Expert Systems with Applications 39 (2012) 10832–10840

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A robust video text detection approach using SVM

Yi Cheng Wei, Chang Hong Lin *

Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, ROC

ARTICLE INFO

Keywords:
Discrete wavelet transform
K-means
Principal component analysis
Pyramidal method
Support vector machine
Video text detection

ABSTRACT

A new method for detecting text in video images is proposed in this article. Variations in background complexity, font size and color, make detecting text regions in video images a difficult task. A pyramidal scheme is utilized to solve these problems. First, two downsized images are generated by bilinear interpolation from the original image. Then, the gradient difference of each pixel is calculated for three differently sized images, including the original one. Next, three K-means clustering procedures are applied to separate all the pixels of the three gradient difference images into two clusters: text and non-text, separately. The K-means clustering results are then combined to form the text regions. Thereafter, projection profile analysis is applied to the Sobel edge map of each text region to determine the boundaries of candidate text regions. Finally, we identify text candidates through two verification phases. In the first verification phase, we verify the geometrical properties and texture of each text candidate. In the second verification phase, statistical characteristics of the text candidate are computed using a discrete wavelet transform, and then the principal component analysis is further used to reduce the number of dimensions of these features. Next, the optimal decision function of the support vector machine, obtained by sequential minimal optimization, is applied to determine whether the text candidates contain texts or not.

1. Introduction

Since the text-information in video frames always carries a crucial clue for video data indexing and retrieving applications, video text detection plays an increasingly important role in recent years. Although a large number of approaches have been proposed over the past years, detecting texts in video is still a challenging problem. It is because video images often have complex background and texts in the video images may have different sizes, colors, styles and alignment. These factors influence video text detection results significantly.

Generally, video texts can be characterized into two types: caption texts and scene texts. Caption texts are superimposed during video editing processes and scene text exists in the real natural world. Caption texts are easier to be detected because they usually have higher contrast and are much clearer than scene texts in the video images (Shivakumara, Phan, & Tan, 2009a, 2009b; Zhang & Kasturi, 2008).

Text detection methods can generally be classified into five categories: edge (gradient) based (Liu, Wang, & Dai, 2005; Lyu & Song, 2005; Ngo & Chan, 2005; Phan, Shivakumara, & Tan, 2009; Shivakumara et al., 2009a, 2009b; Wang, Huang, & Liu, 2010), connected component and color based (Kim & Kim, 2009; Mariano & Kasturi, 2000; Sun, Liu, Qian, & Gao, 2009; Wong & Chen, 2000; Zhang, Sun, & Gu, 2010), texture based (Gllavata, Ewerth, & Freisleben, 2004; Ye, Gao, Wang, & Zeng, 2003; Ye & Huang, 2004), video temporal information based (Miao, Huang, Jiang, & Gao, 2008; Tang, Luo, Gao, Pissaloux, & Zhan, 2002) and stroke based (Gong, Zeng, & Zhang, 2011; Liu, Jung, Kim, Moon, & Kim, 2006) methods. For edge based algorithms, it is difficult to select the threshold value during text extraction, even though they are faster and efficient. Connected component and color based algorithms need uniform color text conditions for detection, and are more time consuming when the color space analysis is performed. Texture based algorithms extract video image features of text and non-text regions by using wavelet, FFT, DCT, PCA, or Gabor filters, etc., and then apply those features to a learning machine to locate text regions. Video temporal information based algorithms need more video frames to detect video texts, and their drawback is that they cannot detect moving captions and the background of the video stream must be changing successively during the text detection time. Stroke based algorithms use a stroke filter to search characters in the video images, and are the latest developed algorithms. The researchers usually use more than one of the above algorithms to accomplish text detection for better results.

In this article, we develop a new robust algorithm to detect texts in video images based on the pyramidal gradient difference classification. First, we resize the original image to form a three-level pyramid in order to detect fonts in different sizes. Second, we calculate the gradient differences of these three different-sized images, separately. Third, we use the K-means clustering to classify

* Corresponding author.
E-mail address: chlin@mail.ntust.edu.tw (C.H. Lin).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.010

each gradient difference map into text and non-text regions. Next, we combine the three clustering results to form the text regions of the image. Then, we refine each text candidates' boundary through adaptive profile analysis based on the Sobel edge map (SM). Finally, two phases of text candidate verification is applied to remove false positives by using text candidates' geometrical properties, including the texture of text candidates and the support vector machine (SVM) method.

In Section 2, we review some prior and related work. Section 3 presents the proposed text detection method. In Section 4, we describe our experimental results. Finally, conclusions and future work are presented in Section 5.

2. Prior and related work

Many researchers have put efforts on detecting texts in video streams. Shivakumara et al. (2009b) adopt a zero crossing technique to determine the bounding boxes for the detected text lines which does not require complete spacing between the text lines. Wang et al. (2010) classify the text background into three categories. Then, for the three different types of video frames, they use different text location methods to handle them. Ngo and Chan (2005) classify the text background into four categories. A binomial filter and shift operators are applied to the edge image to extract the text regions. Zhang et al. (2010) have presented a color-edge combined method which uses the RGB colors to generate the transition map of input images, and then uses the text edges of different gray-level images to yield the possible text area. Gllavata et al. (2004) apply a wavelet transform to the image, and uses high-frequency wavelet coefficients to classify text area by K-means. Liu et al. (2006) have presented a stroke filter to find the text strokes. They then use morphologic operation and projection profiles to detect text candidates. After that, they use an SVM classifier to verify text candidates. Several recent literatures for text classification can also be found in Gong et al. (2011), Yun, Jing, Yu, and Huang (2010), Fernandez-Caballero, Lopez, and Castillo (2012).

In this article, we propose a video text detection method based on pyramidal gradient differences clustering. Pyramidal method can help us to detect texts in different font sizes and generate denser edge maps to refine the text candidate boundaries, as shown in Fig. 1.

3. Proposed method

The proposed approach is composed of three major steps: (*A*) text region detection using pyramidal gradients; (*B*) text candidate boundary refinement; (*C*) text candidate identification. The flowchart of the proposed method is shown in Fig. 2.

