Multimedia Systems 10: 261–272 (2005)
Digital Object Identifier (DOI) 10.1007/s00530-004-0157-0

Multimedia Systems

Video text detection and segmentation for optical character recognition

Chong-Wah Ngo, Chi-Kwong Chan

Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon
(e-mail: {cwngo,ckchan}@cs.cityu.edu.hk)

Published online: 14 December 2004 – © Springer-Verlag 2004

Abstract. In this paper, we present approaches to detecting and segmenting text in videos. The proposed video-text-detection technique is capable of adaptively applying appropriate operators for video frames of different modalities by classifying the background complexities. Effective operators such as the repeated shifting operations are applied for the noise removal of images with high edge density. Meanwhile, a text-enhancement technique is used to highlight the text regions of low-contrast images. A coarse-to-fine projection technique is then employed to extract text lines from video frames. Experimental results indicate that the proposed text-detection approach is superior to the machine-learning-based (such as SVM and neural network), multiresolution-based, and DCT-based approaches in terms of detection and false-alarm rates. Besides text detection, a technique for text segmentation is also proposed based on adaptive thresholding. A commercial OCR package is then used to recognize the segmented foreground text. A satisfactory character-recognition rate is reported in our experiments.

Keywords: Video text detection – Text segmentation – Text recognition

1 Introduction

Due to the drastic advances in video technologies, effective indexing of video content plays an important role. Like color, texture, motion, and objects, text in videos offers valuable information for effective content organization of videos. The success in the detection, segmentation, and recognition of video text can have a great impact for multimedia applications like digital libraries [5], home video summarization [11], and lecture video indexing [14].

Extracting text information from videos generally involves three major steps:

- *Text detection*: Find the regions that contain text.
- *Text segmentation*: Segment text in the detected text regions. The result is usually a binary image for text recognition.
- *Text recognition*: Convert the text in the video frames into ASCII characters.

Text in videos is usually not easily extracted especially when it is embedded in complex background scenes and suffers from poor visual quality due to the effects of motion blur and compression artifacts.

Based on the adopted visual features, current techniques in video text detection can be broadly categorized into two major groups: geometric-based [11, 16] and texture-based [6, 9, 23] approaches. Geometric-based approaches apply conventional image-processing techniques to extract and model text by analyzing the geometric arrangement of edges or regions that belong to characters. Texture-based approaches, on the other hand, view video text as regions composed of special texture patterns. Low-level image features such as edge gradients [6], corners [6], and DCT [23] are used to model and detect the patterns that belong to textual information. Recently, supervised learning methods such as classifiers based on support vector machines [8] and neural networks [9, 13] have also been adopted for the classification of text and nontext regions.

The segmentation of foreground text and complex background scenes from video frames of low visual quality is a challenging problem. Typical techniques include adaptive thresholding [15, 17, 19, 21], clustering [20], and character extraction filter [18]. Adaptive thresholding techniques [15,17,19,21] normally decide a threshold value for image binarization based on the statistical information computed from a local window. These methods are generally sensitive to background scenes and will create noise if the background scene is not clean. Clustering approach [20] normally assumes that a detected text box consists of only two clusters (e.g., a foreground text color and a background color). Gaussian mixture model or color histogram is usually employed to characterize foreground and background colors. Nevertheless, these techniques usually fail especially for background scenes composed of multiple colors. The character extraction filter, proposed in [18], is designed specifically for newscast applications. It is basically composed of two steps: binarization and vertical projection profile. This approach is sensitive to text line rotation and could cause oversegmentation of characters. The character extraction filter is a good technique for segmenting superimposed captions in news videos but might not be effec-

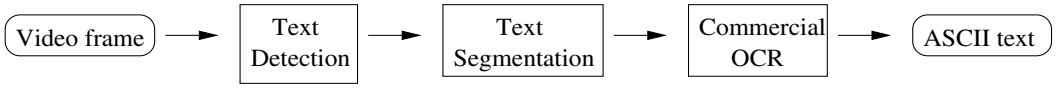


Fig. 1. Overview of our system

tive for segmenting scene text that appears in home videos or lecture videos.

