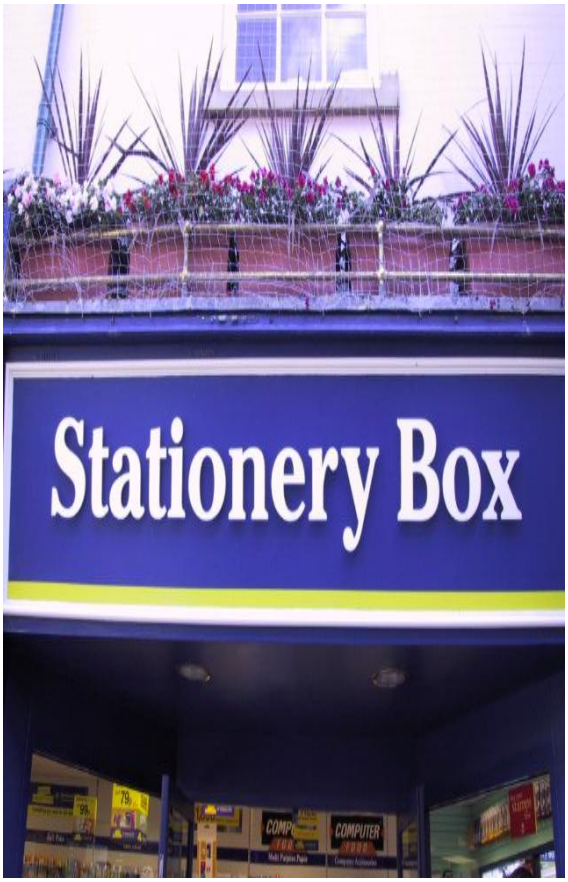
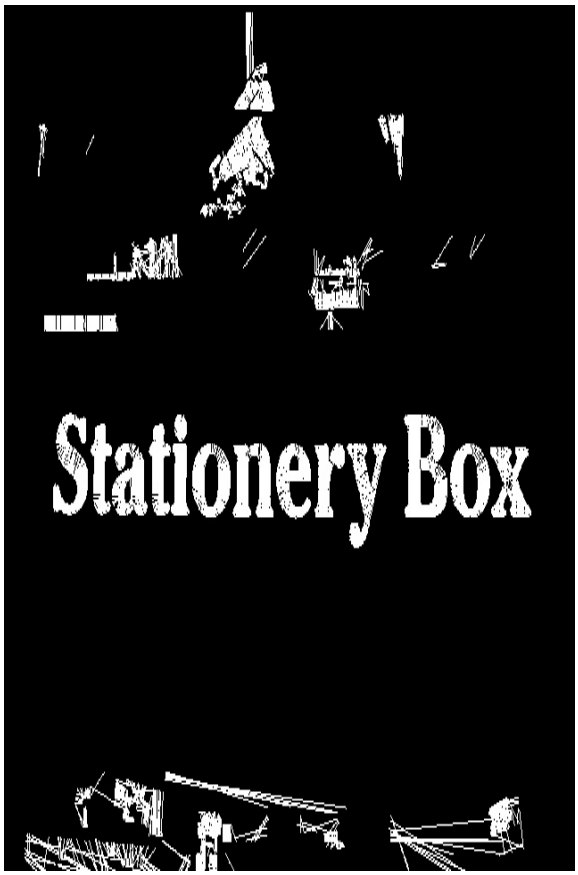




Text Detection in Natural Images

Guided By:

Dr. Vinod Pankajakshan

Submitted By:

Arjun Krishna 15115030

A S Akil Arif Ibrahim 15116001

Table of contents

Abstract 1

Introduction 2

Related Works 2

Theory 3

- MSER 3
- Stroke Width Transform (SWT) 4
- Finding letter candidates 6
- Grouping letter candidates into text regions 7

Experiments 8

Results Error! Bookmark not defined.

Conclusion 16

References 17

Abstract

Detecting text in natural images, as opposed to documents is an important phenomenon. Automatic text recognition induces a lot of potential applications such as:

- Helping a foreigner to understand the contents on an information board (by translating the recognized text into his/her native language) .
- Computerized aid for visually impaired.
- Robotic navigation and intelligent transport system.