3.1. Text region detection using pyramidal gradients

In this step, we first resize the input image into grayscale images of three different sizes, *original, shrunk by 1/2* and *shrunk by 1/4*. The resized images are constructed by using bilinear interpolation which the pixel value of the shrunk image is a weighted average of pixels in the nearest 2×2 neighborhood. Then, the horizontal gradients, vertical gradients and the maximum gradient difference (MGD) maps of the image pyramid are calculated.

Since gradients always indicate the directional change in the intensity of the images, and text regions often have high contrast to its background, gradients are often used to differentiate text and non-text regions. The other advantage of gradient based algorithms is they will not be influenced by the polarity of text colors (text color could be either darker or lighter than its background).

We first use a horizontal mask $[-1\,1]$ and a vertical mask $[-1\,1]^T$ to compute the gradient image f_h and f_v by convolving each size of the image with the masks separately. Then MGD maps are obtained by calculating the difference between the maximum and minimum gradient values from the two direction gradient images within a $1 \times N$ local window centered at each gradient image pixel (i,j), as shown in Eq. (1) (Phan et al., 2009; Shivakumara et al., 2009a).

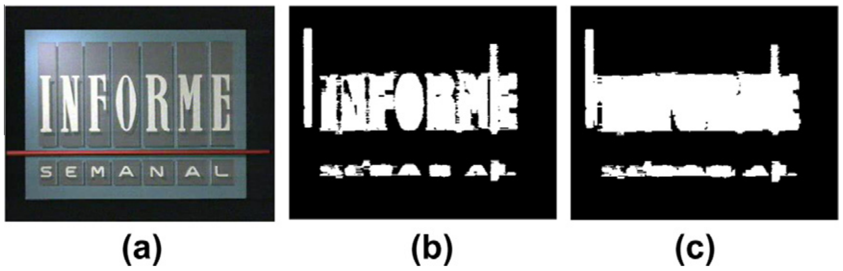


Fig. 1. Pyramidal method text detection. (a) Input image. (b) Non-pyramidal method clustering. (c) Pyramidal method clustering.

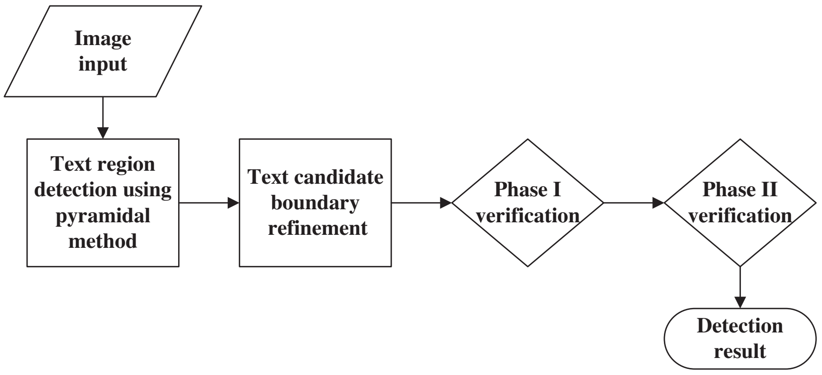


Image input
Text region detection using pyramidal method
Text candidate boundary refinement
Phase I verification
Phase II verification
Detection result

Fig. 2. Flowchart of the proposed method.

$$\mathrm{MGD}(i,j) = \max_{c\in h,v}(f_c(i,j+k)) - \min_{c\in h,v}(f_c(i,j+k)) \quad (1)$$

where $k \in \left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$. MGD map is generated by moving the local window over the three size gradient images at the two directions.

$$Eg(i,j) = \sum_{i,j}\mathrm{MGD}^2(i,j) \quad (2)$$

Since using both MGD maps and their energy uniformities can generate a stable center of the K-means clustering, this will result in stable text regions. Another advantage of adding the energy uniformity of MGD maps to the K-means clustering also can obtain better clustering result. We normalize the values of MGD maps to the range of [0,1], and calculate the energy uniformity maps of MGD maps within a 8×4 local window centered at each MGD map pixel (i,j), as shown in Eq. (2) (Jung, Liu, & Kim, 2009), then apply them to the K-means clustering to separate the MGD map into two clusters: text and non-text clusters, as shown in Fig. 3.

After completing the K-means clustering of each size of the MGD map, we resize the two downsized images to its original size through bilinear interpolation, and discard unwanted connected components. The three images are then combined to form the final K-means result, which we choose the intersection of the results of the two downsized images and then union with the result of the original image, as shown in Fig. 4.

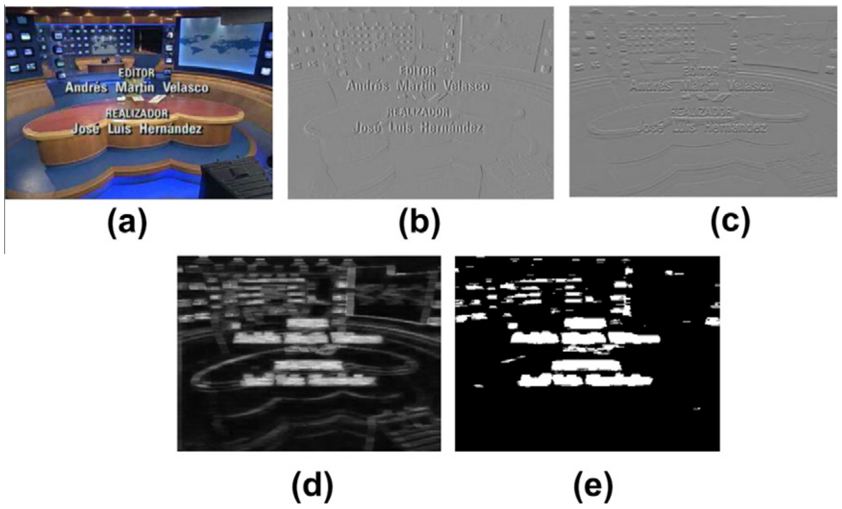


Fig. 3. Text region detection. (a) Input image. (b) Horizontal gradient image. (c) Vertical gradient image. (d) MGD map. (e) Text clustering.