Compared with text detection and segmentation, relatively few studies have been reported for video text recognition [1, 18, 22]. In fact, most approaches directly apply commercial OCRs for character recognition. To date, most commercial OCR systems can give excellent performance for high-resolution document images but perform poorly when used to recognize segmented text in videos. As reported in [1], only about 50% recognition accuracy is attained by commercial OCR applications. Several attempts have been made to improve recognition performance of commercial OCR applications. These efforts include (i) preprocessing prior to OCR, (ii) postprocessing after OCR, and (iii) development of a new OCR specifically for video text recognition [1,18,22]. Preprocessing normally involves techniques in image enhancement and multiframe integration [13, 18, 21]. Postprocessing takes into account the conceptual relationships among the recognized characters [1]. Some recognition errors can be corrected after postprocessing. Recently, the authors in [1, 18, 22] have also developed their own video OCR applications. The improvement over commercial OCR applications has also been reported in [1].

In this paper, we mainly address two issues: text detection[1] and segmentation. A commercial OCR application is also employed to show the effectiveness of our approaches. Figure 1 shows the flow of our system. For video text detection, we propose a texture-based approach. The novelty of our proposed approach is mainly based on its ability to apply appropriate operations for images of different complexity. For instance, text enhancement is applied to video frames with low contrast, while effective operators such as repeated left and right shifting are applied to remove background noise of video frames with high edge density. The proposed approach, in contrast to existing methods, can adaptively select and apply four sets of different operations depending on the background complexity. As a result, our approach can balance both the detection rate and the false-alarm rate. For text segmentation, our approach is based on adaptive thresholding. Compared with the techniques in [15,17,21], our approach is relatively simple yet effective and less sensitive to background noise.

The remaining of the paper is organized as follows. Section 2 presents our video-text-detection approach. Section 3 describes our proposed text-segmentation approach. Section 4 shows the experimental results, while Sect. 5 concludes the paper.

2 Video text detection

Figure 2 illustrates the overview of our video-text-detection approach. Initially, an edge image is computed for a video frame. A global threshold is set to suppress the nonedge pixels. The threshold is set as low as possible in order not to filter possible text regions. The video frames will be classified into four different cases according to their edge density. Scene-dependent analysis will then be carried out to adaptively apply appropriate operators for video frames of different cases. These operators are mainly for text region enhancement and noise removal. Two-level projection is applied to the resulting images to coarsely and finely segment the text lines into appropriate textboxes. Those textboxes will be further verified based on their vertical edge strength.

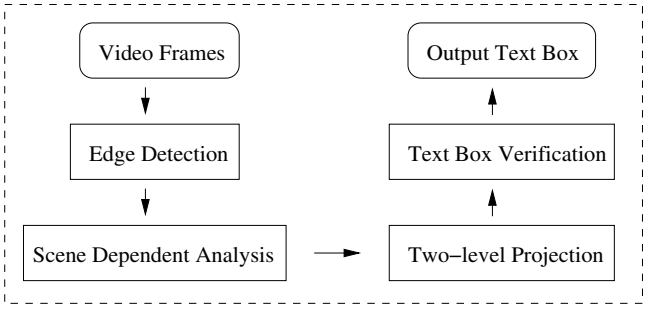


Fig. 2. Overview of our text-detection approach

2.1 Scene-dependent analysis

Textlike regions usually remain after edge detection. The main challenges in detecting text regions include: (1) distinguishing text regions from nontext regions that have high edge density and strength; (2) extracting low contrast text regions from simple background image. Conventional approaches for text detection usually rely on a single unified framework (e.g., edge smoothing and detection) to classify regions with high edge strength and density as text regions. One major deficiency of these approaches is that the scene complexity is not taken into account. For instance, when 2D smoothing is applied to a video frame with high edge strength in the background, both text and nontext regions will be equally smoothed and densified. This could lead to the text and nontext regions having indistinguishable visual properties. While one can utilize more sophisticated (with higher time complexity as well) approaches to specifically deal with video frames of high scene complexity, this could degrade the overall performance since a video frame with simple background can be readily detected by simple edge thresholding. A single unified framework that does not adapt to scene complexity can usually work effectively and efficiently for certain video frames, but not all. In this paper, we consider four different types of frames for video text detection: (1) simple background, (2) moderately complex background, (3) complex background, and (4) highly complex background. Because different operators are applied for video frames of different scene complexity (Table 2), we call our

[1] In this paper, we only analyze frames extracted from low-quality video; the temporal analysis of frames is explicitly excluded.