In this project, we have used two approaches to detect text regions in natural scene images- Stroke Width Transform (SWT) and MSER detection followed by SWT. SWT can

detect texts in many fonts and languages. We have implemented this algorithm in python, and made use of OpenCV. Finally, we tested this algorithm on some images from ICDAR 2003 dataset.

Introduction

With the rapid growth of technology there are many camera based applications are available in portable devices like cell phones,Tablets etc. Everyone is able to capture the images easily, but whenever we want to read the text presented in those images are very difficult.This is the main problem in computer vision community.The problem of text detection was considered in a number of recent studies[11],[12],[13],[14],[15].

There are 4 stages of extracting text from an image:

1. Text detection: Finding the bounding region of the text in an image.
2. Text localization: Grouping text regions into text instances and generating a set of tight bounding boxes around all text instances.
3. Text binarization: Binarizing the text bounded by text regions and marking text as one binary level and background as the other.
4. Text recognition: Performing Optical Character Recognition (OCR) on the binarized text image.

Researches have shown that the performance of text retrieval algorithms depend critically on the performance of their text detection modules. Therefore, we focus on that problem in this project.

Related Works

The available methods for text detection can be classified into two groups:

1.Texture based: These methods scan the image at various scales and classify neighbourhood regions based on properties like *high density of edges, low gradients above and below, high variance of DCT coefficients, intensity etc* [2],[3],[4],[5]. The limitations of the methods include the computational complexity due to need of scaling and the problem

of how to combine information from different scales and lack of precision due to the fact that only a few scales of images contain the text.

2.Region based: These type of methods rely on scale invariant features like colour constancy, stroke width etc[7],[8],[9]. They group together neighbourhood of pixels having these features and form connected components (CCs). The resulting connected components are then filtered geometrically to exclude CCs that are certainly not texts.

Theory

The algorithm receives an RGB image and returns an image of the same size, where the regions of suspected text are marked. It has 4 major steps- MSER detection, the stroke width transform, grouping the pixels into letter candidates based on their stroke width, and finally, grouping letter candidates into regions of text.

Maximally Stable Extremal Regions (MSER)

MSER is a method for blob detection in the images. It is a stable connected component of some gray level sets of the image [6]. MSER depends on the threshold of the image. The pixels below that threshold value are white and all those above or equal are black. MSER detects the objects (Connected components) and all the objects can be filled with different color. In this process, some of the regions include the extra background pixels. Those are removed in the canny edge detection process.

Implementation of MSER:

1. First, sweep the threshold of intensity from black to white performing a simple luminance thresholding of the image.
2. Then, extract the connected components (Extremal Regions). Find a threshold when an extremal region is maximally stable; a specified number (in the paper[6], the number used was 5) of consecutive thresholds give highly overlapping connected components.
3. Finally, we get the regions descriptors as features of MSER.

Source Image MSER Detection

Stroke Width Transform (SWT)

The *Stroke Width Transform* (SWT) is a local image operator which computes per pixel the width of the most likely stroke containing the pixel [1]. The output of the SWT is an image of size equal to the size of the input image where each element contains the width of the stroke associated with the pixel. We define a stroke to be a contiguous part of an image that forms a band of a nearly constant width. We do not assume to know the actual width of the stroke but rather recover it.

Calculation of SWT:

The initial value of each element of the SWT is set to ∞. In order to recover strokes, we first compute edges in the image using Canny edge detector.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

After that, a gradient direction dp of each edge pixel p is considered (for gradient calculations we have used sobel filter because it gives better approximation of gradient of a pixel; a sobel filter is a gaussian filter followed by an edge detector).