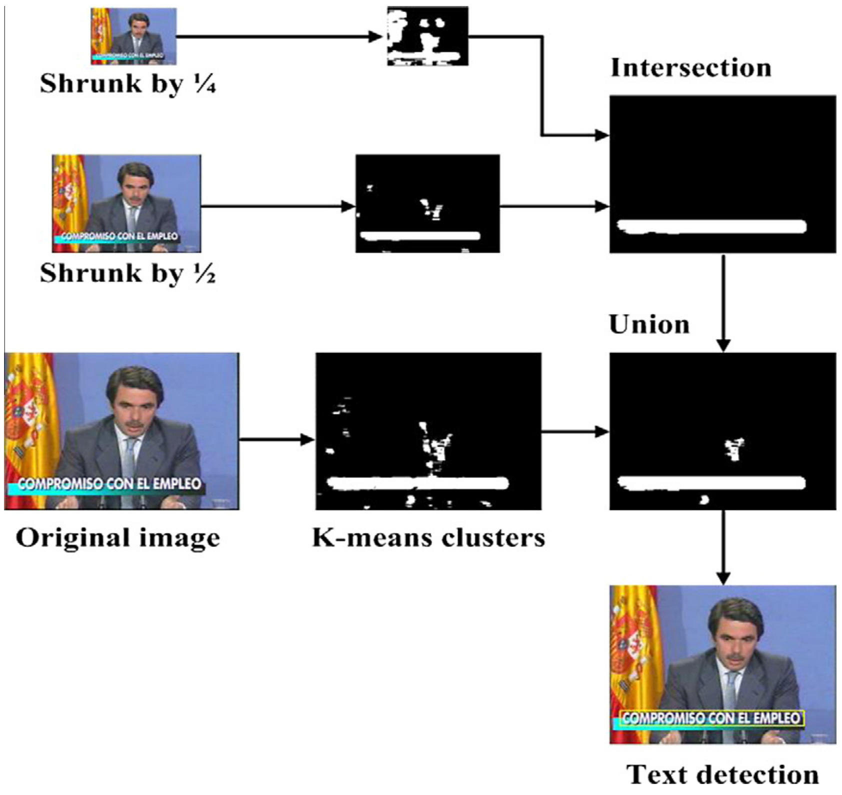


Fig. 4. Combined K-means clusters.

3.2. Text candidate boundary refinement

To refine the text regions from the K-means clustering, we label each text region as a connected component using 4-adjacency type (Suzuki, Horiba, & Sugie, 2003), and discard regions too small to be text regions. Then, we perform the refinement to every labeled connected component by using the SM generated from the gray-scale image of the input image.

Determining the boundary of each text region, we adopt an adaptive method to find the projection profile threshold. First, we used the SM to calculate the background complexity (*BC*) (Wang et al., 2010) and the projection profile arithmetic mean value (*Mean*(*i*)) of each text region *i* as follows:

$$BC(i) = \frac{\text{Edge number of the text region } i}{\text{Size of the text region } i} \tag{3}$$

$$Mean_y(i) = \frac{1}{m_2 - m_1} \sum_{y=m_1}^{m_2} P_i(y) \tag{4}$$

$$Mean_x(i) = \frac{1}{n_2 - n_1} \sum_{x=n_1}^{n_2} P_i(x) \tag{5}$$

where $P_i(y)$ denotes the horizontal projection profile value, $P_i(x)$ denotes the horizontal projection profile of the SM of each text region *i*, *m* is the coordinates of the text region on the *y* axis, and *n* is the coordinates of the text region on the *x* axis.

Since it is difficult to determine the projection profile threshold to refine the boundary of each text region, we select the *BC* value and the text region size as references to set the threshold according to the mean value of the text region. If $P_i(y)$ is smaller than the threshold, row *y* will be set as a gap; otherwise it will be set as a text line. We perform boundary refinement procedure repeatedly until there is no more change on the boundary.

The *x* axis vertical projection profile threshold of each text region can be obtained as the same manner. Example of refining a text region boundary, is shown in Fig. 5. After the boundary refinement procedure completes, the text candidates are obtained.

3.3. Text candidate identification (Phase I)

At the phase one of the text candidate identification to eliminate false positives, we use geometrical properties and text textures of text candidates to identify each text candidate. For geometrical properties, we calculate the width (W_i), height (H_i), aspect ratio (AR_i), area (A_i), and SM density (SD_i) of each text candidate *i* as follows (Phan et al., 2009).

$$A_i = W_i \times H_i \tag{6}$$

$$AR_i = \frac{W_i}{H_i} \tag{7}$$

$$SD_i = \frac{\sum SM(i,j),\ \text{where } (i,j) \in \text{text candidate } i}{\text{text candidate } i \text{ area}} \tag{8}$$

If $AR_i < T_1$ or $SD_i < T_2$, the text candidate is thought as a false positive; otherwise, it is accepted as a text block. After the geometrical properties being identified, we then examine the texture of text candidates to decrease the number of false positives of the text candidates.

For the text texture analysis, we use the local binary pattern (LBP), which is a highly discriminative and efficient tool to indicate the texture of grayscale images (Kim & Kim, 2009). First, we compute LBPs and the number of different LBPs (NOL), which is normalized by the maximum of the number of different LBPs for each text candidate. The LBPs can be calculated as follows:

$$\text{LBP}_{8,R} = \sum_{i=1}^{8} s(g_i - g_c) \times 2^{i-1}, \quad \text{where } s(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geqslant 0 \end{cases} \tag{9}$$

where *R* denotes the radius and 8 is the number of neighbor pixels of a specific pixel. g_c and g_i denote the intensity of the current pixel and its neighbor pixels, respectively. An example of computing LBP is shown in Fig. 6. Then, we can calculate the edge density *D* as a weight of each text candidate as follows:

$$D(m,n) = \frac{1}{(2J+1)(2K+1)} \sum_{j=-J}^{J} \sum_{k=-K}^{K} E(m+j, n+k) \tag{10}$$

where $E(j,k)$ is the value of pixel (j,k) on the SM and for our experiment we choose $J = K = 1$. Next, we produce the *FOL* (Factor of LBPs) by multiplying NOL to the edge density (D_i) of each text candidate *i*.

$$FOL_i = D_i \times NOL_i \tag{11}$$

If the FOL_i of the text candidate is larger than a predefined threshold, the text candidate is accepted as a text block; otherwise, it is considered as a false positive. On various experimental results, the threshold value of *FOL* is selected as 0.1 empirically.