Table 1. Cases considered

Case	Description	Edge density
1	Simple background	$0\% \leq \mathbf{D} \leq 5\%$
2	Moderately complex background	$5\% < \mathbf{D} \leq 15\%$
3	Complex background	$15\% < \mathbf{D} \leq 30\%$
4	Highly complex background	$30\% < \mathbf{D} \leq 100\%$

Table 2. Operations applied to different cases

Case	2D Smooth	Left-right shift	Up-down shift
1	$\surd$	$\times$	$\times$
2	$\surd$	$\surd$	$\times$
3	$\times$	$\surd$	$\times$
4	$\times$	$\surd$	$\surd$

approach scene-dependent analysis. As indicated in Table 1, edge density is used as a measure for the indication of scene complexity. The edge density $\mathbf{D}$ of a frame is defined as

$$\mathbf{D} = \frac{\sum_{edge\ pixel}\{\text{Edge strength}\}}{\text{Total number of pixels}}. \tag{1}$$

The value of edge strength[2] ranges from $[0, 1]$. The quantization of $\mathbf{D}$ for different cases is obtained from the training set in Table 7. Figure 3 shows four different cases.

The superiority of scene-dependent analysis over four single independent approaches is empirically verified (one example is shown in Figs. 10 and 11). When the three approaches (cases 1, 2, and 3) are independently applied to a video frame of gradually increasing edge density (by adding uniform noise), they break down sharply at three different levels of scene complexity. The approach based in case 4 is capable of detecting text in video frames of different complexities; however, its performance is no better than the approach used in cases 1 and 2 when the background is not complex. Scene-dependent analysis, in contrast, can surpass the performance of the four different cases and yield satisfactory results.

2.1.1 Case 1: Simple background

The video frames in this case are mostly occupied by low-edge-density areas. To detect text regions that normally consist of higher edge density compared to background scenes, a 5×5 smoothing mask based on two 1D binomial filters [2] is employed as follows:

$$\mathcal{B}^4 = \frac{1}{16}\begin{bmatrix}1\,4\,6\,4\,1\end{bmatrix} * \frac{1}{16}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} = \frac{1}{256}\begin{bmatrix}1&4&6&4&1\\4&16&24&16&4\\6&24&36&24&6\\4&16&24&16&4\\1&4&6&4&1\end{bmatrix}, \tag{2}$$

where $*$ is convolution. A binomial filter ($\mathcal{B}^4$ in our case) has the nice properties that it approximates Gaussian distribution and involves mainly integer calculation to speed up convolution. With $\mathcal{B}^4$, edge strength in a low-edge-density region (probably noise) could be suppressed, while edge strength in a high-edge-density region could be enhanced.

2.1.2 Case 2: Moderately complex background

When the background is complex, the spatial cohesion of video characters is exploited based on two visual properties: edge strength and edge density. To distinguish text and nontext regions, besides applying $\mathcal{B}^4$, we also use two operators derived from the binomial filters to repeatedly shift and smooth the edge strength. Because the characters in text regions are usually upright and appeared in clusters before the operations, they normally remain in clusters after the operations. The edge strength of nontext regions, in contrast, will be gradually weakened and disappear during the shift and smooth operations.

The two operators, i.e., left shift and right shift operators, are formed by the following 1D binomial filters [2]:

$$\mathcal{L} = \frac{1}{2}\begin{bmatrix}\underline{1}\,1\end{bmatrix}, \mathcal{R} = \frac{1}{2}\begin{bmatrix}1\,\underline{1}\end{bmatrix}$$

which average the edge strength of its right or left neighboring pixel (the underlined number is the one where the average value is stored). The left shift ($\mathcal{L}^m$) and right shift operators ($\mathcal{R}^m$) are, respectively,

$$\mathcal{L}^m = \underbrace{\mathcal{L} * \mathcal{L} * \ldots * \mathcal{L}}_{\text{for } m \text{ times}}, \tag{4}$$

$$\mathcal{R}^m = \underbrace{\mathcal{R} * \mathcal{R} * \ldots * \mathcal{R}}_{\text{for } m \text{ times}}, \tag{5}$$