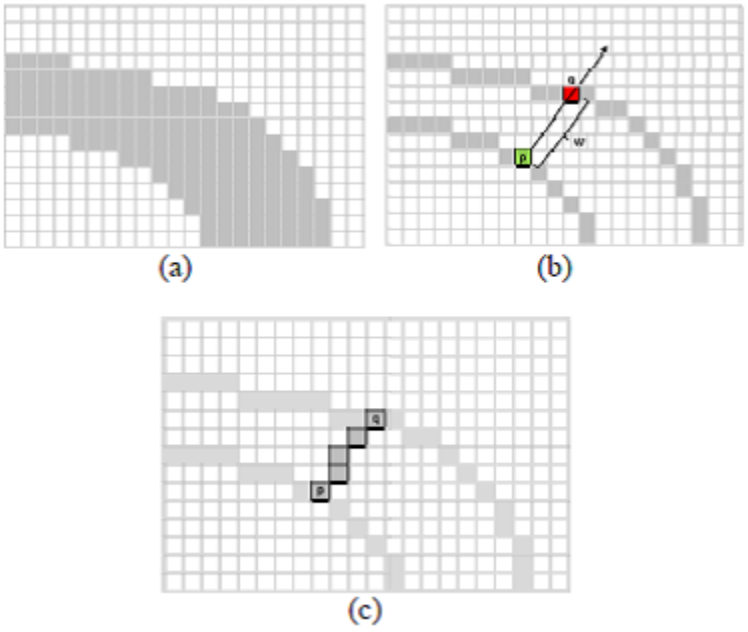
-1	-2	-1
0	0	0
1	2	1

Horizontal

-1	0	1
-2	0	2
-1	0	1

Vertical

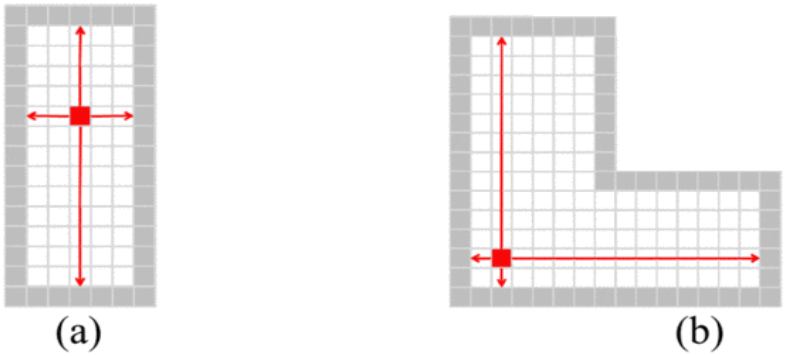
If p lies on a stroke boundary, then $\overrightarrow{dp}$ must be roughly perpendicular to the orientation of the stroke. We follow the ray $\vec{r}=\vec{p}+n\cdot\overrightarrow{dp}$, n>0 until another edge pixel q is found. We then consider the gradient direction dq at pixel q. If $\overrightarrow{dq}$ is roughly opposite to $\overrightarrow{dp}$ ($\overrightarrow{dq}=-\overrightarrow{dp}\pm\pi/6$), each element s of the SWT output image corresponding to the pixels along the segment [p,q] is assigned the width $\|\vec{p}-\vec{q}\|$ unless it already has a lower value . Otherwise, if the matching pixel q is not found, or if $\overrightarrow{dq}$ is not opposite to $\overrightarrow{dp}$, the ray is discarded. The figure below shows the process of SWT computation.



In order to accommodate both bright text on a dark background and dark text on a bright background, we need to apply the algorithm twice: once with the ray direction $\overrightarrow{dp}$ and once with -$\overrightarrow{dp}$.

After the first pass described above, pixels in complex locations might not hold the true stroke width value (see below figure). For that reason, we will pass along each

non-discarded ray, where each pixel in the ray will receive the minimal value between its current value, and the median value along that ray.



Finding letter candidates

We now have a map of the most likely stroke-widths for each pixel in the original image. The next step is to group these pixels into letter candidate. This will be done by first grouping pixels with similar stroke width, and then applying several rules to distinguish the letter candidates.

The grouping of the image will be done by using a Connected Component algorithm. In order to allow smoothly varying stroke widths in a letter, we will let two pixels to be grouped together if their SWT ratio is less than 3.0.

Now we must detect the connected components which can pass as letter candidates, by applying a set of fairly flexibly rules. These rules are as follows:

1. **Variance of SWT**: The variance of the stroke-width within a component must not be too big. This helps with rejecting foliage in natural images, which are commonly mistaken for text. The selected threshold is half the average stroke width of a particular connected component.
2. **Aspect Ratio**: The aspect ratio of a component must be within a small range of values, in order to reject long and narrow components. We limit the aspect ratio to be between 0.1 and 10.

3. The **ratio between the diameter of the component and its median stroke width** to be less than a learned threshold. This also helps reject long and narrow components. We limit the ratio between the diameter of the connected component and its median stroke width to be a value less than 10.
4. **CC size**: Components whose size is too large or too small will also be ignored. This is done by limiting the height and width of the component. We limit the width and font height to be between 10 and 300 pixels.