3.4. Text candidate identification (Phase II)

Even though some false positives can be eliminated at the first phase of the text candidate identification, the effect is limited. This

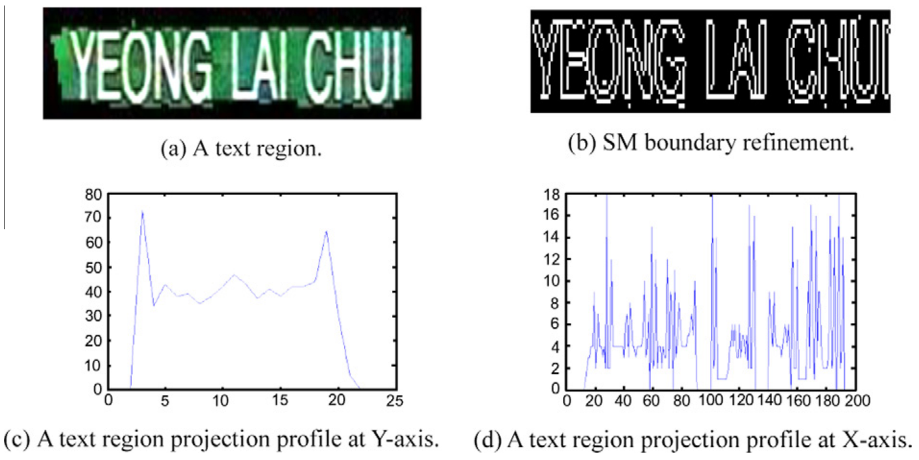


Fig. 5. Text region boundary refinement using projection profile.

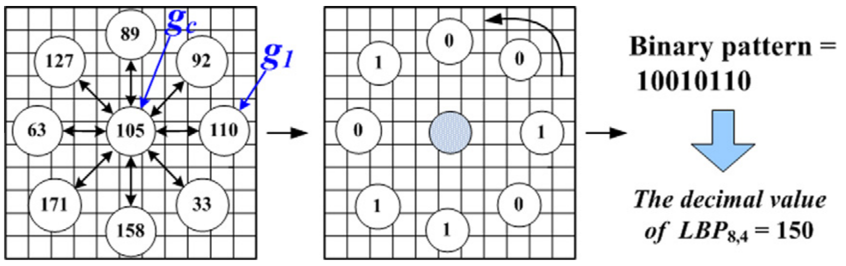

89
127
92
63
105
110
171
33
158
g_c
g_1
0
1
0
0
1
1
1
0
Binary pattern = 10010110
The decimal value of $LBP_{8,4}$ = 150

Fig. 6. A LBP computing example.

is because the geometrical properties and text textures of text candidates lack of the flexibility to verify text candidates. Hence we add another phase by using SVM to verify text candidates to increase the precision pate. In the phase two of the text candidate identification, we first extract the statistic feature of each text candidate by resizing each text candidate to the same 128 × 128 size by using bilinear interpolation as mention above. We then use Haar wavelet transform to decompose the text candidate to the four sub-band images including: low frequency (LL) band, vertical high frequency (LH) band, horizontal high frequency (HL) band and high frequency (HH) band. An example of Haar wavelet transformed image as shown in Fig. 7.

Next, we calculate the features in four sub-bands including mean, standard deviation and entropy of each sub-band as follows:

Mean:

$$\mu = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N} w(i,j) \tag{12}$$

Standard deviation:

$$\sigma = \sqrt{\frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}[w(i,j) - \mu]^2} \tag{13}$$

Entropy:

$$\text{Entropy} = -\sum_{i=1}^{M}\sum_{j=1}^{N} w(i,j)\log_2 w(i,j) \tag{14}$$

Using three features for four sub-bans each, we can then obtain 12 dimension features for each text candidate. In addition to these statistic features, we also compute the gray-level co-occurrence matrix (GLCM) which can describe the two dimensional spatial relations of the grayscale image (Gonzalez & Woods, 2008). First, four 10 × 10 GLCMs at directions 0°, 45°, 90°, 135° with the co-occurrence distance of one pixel are calculated for each candidate. An example of calculating GLCM at 45° direction is illustrated in Fig. 8.

Five features of the GLCM, energy, entropy, contrast, homogeneity and correlation, are calculated for each four direction in four wavelet sub-bands as follows:

Energy:

$$Eg(d,\theta) = \sum_i\sum_j C^2(i,j|d,\theta) \tag{15}$$

Entropy:

$$Et(d,\theta) = -\sum_i\sum_j C(i,j|d,\theta)\log_2 C(i,j|d,\theta) \tag{16}$$

Contrast:

$$I(d,\theta) = \sum_i\sum_j (i-j)^2 C(i,j|d,\theta) \tag{17}$$

Homogeneity:

$$H(d,\theta) = \sum_i\sum_j \frac{C(i,j|d,\theta)}{1+(i-j)^2} \tag{18}$$

Correlation:

$$R(d,\theta) = \sum_i\sum_j \frac{(i-\mu_x)(j-\mu_y)(C(i,j|d,\theta)}{\sigma_x\sigma_y} \tag{19}$$

where μ_x, μ_y and σ_x, σ_y are means and variances of the GLCM $C(i,j|d,\theta)$. θ is the direction of cooccurrence. d is the co-occurrence distance, which is set to be a pixel in the experiment. By calculating these five features in 16 GLCMs, we can get 80 GLCM features. Together with aforementioned mean, variance and entropy features, we can get a 92-dimension feature vector for each text candidate.

After the statistic features of each text candidate are extracted. A principal component analysis (PCA) reduced dimension method is performed to reduce directly the original 92-dimension statistic feature to a 36-dimension vector that can improve the performance significantly, and the 36-dimension reduced statistic feature will then be applied to the SVM as training and testing samples.

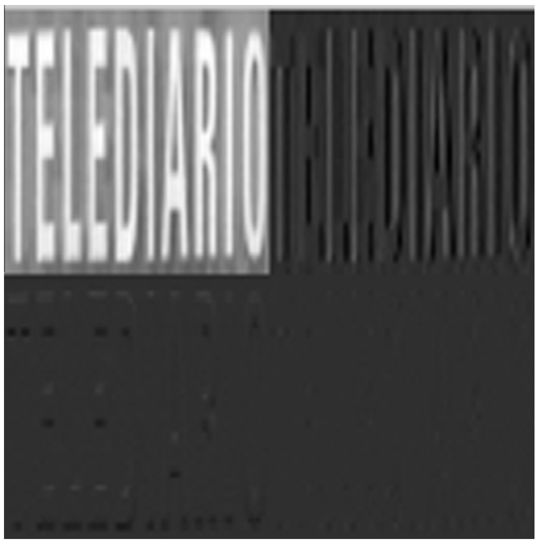


Fig. 7. Transformed image by Haar wavelet.