where $*$ represents convolution. In $\mathcal{L}^m$ (or $\mathcal{R}^m$), the average edge strength of m neighboring pixels is accumulated to the left (or right). For instance, when $m = 4$, $\mathcal{L}^m = \frac{1}{16}[\underline{1}\,4\,6\,4\,1]$ and $\mathcal{R}^m = \frac{1}{16}[1\,4\,6\,4\,\underline{1}]$. Since the text regions have high edge density, the regions can usually survive as clusters after applying $\mathcal{L}^m$. Figure 4c shows one example. When $m = 35$, the text lines appear as several smooth and dense clusters. In contrast, the nontext regions with lower edge density (e.g., the clusters formed by the edge strength of background regions and facial features) are dispersed. To restore the original positions of edge pixels, $\mathcal{R}^m$ is applied; in the meantime, this further weakens the regions with low edge density and strength, as shown in Fig. 4d.

[2] The edge strength, $\mathcal{S}'$, of a pixel (P_5) is computed as

$$\mathcal{S} = \max\{|P_1 - P_9|, |P_2 - P_8|, |P_3 - P_7|, |P_4 - P_6|\}$$

$$\mathcal{S}' = \begin{cases}\frac{\mathcal{S}}{256} & \mathcal{S} \geq 60\\ 0 & \text{Otherwise}\end{cases}$$

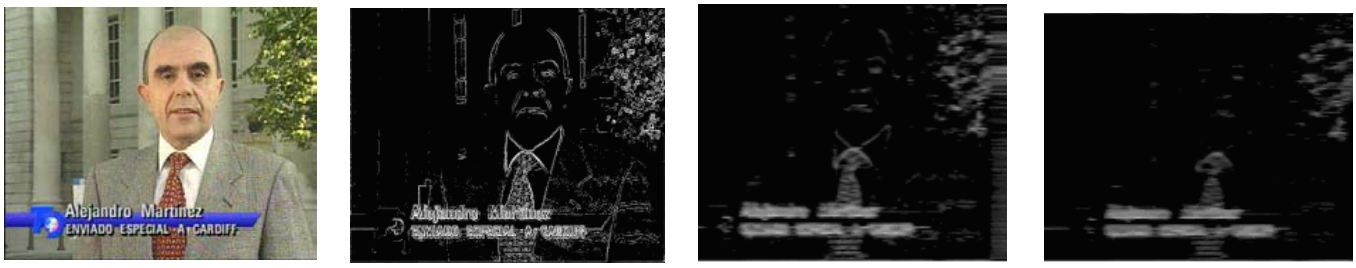
P_1	P_2	P_3
P_4	$\underline{P_5}$	P_6
P_7	P_8	P_9

Neighborhood configuration

a **b** **c** **d**

e **f** **g** **h**

Fig. 3a–h. Sample frames for different cases. **a** Sample 1. **b** Sample 2. **c** Sample 3. **d** Sample 4. **e** Case 1: $\mathbf{D} = 4.32\%$. **f** Case 2: $\mathbf{D} = 10.39\%$. **g** Case 3: $\mathbf{D} = 19.21\%$. **h** Case 4: $\mathbf{D} = 46.25\%$



a **b** **c** **d**

Fig. 4a–d. Results of left and right shifting operations for a video frame. **a** Video frame. **b** Edge detection. **c** Left shifting (35 times). **d** Right shifting (35 times)

The value of m, intuitively, should be directly proportional to the edge density of a frame, i.e., $m = \alpha \times 100 \times \mathbf{D}$, where $\alpha > 0$ is a parameter. In case 2, because the range of $\mathbf{D}$ is relatively narrow (10%), we simply set $m = 35$ (the value is empirically obtained from the training set in Table 7) for all images in case 2. As a result, both $\mathcal{L}^m$ and $\mathcal{R}^m$ have 72 taps. $\mathcal{L}^m$ and $\mathcal{R}^m$ can be implemented efficiently by repeatedly applying $\mathcal{L}$ m times and then repeatedly applying $\mathcal{R}$ another m times. In total, only $2 \times m$ averaging is needed. Since the filter support (72 taps) is large, it would be more efficient to implement both operators in the Fourier domain.

