences.

1. Introduction

Indexing images or videos requires information about their content. This content is often strongly related to the textual information appearing in them, which can be divided into two groups:

- Text appearing accidentally in an image that usually does not represent anything important related to the content of the image. Such texts are referred to as *scene text* [8].
- Text produced separately from the image is in general a very good key to understand the image. In [8] it is called *artificial text*.

In contrast to scene text, artificial text is not only an important source of information but also a significant entity for indexing and retrieval purposes. Localization of text and simplification of the background in images is the main objective of automatic text detection approaches. However, text localization in complex images is an intricate process due to the often bad quality of images, different backgrounds or different fonts, colors, sizes of texts appearing in them. In order to be successfully recognizable by an OCR system, an image having text must fulfill certain requirements, like a monochrome text and background where the background-to-text contrast should be high.

In this paper, we present an approach that allows to detect, localize and extract texts from color images with complex backgrounds. The approach is targeted towards being robust with respect to different kinds of text appearances, including font size, color and language. To achieve this aim, the main focus of the proposed algorithm is centered on the recognition of the specific edge characteristics of characters. Based on the way how possible text areas are detected and localized, our method can be classified as a connected-component based approach. It essentially works as follows: Color images are first converted to grayscale images. An edge image is generated using a contrast segmentation algorithm, which in turn uses the contrast of the character contour pixels to their neighboring pixels. This is followed by the analysis of the horizontal projection of the edge image in order to localize the possible text areas. After applying several heuristics to enhance the resulting image created in the previous step, an output image is generated that shows the text appearing in the input image with a simplified background. These images are ready to be passed to an OCR system. The software is completely written in JAVA to be able to easily run the code in parallel on possibly heterogeneous networked computing platforms. The performance of our approach is illustrated by presenting experimental results for different sets of images.

The paper is organized as follows. Section 2 gives an overview of related work in the field. Section 3 presents the individual steps of our approach to text localization. Section 4 contains 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 detection in images and videos have been proposed in the past. Based on the methods being used to localize text regions, these approaches can be categorized into two main classes: *connected component based methods* and *texture based methods*.

The first class of approaches [1, 2, 4, 5, 6, 8] 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, Cai et al.[2] have presented a text detection approach which is based on character features like edge strength, edge density and horizontal distribution. First, they apply a color edge detection algorithm in YUV color space and filter out non-text edges using a low threshold. Then, a local thresholding technique is employed in order to keep low-contrast text and simplify the background. Finally, projection profiles are analyzed to localize text regions.

Lienhart and Effelsberg [8] have proposed an approach which operates directly on color images using the RGB color space. The character features like monochromacity and contrast within 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 the criteria of having similar color. At the end, specific ranges of width, height, width-to-height ratio and compactness of characters are used to discard all non-character regions.

Kim [6] has proposed an approach in which LCQ (Local Color Quantization) is performed for each color separately. Each color is assumed as a text color without knowing whether it is real text color or not. To reduce processing time, an input image is converted to a 256-color image before color quantization takes place. To find candidate text lines, the connected components that are extracted for each color are merged when they show text region features. The drawback of this method is the high processing time since LCQ is executed for each color.

Agnihotri and Dimitrova [1] have presented an algorithm which uses only the red part of the RGB color space, with the aim to obtain high contrast edges for the frequent text colors. By means of a convolution process with specific masks they first enhance the image and then detect edges. Non-text areas are removed using a preset fixed threshold. Finally, a connected component analysis (eight-pixel neighborhood) is performed on the edge image in order to group neighbouring edge pixels to single connected components structures. Then, the detected text candidates undergo another treatment in order to be ready for an OCR.

Garcia and Apostolidis [4] perform an eight-connected component analysis on a binary image, which is obtained as the union of local edged maps that are produced by applying the band Deriche filter on each color.

Jain and Yu [5] 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 combined with projection profile features are performed on each of them to localize text candidates. This method can extract only horizontal texts of large sizes.

The second class of approaches [7, 9] 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 [7] typically use a small window of 16x16 pixels to scan the image and classify each of them as a text or non-text window using a three-layer neural network. For a successful detection of various text sizes, they use a three-level pyramid approach. Text regions are extracted at each level and then extrapolated at the original scale. The bounding box of the text area is generated by a connected component analysis of the text windows.