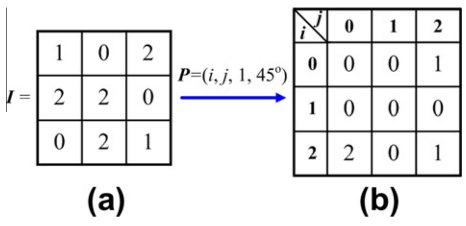


Fig. 8. An example to calculate GLCM at direction 45° and the co-occurrence distance = 1. (a) Image I is a 3 × 3 compressed gray level image. (b) GLCM at 45°.

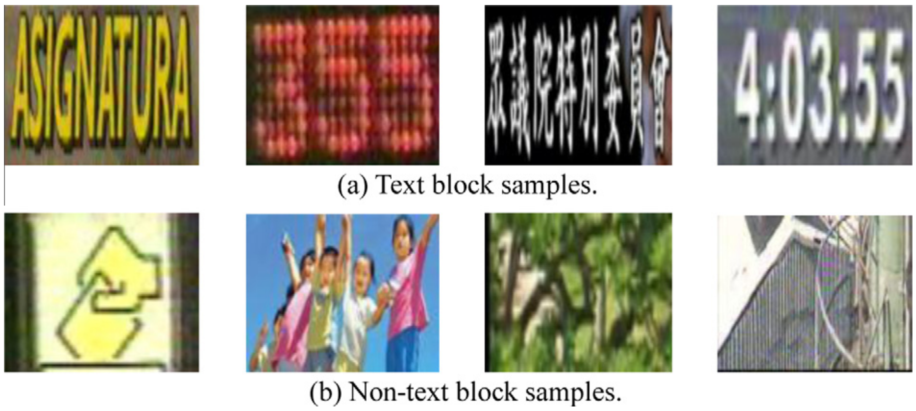


Fig. 9. Some SVM classifier training samples.

Table 1
Compare the recall rate of kernel functions.

Kernel function	Linear	Quadratic	RBF	Polynomial	MLP
Recall rate	88.2%	86.4%	100%	91.7%	87%

Table 2
Compare the execution time of kernel functions.

Kernel function	Linear	Quadratic	RBF	Polynomial	MLP
Execution time (sec)	1.6	1.1	1.1	1.2	1.1

For the SVM training, two training conditions should be selected.

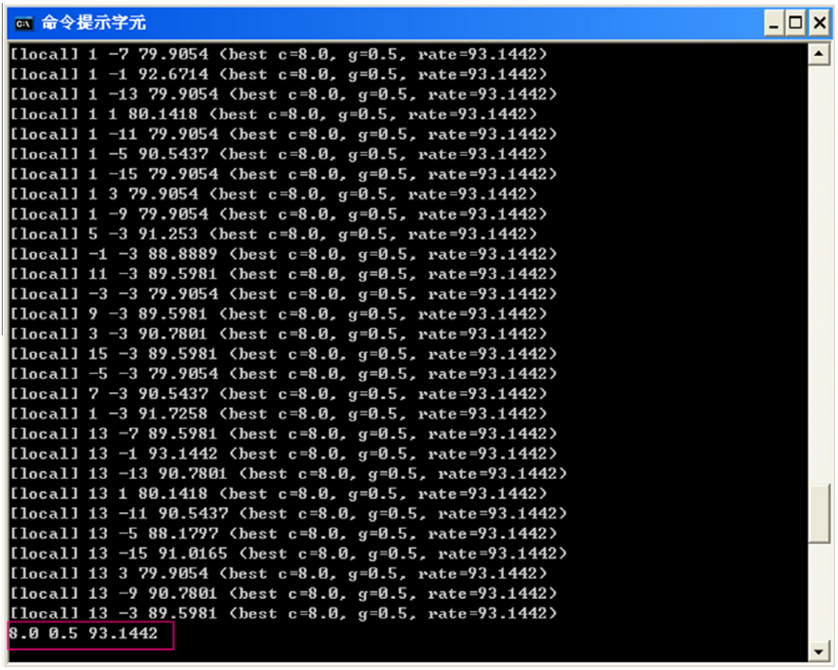
(1) For the optimization method, we choose the sequential minimal optimization (SMO) algorithm (Chang & Lin, 2011) to find the optimal values α and b^* that can obtain the maximum value of target function y for our SVM training, as follow:

$$y = \sum_{i=1}^{N} \alpha_i^* y_i(\vec{x}_i\vec{x}) + b^* \tag{20}$$

where y is the target function, $\vec{x}$ is SVM training samples and $\vec{x}_i$ is SVM training samples. N denotes the number of samples.

(1) For the kernel functions, SVM kernel function is to define a similarity measure on the basis of the dot product that transforms the data into feature space. Hence the inner product does not need to be evaluated in the feature space. The kernel function performs mapping of the data of the input space to the feature space. It plays a critical role in SVM and its performance. For the kernel function selection, we have chosen 150 images to test the recall rate and execution time of the SVM classifier. Some training samples are shown in Fig. 9. The recall rate is calculated as follows:
recall rate

$$= \frac{\text{The number of text blocks detecting in phase two}}{\text{The number of text candidate detecting in phase one}} \tag{21}$$



```
[local] 1 -7 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 -1 92.6714 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 -13 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 1 80.1418 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 -11 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 -5 90.5437 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 -15 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 3 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 -9 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 5 -3 91.253 (best c=8.0, g=0.5, rate=93.1442)
[local] -1 -3 88.8889 (best c=8.0, g=0.5, rate=93.1442)
[local] 11 -3 89.5981 (best c=8.0, g=0.5, rate=93.1442)
[local] -3 -3 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 9 -3 89.5981 (best c=8.0, g=0.5, rate=93.1442)
[local] 3 -3 90.7801 (best c=8.0, g=0.5, rate=93.1442)
[local] 15 -3 89.5981 (best c=8.0, g=0.5, rate=93.1442)
[local] -5 -3 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 7 -3 90.5437 (best c=8.0, g=0.5, rate=93.1442)
[local] 1 -3 91.7258 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -7 89.5981 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -1 93.1442 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -13 90.7801 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 1 80.1418 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -11 90.5437 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -5 88.1797 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -15 91.0165 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 3 79.9054 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -9 90.7801 (best c=8.0, g=0.5, rate=93.1442)
[local] 13 -3 89.5981 (best c=8.0, g=0.5, rate=93.1442)
8.0 0.5 93.1442
```

Fig. 10. Search the best γ and error cost coefficient C value.