The remaining connected components are considered letter candidates, and are now to be aggregated into regions of text.

Grouping letter candidates into text regions

Since single letters are not expected to appear in images, we will now attempt to group closely positioned letter candidates into regions of text. This filters out many falsely-identified letter candidates, and improves the reliability of the algorithm results.

Again, we will use a small set of rules to group letters together into regions of text. These rules will consider pairs of letters, and are as follows:

1. Two letter candidates should have similar stroke width. For this reason we limit the ratio between the median stroke-widths to be less than some threshold. That threshold is chosen to be 2.0.
2. The ratio between the heights of the letters must not exceed 2.0. This is due to capital letters next to lower case letters.
3. The distance between letters must not exceed three times the width of the wider one.

At the next step of the algorithm, the candidate pairs determined above are clustered together into chains. Initially, each chain consists of a single pair of letter candidates. Two chains can be merged together if they share one end and have similar direction. The

process ends when no chains can be merged. Each produced chain of sufficient length (at least 3 letters) is considered to be a text line.

In this way, text localization is performed.

Experiments

We used images from the ICDAR 2003 dataset. We implemented two models- one which performs text localization using the SWT, and the other uses MSER as well as SWTs. In the latter model, MSER detection is done first, and SWT is performed only in these regions. The implementation was carried out in python. OpenCV was used.

First, Canny edge detection is done. Then, gradient map is created using two sobel filters, one in both horizontal and vertical directions. Then, SWT is calculated using the algorithm discussed earlier. Next, letter candidates are found and then grouped into text regions. The final image contains the text regions detected by the algorithm.

The implementation can be found here: https://github.com/akilarif/Text-Detection-in-Natural-Images-Using-SWT-and-MSER-followed-by-SWT

Results

(1) Source Image

Letters

Labels

Final Result

(2) Source Image Labels

Letters Final Result

(3) Source Image

Labels

Letters

Final Result

(4) Source Image

Labels

Letters

Final Result

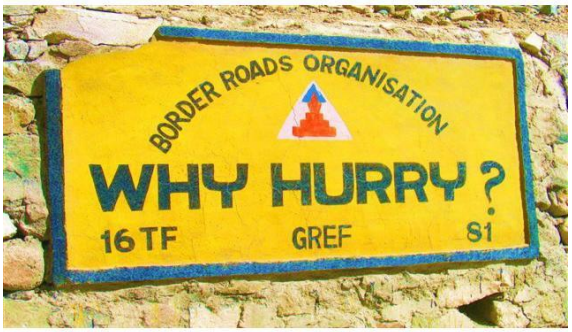
Strengths of the algorithm:

This algorithm can detect not only English letters, but can also detect letters of different languages, like Japanese (4). The texts can be of varying sizes (1).

Weakness of the algorithm:

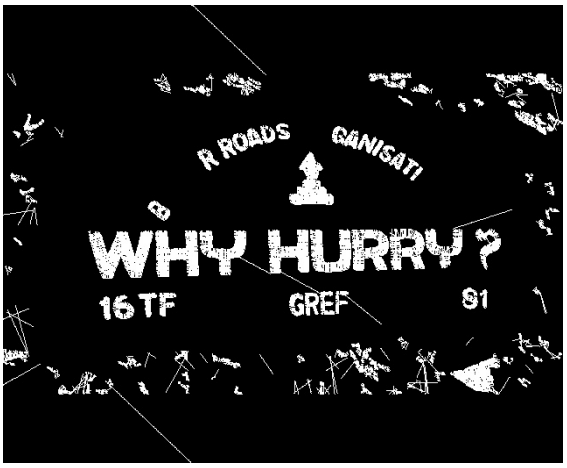
Noise can be detected in certain cases (1, 2). This becomes more prominent when there is foliage in the image. Round and curved letters may not be detected, like the one below.