Wu et al. [9] have proposed an automatic text extraction system, where second order derivatives of Gaussian filters followed by several non-linear transformations are used for a texture segmentation process. Then, features are computed to form a feature vector for each pixel from the filtered images in order to classify them into text or non-text pixels. In 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. The Proposed Text Localization Method

In this section, the processing steps of the proposed text localization approach are presented. Our intention is to build an automatic text localization and extraction system which is able to accept different types of still images (or video frames) possibly with a complex background. The system design is based on the following assumptions: (a) the input to our system can be a grayscale or a color image; (b) the current version can only detect texts with a horizontal alignment, and (c) texts that are smaller than a certain (small) font size will not be detected.

In contrast to many other text detection approaches, our complete implementation has been written in the JAVA programming language, which allows the code to be easily distributed and run in parallel on heterogenous platforms connected via the Internet. This allows to treat text localization as a scalable compute-intensive application of the *Grid computing* paradigm [3]. The different steps of our approach are as follows.

Step 1: Image Preprocessing. If the image data is not represented in YUV color space, it is converted to this color

Algorithm 3.1.
Image generateEdgeImage(Image $grayImg$)

comment: Create an XxY output image $edgeImg$
comment: $grayImg$ is the XxY result image created in step 1
$x \leftarrow 0$
$y \leftarrow 0$
$left \leftarrow 0$
$upper \leftarrow 0$
$rightUpper \leftarrow 0$
for all $pixel_{x,y} \in grayImg$ **do**
  if $(0 < x < X - 1)$ **and** $(0 < y < Y)$ **then**
    $left \leftarrow |pixel_{x,y} - pixel_{x-1,y}|$
    $upper \leftarrow |pixel_{x,y} - pixel_{x,y-1}|$
    $rightUpper \leftarrow |pixel_{x,y} - pixel_{x+1,y-1}|$
    $edgeImg_{x,y} \leftarrow$ **max** $(left, upper, rightUpper)$
  else
    $edgeImg_{x,y} \leftarrow 0$
  end if
end for
$edgeImg \leftarrow$sharpen$(edgeImg)$
return$(edgeImg)$

Figure 1. Pseudocode for generating the edge image, see step 2.

space by means of an appropriate transformation. In contrast to the approaches presented in [1, 2, 8] our system only uses the luminance data (Y channel of YUV) during further processing. After that, luminance value thresholding is applied to spread luminance values throughout the image and increase the contrast between the possibly interesting regions and the rest of the image.

Step 2: Edge Detection. This step focuses the attention to areas where text may occur. We employ a simple method for converting the gray-level image into an edge image. Our algorithm (see Figure 1) is based on the fact that the character contours have high contrast to their local neighbors. As a result, all character pixels as well as some non-character pixels which also show high local color contrast are registered in the edge image. In this image, the value of each pixel of the original image is replaced by the largest difference between itself and its neighbors (in horizontal, vertical and diagonal direction). Despite its simplicity, this procedure is highly effective. Finally, the contrast between edges will be increased by means of a convolution with an appropriate mask.

Step 3: Detection of Text Regions. The horizontal projection profile of the edge image is analyzed in order to locate potential text areas. Since text regions show high contrast values, it is expected that they produce high peaks in horizontal projection. First, the histogram H is computed, where $el_y \in H$ is the number of pixels in line y of the edge image exceeding a given value.

In subsequent processing, the local maxima are calculated by the histogram determined above. Two thresholds are employed to find the local maxima. A line of the image is accepted as a text line candidate if either it contains a sufficient number (MinEdges) of sharp edges or the difference between the edge pixels in one line to its previous line is bigger than a threshold (MinLineDiff). Both thresholds are defined empirically and are fixed. In this way, a text region is isolated which may contain several texts aligned horizon-

Algorithm 3.2.
textRegion[] detectTextRegions(Image $edgeImg$)

comment: $edgeImg$ is created with Alg. 3.1
comment: textRegion is a data structure with 4 fields: x0, y0, x1, y1
comment: determineYCoordinates uses the Alg. 3.3
comment: determineXCoordinates uses the Alg. 3.4

Integer[] $H \leftarrow$calculateLineHistogram$(edgeImg)$
textRegions[] $TC \leftarrow$determineYCoordinate(H)
$TC \leftarrow$determineXCoordinate$(edgeImg, TC)$
return(TC)

Figure 2. Pseudocode for localizing text candidates, see step 3.