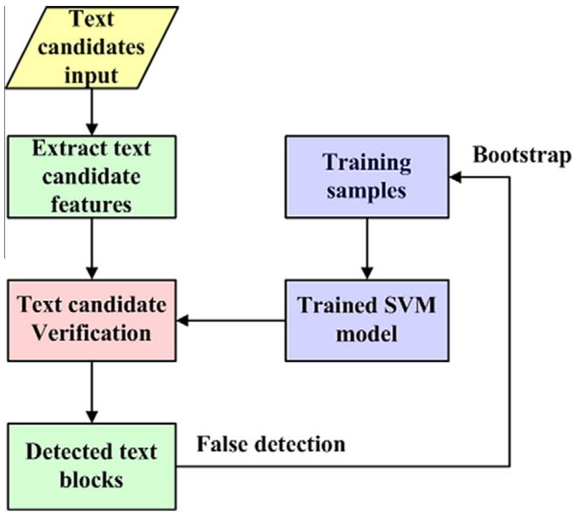


Fig. 11. Bootstrap procedure diagram.

According to the testing results, as shown in Tables 1 and 2, the Gaussian radial basis function (RBF) performs better than other kernel functions. Hence we choose RBF as our kernel function, which can map the data of the input space to the feature space without complicated inner product, as shown in Eq. (22).

$$K(x,y) = \exp\left(-\frac{\|x-y^2\|}{2\sigma^2}\right) \tag{22}$$

Before the 36-dimension reduced statistic features of the text candidates can be used to train the SVM classifier, they must be scaled first to avoid the different data range that may affect the justness of the classifier. We use the max–min linear scaling method to scale the 36-dimension data as follows:

$$val_{\text{new}} = \frac{val - \min}{\max - \min}(\max_{\text{new}} - \min_{\text{new}}) + \min_{\text{new}} \tag{23}$$

where *val* is the original feature value in the range $[\min, \max]$, val_{new} is the scaled feature value in the scaled range $[\min_{\text{new}}, \max_{\text{new}}]$, and in our experiments we choose $[-1, 1]$ as the scaled range.

To improve the RBF kernel function performance, we need to choose two parameters γ and the error cost coefficient C for the RBF kernel function to training the model. To select γ and the error cost coefficient C, we use a tool "grid.py", which is a python script running under DOS system (Chang & Lin, 2011) to get the best γ and error cost coefficient C value, as shown in Fig. 10.

After completing the SVM classifier training model, we adopt "bootstrap" strategy to implement the SVM classifier testing procedure. That is, in the detection process, false positive will be added into the training set for re-training. Then, the SVM classifier will perform better and better in the bootstrap procedure, the bootstrap procedure diagram is shown in Fig. 11.

4. Experimental results

To evaluate the performance of our method, we have chosen 200 images from TV news, web images, and movie clips for testing. The size of the test images is 288 × 352, and images with texts from different languages are included. Our method is implemented using MATLAB and run on a 1.6 GHz Pentium M processor with 1.5 GB main memory. The approximate processing time to detect the text lines for each video image is around 7 s for text detection. The parameter values used for the experiments are empirically determined as follows: (1) for the K-means clustering of the text region detection using pyramidal gradients step, we selected $N = 11$, (2) for the geometrical properties of the text candidate identification step, we chose $T_1 = 0.5$, $T_2 = 0.08$.

We have selected three existing methods (Liu et al., 2005; Mariano & Kasturi, 2000; Phan et al., 2009) for the performance comparison. Method (Phan et al., 2009) (denoted as *Laplacian method*) is based on the gradient which calculates by the Laplacian operator. Method (Liu et al., 2005) denoted as (*edge-based method*) uses edge features for K-means. Finally, method (Mariano & Kasturi, 2000) denoted as (*uniform-colored method*) adopts uniform color information for text detection.

4.1. Sample test results

Fig. 12 shows the text detection results of the proposed method in (f), and the results of the three existing methods stated above