2.1.3 Case 3: Complex background

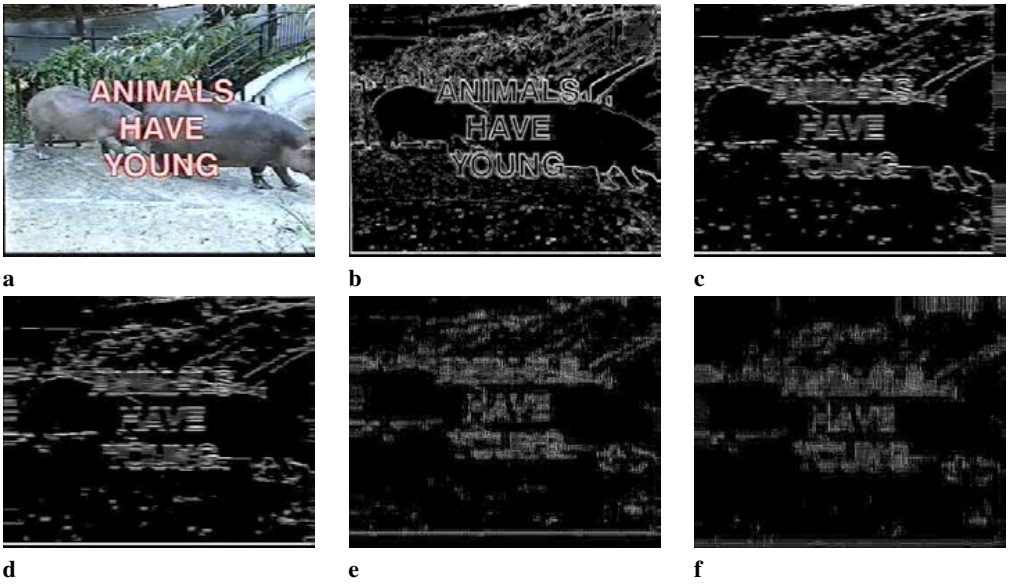
In case 3, the operations involved are similar to those in case 2 except that $\mathcal{B}^4$ is not applied for 2D smoothing. The main reason is that the smoothing will impact the effectiveness of the $\mathcal{L}^m$ and $\mathcal{R}^m$ operators if the edge strength of nontext regions is enhanced by $\mathcal{B}^4$. Figure 4 shows the steps in processing one video frame in case 3. As shown in the figure, most edges in the background are weakened or eliminated after applying $\mathcal{L}^m$ and $\mathcal{R}^m$ ($m = 35$).

2.1.4 Case 4: Highly complex background

When the background is highly complex, both text and nontext regions have high edge strength and density. The operations for the removal of nontext regions may remove the edges of text regions as well. Similarly, the operations for the enhancement of text regions may also enhance the edges of nontext regions. To effectively distinguish text and nontext regions, we use two more operators to repeatedly shift the edges in four different directions (left, right, up, down). Only clusters that are dense in both horizontal and vertical directions can survive after these operations. In addition to $\mathcal{L}^m$ and $\mathcal{R}^m$ operators, the up and down shifting operators are applied. These two operators are formed by the following 1D binomial filters: [2]

$$\mathcal{U} = \frac{1}{2}\begin{bmatrix}1\\ \underline{1}\end{bmatrix}, \mathcal{D} = \frac{1}{2}\begin{bmatrix}1\\ \underline{1}\end{bmatrix}$$

which average the edge strength of its down or up neighboring pixel. The up shift ($\mathcal{U}^n$) and down shift operators ($\mathcal{D}^n$) are, respectively,



a **b** **c**

d **e** **f**

Fig. 5a–f. Results of applying shifting operations. **a** Video frame. **b** Edge image. **c** Left shifting. **d** Right shifting. **e** Up shifting. **f** Down shifting

$$\mathcal{U}^n = \underbrace{\mathcal{U} * \mathcal{U} * \ldots * \mathcal{U}}_{\text{for } n \text{ times}}, \tag{7}$$

$$\mathcal{D}^n = \underbrace{\mathcal{D} * \mathcal{D} * \ldots * \mathcal{D}}_{\text{for } n \text{ times}}, \tag{8}$$