Algorithm 3.3.
textRegion[] determineYCoordinate(Integer[] H)

comment: H is the line histogram, see step 3
textRegion $rect$
textRegion[] TC
$y \leftarrow 1, j \leftarrow 0$
$insideTextArea \leftarrow false$
for $el_y \in H$ **do**
  if $((el_y > MinEdges)$ **or**
  $((el_y - el_{y-1}) > MinLineDiff))$ **then**
    if not $insideTextArea$ **then**
      $rect.y0 \leftarrow y$
      $insideTextArea \leftarrow true$
    end if
  else if $insideTextArea$ **then**
    $rect.y1 \leftarrow y - 1$
    if $((rect.y1 - rect.y0) > MinLines)$ **then**
      $TC[j] \leftarrow rect$
      $j \leftarrow j + 1$
    end if
    $insideTextArea \leftarrow false$
  end if
end for
return(TC)

Figure 3. Pseudocode for determining the y coordinates of text regions, see step 3.

Algorithm 3.4.
textRegion[] determineXCoordinate(Image $edgeImg$, textRegion[] TC)

$left \leftarrow maxInt$, $right \leftarrow -1$
for $textCandidate_i \in TC$ **do**
  for all $pixel_{x,y} \in textCandidate_i$ **do**
    if $(edgeImg_{x,y} \neq 0)$ **then**
      if $(left > x)$ **then**
        $left \leftarrow x$
      end if
      if $(right < x)$ **then**
        $right \leftarrow x$
      end if
    end if
  end for
  $textCandidate_i.x0 \leftarrow left$
  $textCandidate_i.x1 \leftarrow right$
end for
return(TC)

Figure 4. Pseudocode for determining the x coordinates of text regions, see step 3.

tally (whereby their y-coordinates are already defined). In a later step, we define the x-coordinates of the leftmost and rightmost, top and bottom point of the text region (see Figures 2, 3, 4). Finally, the exact coordinates for each of the detected areas are used to create bounding boxes.

Step 4: Enhancement and Segmentation of Text Regions. First, geometric properties of the text characters like the possible height, width, width to height ratio are used to discard those regions whose geometric features do not fall into the predefined ranges of values. All remaining text candidates undergo another treatment in order to generate the so-called text image where detected text appears on a simplified background. The binary edge image is generated from the edge image, erasing all pixels outside the predefined text boxes and then binarizing it. This is followed by the process of gap filling. If one white pixel on the binary edge image is surrounded by two black pixels in horizontal, vertical or diagonal direction, then it is also filled with black. The gap image is used as a reference image to refine the localization of the detected text candidates. Text segmentation is the next step to take place. It starts with extraction of text candidates from the gray image. Then, the segmentation process concludes with a procedure which enhances text to background contrast on the text image (see Figure 5).

4. Experimental Results

The proposed approach has been evaluated using data sets containing different types of images. The whole test data consists of 326 images where 296 of them were extracted from various MPEG videos, which were kindly provided to us by Lienhart [8]. These images can be further

Algorithm 3.5.
Image SegmentTextRegions(Image $edgeImg$, textRegion[] TC)

comment: $edgeImg$ is created with Alg. 3.1
comment: TC is the array returned from Alg. 3.4
Image $reducedImg$ ←erase$(TC, edgeImg)$
Image $binaryImg$ ←binarize$(reducedImg)$
Image $gapImg$ ←fillGaps$(binaryImg)$
TC ←refineCoordinates$(edgeImg, gapImg, TC)$
Image $textImg$ ←extractImage$(grayImg, TC)$
$textImg$ ←enhanceContrast$(textImg)$
return$(textImg)$

Figure 5. Pseudocode for generating the text image, see step 4.

Figure 6. An image from the "credits test set".

Figure 7. The text detection result for the image from the "credits test set".

divided into two groups, based on the video genre:

- Sequences from different commercial advertisements (called the ”commercials test set”).
- Sequences from different films, with a lot of text lines scrolling downwards, pre-title and credits title sequences (subsequently called the ”credits test set”).