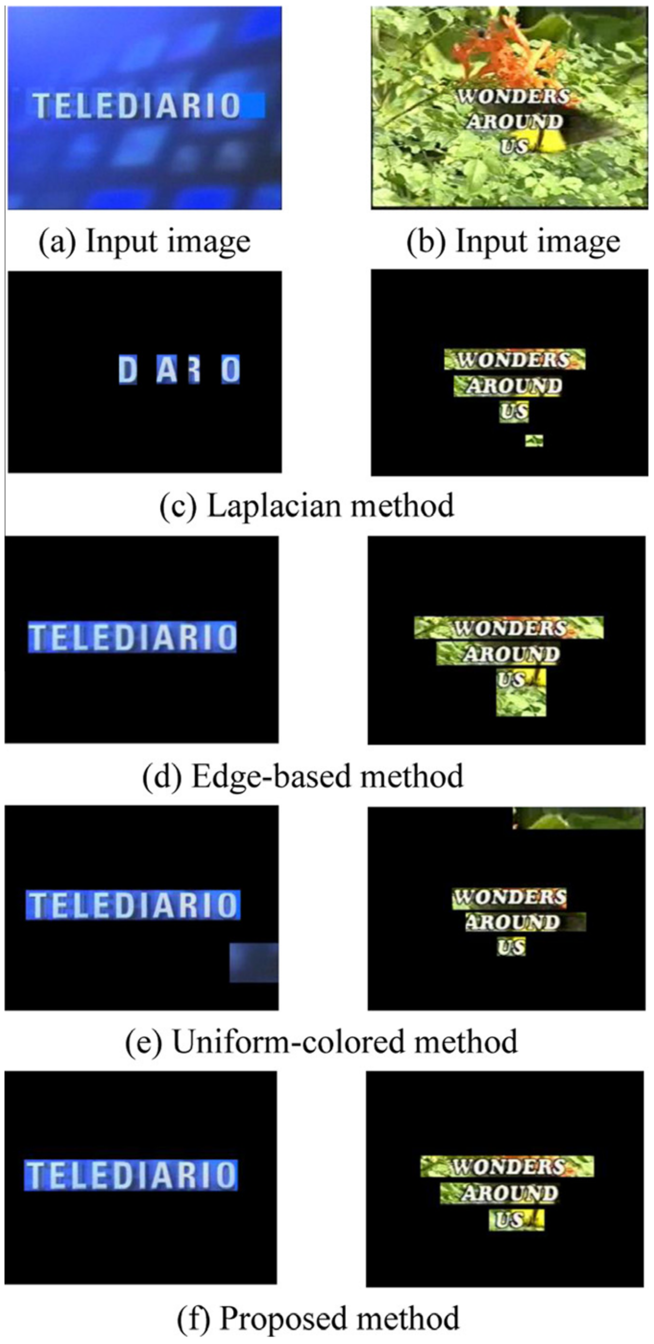


Fig. 12. The detected results for the input image (a) and (b).

are shown in (c)–(e). Image (a) has a simpler background and a bigger font size. The Laplacian method only detects some characters but not completely. The edge-based method detects the whole text line but it costs much time. The uniform-colored based method can detect the whole text line but it has a false positive. Proposed method detects the whole text line and is more efficient than the edge and uniform-colored based methods.

Image (b) has a more complicated background and three text lines. The Laplacian method can detect text lines completely; however, it also detects a false positive. The edge-based method detects all the text lines with much larger regions. The uniform-colored based method can detect all the text lines but with a false positive as well. Proposed method can detect all the text lines without false positives. The extra regions detected are smaller than the ones detected by the edge-based method as well.

Fig. 13 shows six images from different languages using the proposed method. It is clear that the proposed method can work for different languages. Fig. 14(a) shows where the proposed method fails to detect one text line ("25.2"), due to the very low contrast between the text and its background. Fig. 14(b) shows the situation that the proposed method can detect a low contrast scene text line correctly; however, under the detected text line, there is a low contrast text line, which is too small to be detected.

4.2. Comparison metrics

For comparison metrics, we use the following measurements to identify the performance of text detection methods.

- Actual Text Blocks (ATB): the total number of ground truth text in the test image dataset.
- Truly Detected Block (TDB): A detected block that contains a text line, partially or fully.
- False Detected Block (FDB): A detected block that does not contain text.
- Text Block with Missing Data (MDB): A detected block that misses some characters of a text line.

For each test image from the test image dataset, we count the above amounts and calculate the performance measurements defined as follows:

Definition 4.1. Detection Rate (DR) = TDB/ATB.

Definition 4.2. Precision Rate (PR) = TDB/(TDB + FDB).

Definition 4.3. Misdetection Rate (MDR) = MDB/TDB.

The performance comparison of the proposed method with the three existing methods is summarized in Table 3. For the testing dataset, the proposed method has the best PR and the lowest MDR and its PR will become better when the SVM Classifier bootstrap procedure is applied. The proposed method is also superior to the uniform-colored method in all the performance measurements. Compared to the Laplacian method, although its average processing time (APT) is better than the proposed method but it also has the highest MDR. This is because the Laplacian method is more suitable for dense text lines. When the text lines are sparse, the Laplacian method usually misses some characters. The edge-based method though has rather low MDR and the best DR but has the lowest PR and the highest APT.

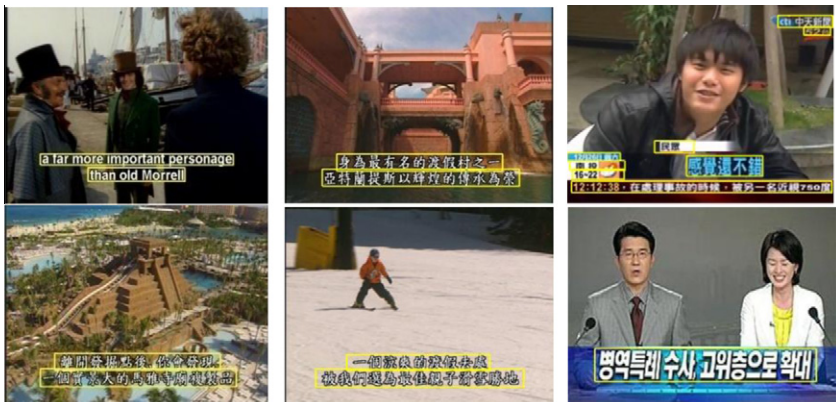

a far more important personage
than old Morrell
身為最有名的渡假村之一
亞特蘭提斯以驛站的傳水為榮
感覺還不錯
병역특례 수사 고위층으로 확대

Fig. 13. Text detection for different language images.

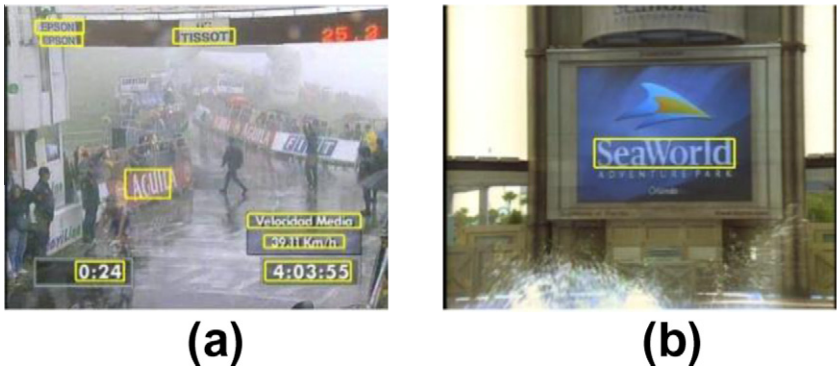


Fig. 14. Text detection for low contrast images.

Table 3
Performance of different methods.

Method	DR	PR	MDR	APT (sec)
Edge-based (Liu et al., 2005)	96.2%	72.3%	11.4%	69.7
Uniform-colored (Mariano & Kasturi, 2000)	68.5%	75.3%	27.6%	50.2
Laplacian (Phan et al., 2009)	93.8%	84.8%	35.9%	5.1
Proposed	95.1%	89.6%	5.2%	8.7

Table 4
Time complexity of different methods.