Source Image

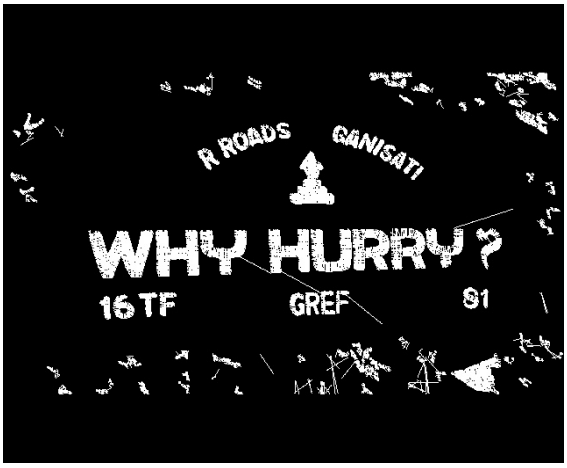


Labels

Letters



Final Result

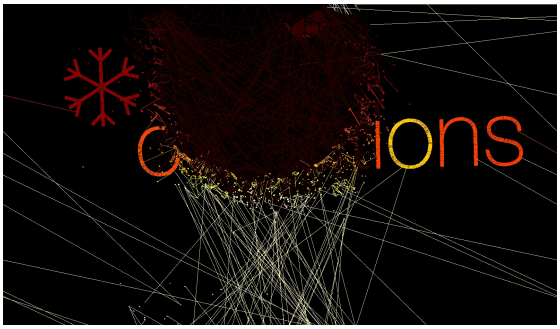


There is significant noise in the result due to the presence of the rocky surface. If the thresholds in the 'grouping letter candidates into text regions' step are relaxed, then the curved letters might be recognized, but there will be more noise.

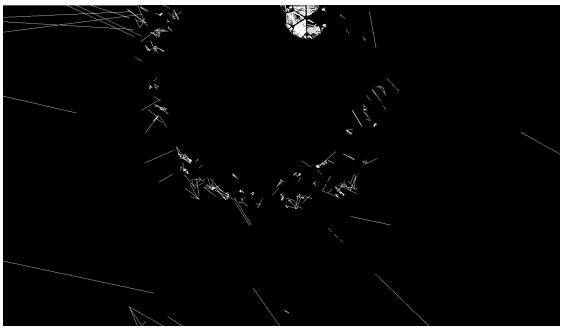
Also, this algorithm fails when there are strong highlights in the image, like the one below.

Source Image

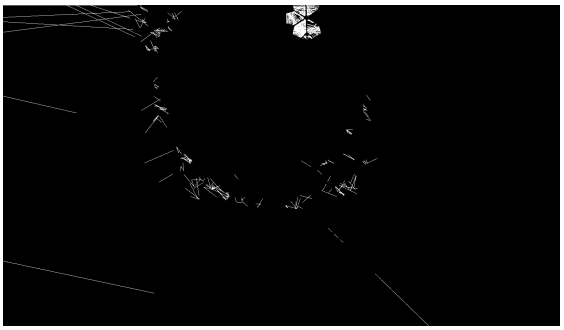
Labels



Letters



Final Results



Results of MSER followed by SWT:

(1) MSER Detection Final Result

The final result in this case is worse than that obtained by SWT only algorithm. This is because the MSER detection didn't detect the text regions correctly.

(2) MSER Detection Final Result

There is slightly less noise in the final result of this algorithm compared to SWR only algorithm.

(3) MSER Detection Final Result

The final results of this algorithm and SWT only algorithm are pretty much the same.

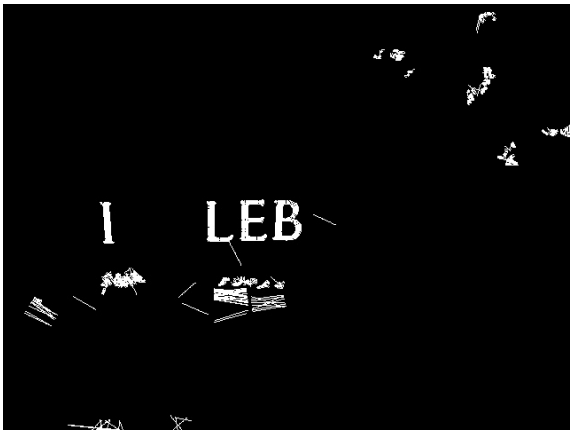
(4) MSER Detection Final Result

The noise in the top right corner in the final result of SWR only algorithm is not present in the final result of this algorithm. But, there is a small addition of noise near the end of second line. Both the final results are very similar.