Test set	#Images	#Textlines	#Correct detected	False positive	Recall(%)	Precision(%)
Credits	209	900	823	131	91.4	86.3
Commercials	87	151	117	49	77.4	70.4
News	30	53	39	8	73.5	82.9
Total #images	326	1104	979	188	88.7	83.9

Table 1. The text detection results for the whole data set.

Figure 8. An image from the "commercials test set".

Figure 9. The text detection result for the image from the "commercials test set".

Figure 10. An image from the "news test set".

Figure 11. The text detection result for the image from the "news test set".

We have chosen to extract one frame each second if either the background or the text being displayed changes significantly within that second. Otherwise, we have checked the next frame one second later in the same way. All these images were coded as 24-Bit RGB JPEG images with a size of 384 by 288 pixels.

The remaining images were taken from our image database which had been created during a former media research project on the presentation of politicians in selected TV evening news broadcasts between 1950 and 2000 (called "news test set"). They have a resolution of 384*288 pixels. The selection of these test images is based on the complexity of background and text regions. We have manually checked the output for each image and measured the number of correctly detected text regions as well as the number of falsely detected text regions (regions without text). A detected text line has been accepted as correct if from our point of view the probability is high that an OCR system could recognize the text characters. It should be pointed out that all the system parameters remain the same throughout the whole set of test images.

One sample for each of our test sets is shown in figure 6 ("credits test set"), figure 8 ("commercials test set") and figure 10 ("news test set"). In these figures the original images are presented, while in figure 7, figure 9 and figure 11 the output image is shown for each image. The output image of the proposed algorithm only consists of detected text regions. The results for the experiments are presented in table 1 where the number of really existing text lines, the number of detected text lines, the number of false alarms and the corresponding values for recall and precision are listed. Recall is defined as:

$$Recall = \frac{Correct\ Detected}{(Correct\ Detected + Missed\ Text\ Lines)}$$

whereas precision is defined as:

$$Precision = \frac{Correct\ Detected}{(Correct\ Detected + False\ Positives)}$$

The experimental results vary depending on the different test sets. For example, a good performance has been

achieved for the "credits test set" with a recall of above 91% and a precision of 86%. The performance for the other test sets is lower because those images have even more complex backgrounds and text in these images varies in fonts, sizes and colors. Altogether, the overall performance for the whole test set of images is 88.7% for recall and 83.9% for precision.

5. Conclusions

In this paper, we have presented an approach to detect, localize, and extract texts appearing in grayscale or color images. The proposal is based on employing a color reduction technique, a method for edge detection and region segmentation, and selecting text regions based on their horizontal projection and geometrical properties. The software is completely written in JAVA to be able to easily run the code in parallel on possibly heterogenous networked computing platforms. Experimental results on a set of images have demonstrated the performance of our approach, achieving an overall recall of 88.7% and a precision of 83.9%.

There are several areas for future work. First, we plan to employ an OCR system to check the recognition performance for the text images produced by the proposed algorithm. Second, the low quality of the old news video material respectively the variety of text appearances in advertisement images make the process of text detection more complex, such that the detection performance of our algorithms must be improved. Third, the approach will be extended to also work with video sequences instead of still images. Finally, we plan to implement a hybrid system where connected component-based methods are combined with texture-based methods to possibly obtain further performance improvements.

6. Acknowledgements

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

References

[1] L. Agnihotri and N. Dimitrova. Text Detection for Video Analysis. *In Proc. of the International Conference on Multimedia Computing and Systems,* Florence, Italy, pp. 109-113, 1999.

[2] M. Cai, J. Song and M. R. Lyu. A New Approach for Video Text Detection. *In Proc. of International Conference On Image Processing*, Rochester, New York, USA, pp. 117-120, 2002.

[3] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, 1999.

[4] 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. 4, pp. 2326-2330, 2000.

[5] A. K. Jain and B. Yu. Automatic Text Location in Images and Video Frames. *In Proc. of International Conference of Pattern Recognition (ICPR)*, Brisbane, pp. 1497-1499, 1998.

[6] P.-K. Kim. Automatic Text Location in Complex Color Images Using Local Color Quantization. *IEEE TENCON*, Vol. 1, pp. 629-632, 1999.

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

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

[9] 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, 1997.