Method	Time complexity
Edge-based (Liu et al., 2005)	O(mnd)
Uniform-colored (Mariano & Kasturi, 2000)	O(mn(log(mn)))
Laplacian (Phan et al., 2009)	O(mn)
Proposed	O(mn)

Table 4 compares the proposed method with the three existing methods in terms of time complexity, where $m \times n$ denote the size of the image ($m \times n$). The edge-based method uses d-dimensional features to perform K-means clustering on the image. Therefore, it takes more time. The uniform-colored method applies hierarchical clustering to the color information of the whole image. It also takes more time. The Laplacian method and the proposed method have the same time complexity.

5. Conclusions and future work

In this article, a new method for detecting text regions in video image is proposed. The input images are downsized to form a three layer pyramid in order to detect fonts in different sizes. Three K-means clustering procedures are applied to identify the candidate regions. The results from the three layers are then combined and refined using the projection profile analysis. Finally, false positives are remove using geometrical and texture properties and SVM. Experimental results show that our method is robust for detecting text regions with various font sizes, colors, languages and background complexity. For our future work, we plan to combine methods, such as the one based on temporal information of video images or color detection methods, to reduce the number of false positives and to increase the rate of detection of text. For practical purpose, the proposed method can be improved by using a fuzzy like logic technique. Hence, the SVM classifier also has a correct detection when the text candidate may have changed slightly at different detecting time.

Acknowledgement

This work was supported by the National Science Council of Taiwan, ROC, under the Project No. 99-2218-E-011-001-.

References

Chang, C. -C., & Lin, C. -C. (2011). LIBSVM: A Library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Fernandez-Caballero, A., Lopez, M. T., & Castillo, J. C. (2012). Display text segmentation after learning best-fitted OCR binarization parameters. *Expert Systems with Applications, 39*(4), 4032–4043.

Gllavata, J., Ewerth, R., & Freisleben, B. (2004). Text detection in images based on unsupervised classification of high frequency wavelet coefficients. In *IEEE International Conference on Pattern Recognition*, 1, (pp. 425–428).

Gong, L., Zeng, J., & Zhang, S. (2011). Text stream clustering algorithm based on adaptive feature selection. *Expert Systems with Applications, 38*(3), 1393–1399.

Gonzalez, R. C., & Woods, R. E. (2008). *Digital image processing* (3rd ed.). Person Education.

Jung, C., Liu, Q., & Kim, J. (2009). A stroke filter and its application to text localization. *Pattern Recognition Letters, 30*(2), 114–122.

Kim, W. J., & Kim, C. G. (2009). A new approach for overlay text detection and extraction from complex video scene. *IEEE Transactions on Image Processing, 18*(2), 401–411.

Liu, C., Wang, C., & Dai, R. (2005). Text detection in images based on unsupervised classification of edge-based features. In *IEEE International Conference on Document Analysis and Recognition*, (pp. 610–614).

Liu, Q., Jung, C., Kim, S., Moon, Y., & Kim, J. (2006). Stroke filter for text localization in video images. In *IEEE International Conference on Image Processing*, (pp. 1473–1476).

Lyu, M. R., & Song, J. Q. (2005). A comprehensive method for multilingual video text detection, localization, and extraction. *IEEE Transactions on Circuits and System for Video Technology, 15*(2), 243–255.

Mariano, V. Y., & Kasturi, R. (2000). Locating uniform-colored text in video frames. In *IEEE International Conference on Pattern Recognition*, 4, (pp. 539–542).

Miao, G., Huang, Q., Jiang, S., & Gao, W. (2008). Coarse-to-fine video text detection. In *IEEE International Conference on Multimedia and Expo*, (pp. 569–572).

Ngo, C. W., & Chan, C. K. (2005). Video text detection and segmentation for optical character recognition. *Multimedia Systems, 10*(3), 261–272.

Phan, T. Q., Shivakumara, P., & Tan, C. L. (2009). A Laplacian method for video text detection. In *IEEE International Conference on Document Analysis and Recognition*, (09), (pp. 66–70).

Shivakumara, P., Phan, T. Q., & Tan, C. L. (2009a). Video text detection based on filters and edge features. In *IEEE International Conference on Multimedia and Expo*, (pp. 514–517).

Shivakumara, P., Phan, T. Q., & Tan, C. L. (2009b). A gradient difference based technique for video text detection. In *IEEE International Conference on Document Analysis and Recognition*, (09), (pp. 156–160).

Sun, L., Liu, G., Qian, X., & Gao, D. (2009). A Novel text detection and localization method based on corner response. In *IEEE International Conference on Multimedia and Expo*, (pp. 390–393).

Suzuki, K., Horiba, I., & Sugie, N. (2003). Linear-time connected-component labeling based on sequential local operations. *Source Computer Vision and Image Understanding Archive, 89*(1), 1–23.

Tang, X., Luo, B., Gao, X., Pissaloux, E., & Zhan, H. (2002). Video text extraction using temporal feature vectors. In *IEEE International Conference on Multimedia and Expo*, 1(02), (pp. 85–88).

Wang, X., Huang, L., & Liu, C. (2010). A video text location method based on background classification. *International Journal on Document Analysis and Recognition, 13*(3), 173–186.

Wong, E. K., & Chen, M. (2000). A robust algorithm for text extraction in color video. In *IEEE International Conference on Multimedia and Expo*, 2 (2000), (pp. 797–800).

Ye, Q. X., & Huang, Q. M. (2004). A new text detection algorithm in images/video frames. In *Advances in Multimedia Information Processing – PCM 2004*, 3332, (pp. 858–865).

Ye, Q., Gao, W., Wang, W., & Zeng, W. (2003). A robust text detection algorithm in images and video frames. In *Fourth IEEE Pacific-Rim Conference on Multimedia* 2, (pp. 802–806).

Yun, J., Jing, L., Yu, J., & Huang, H. (2010). A multi-layer text classification framework based on two-level. *Expert Systems with Applications, 39*(2), 2035–2046.

Zhang, J., & Kasturi, R. (2008). Extraction of text objects in video documents: Recent progress. In *Eighth IAPR Workshop on Document Analysis Systems*, (pp. 5–17).

Zhang, X., Sun, F. C., & Gu, L. (2010). A combined algorithm for video text extraction. *Fuzzy Systems and Knowledge Discovery, 5*(10), 2294–2298.