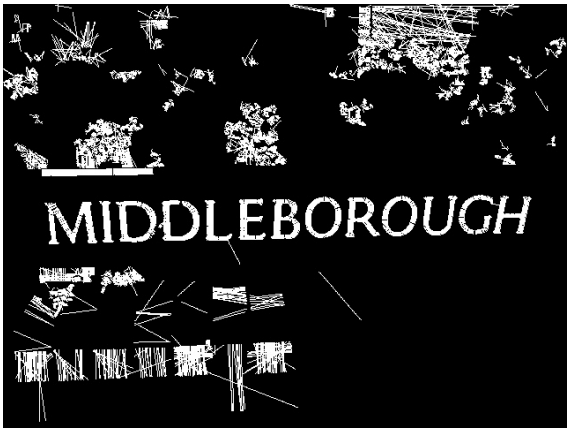
One case where MSER followed by SWT gives a much better result compared to SWT only algorithm is given below.

Source Image

Final Result- SWR only



Final Result- MSER followed by SWT



In this case, SWT gave a worse result due to the presence of foliage close to the text region. MSER detection eliminated some of the foliage.

Conclusion

In this work, we show how to make use of stroke width transform and maximally stable extremal regions for text detection. This provides features that has proven to be reliable and robust for text detection. Unlike previous features used for text detection, SWT combines dense estimation (computed at every pixel) with non-local scope (stroke width depends on information contained sometimes in very far apart pixels). MSER is also dense estimation. Since we have not used any texture maps specific to any language, we can apply the method to many languages and fonts.

There are several possible extensions for this work:

1. The grouping of letters can be improved by considering the directions of the recovered strokes. This may allow the detection of curved text lines as well.
2. Better connected component algorithm to improve grouping.
3. Better Edge detection stage by using a better edge detector that can compensate for noise to a certain extent (for example, bandlet based edge detector[10]).

References

1. Boris Epshtein, Eyal Ofek, and Yonatan Wexler, “Detecting Text in Natural Scenes with Stroke Width Transform”, Proc. 23rd IEEE Conference on Computer Vision and Pattern Recognition, pp. 2963-2970, 2010.
2. X. Chen, A. Yuille, "Detecting and Reading Text in Natural Scenes", Computer Vision and Pattern Recognition (CVPR), pp. 366-373, 2004.
3. R. Lienhart, A. Wernicke, “Localizing and Segmenting Text in Images and Videos”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 4, April 2002, pp. 256- 268.
4. J. Gllavata, R. Ewerth, B. Freisleben, “Text Detection in Images Based on Unsupervised Classification of High-Frequency Wavelet Coefficients”, 17th International Conference on Pattern Recognition (ICPR 2004) - Vol. 1, pp. 425-428 .
5. H. Li, D. Doermann, O. Kia, "Automatic Text Detection and Tracking in Digital Video", IEEE Transactions on Image Processing, Vol. 9, No. 1, January 2000 .
6. Text Detection On Scene Images Using MSER,international Journal of Research in Computer and Communication Technology, Vol 4, Issue 7 , July -2015
7. A. Jain, B. Yu, “Automatic Text Location in Images and Video Frames”, Pattern Recognition 31(12): 2055-2076 (1998)
8. H-K Kim, "Efficient automatic text location method and content-based indexing and structuring of video database". J Vis Commun Image Represent 7(4):336–344 (1996)
9. Y. Liu, S. Goto, T. Ikenaga, "A Contour-Based Robust Algorithm for Text Detection in Color Images", IEICE TRANS. INF. & SYST., VOL.E89–D, NO.3 MARCH 2006
10. Image Text Detection Using a Bandlet-Based Edge Detector and Stroke Width Transform.

11. K. Jung, K. Kim, A. K. Jain, “Text information extraction in images and video: a survey”, Pattern Recognition, p. 977 – 997, Vol 5. 2004.
12. X. Chen, A. Yuille, "Detecting and Reading Text in Natural Scenes", Computer Vision and Pattern Recognition (CVPR), pp. 366-373, 2004
13. R. Lienhart, A. Wernicke, “Localizing and Segmenting Text in Images and Videos” IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 12, NO. 4, APRIL 2002, pp. 256- 268
14. A. Jain, B. Yu, “Automatic Text Location in Images and Video Frames”, Pattern Recognition 31(12): 2055-2076 (1998)
15. H-K Kim, "Efficient automatic text location method and content-based indexing and structuring of video database". J Vis Commun Image Represent 7(4):336–344 (1996)