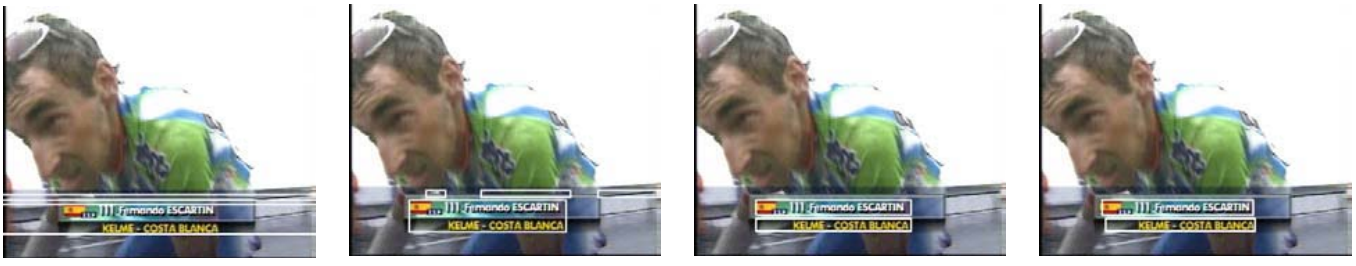
where $*$ is convolution. The values of m and n are dependent on the edge density $\mathbf{D}$. For an image with higher edge density, more shifting iterations are required in order to remove the background edges. In our experiment, $m = n = 80 \times \mathbf{D}$ is obtained from the training set in Table 7. Figure 5 illustrates the results of applying four different operators on an edge image of high density. The final resulting image is shown in Fig. 5f. After the shifting operations, the edge strength of nontext regions is significantly removed, while the text regions are blurred but can still be detected by our approach. One side effect of shift operators is that when the width and height of a text line are smaller than the number of operations (m and n), the text line may not be detected.

2.2 Two-level projection

The objective of projection is to decompose a video frame into text and nontext regions. A text region will be bounded in a rectangular area called a text box. The projection is divided into two phrases, namely, coarse projection and fine projection. Coarse projection will detect text boxes, but each text box may contain more than one text line. Fine projection will further decompose those text boxes that contain more than one text line. In each phrase, both horizontal and vertical projections will be performed to horizontally and vertically split a region into the text and nontext regions. The edge strength and edge density of each row and each column are used as the criteria to determine whether a region should be partitioned. Our approach is similar to that of [3, 13]. Figure 6 illustrates the process of two-level projection on one video frame. In panel a, two text lines are detected by the coarse horizontal projection. The text lines are coarsely segmented into four regions after the vertical projection, as shown in panel b. The horizontal projection is further applied to finely segment the four regions. Some regions are removed as they are partitioned into small segments, which could not be processed by OCR software. The resulting text lines are shown in panel c. Finally, the vertical projection is applied to finely adjust the bounding boxes of text lines as in panel d.

2.3 Text box verification

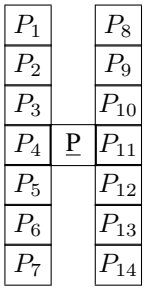
Intuitively, a text box should have high vertical edge strength. To check the validity of a text box, a vertical edge detector



a **b** **c** **d**

Fig. 6a–d. Coarse-to-fine horizontal and vertical projection. **a** Coarse horizontal. **b** Coarse vertical. **c** Fine horizontal. **d** Fine vertical

as shown below is applied to every pixel P in the potential textbox.



The new value of P is:

$$P = \frac{1}{7} \times \left| \sum_{i=1}^{7} P_i - \sum_{i=8}^{14} P_i \right|.$$

The values of all P are summed and normalized by the size of the text box. A text box will be retained for text segmentation and recognition if its total value exceeds a predefined threshold.

3 Text segmentation

Since we employ a commercial OCR package [24] for recognition, the first step is to segment all the detected text boxes into foreground (text) and background images prior to recognition. One of the best segmentation (or binarization) algorithms is known as Niblack's algorithm [15], as indicated in the survey [19]. Niblack's algorithm adopts local adaptive thresholding for text binarization. The threshold is adaptively obtained from the local statistics by sliding a window across an image. This algorithm can nevertheless create noise, especially in a complex background scene. Since most commercial OCR packages are sensitive to background noise, we do not adopt Niblack's algorithm for text segmentation. Instead, we propose another adaptive thresholding approach that is capable of reducing most noise caused by complex background scenes. In our approach, a threshold is adaptively determined based on the global statistic of a text box.

For each detected text box, a 16-bin normalized histogram is computed based on the gray-level values of pixels. Assuming that the text box contains inverse text (i.e., bright text on dark background), the proposed adaptive thresholding approach finds the first peak in the histogram by scanning the bins backward (i.e., the scanning starts with the last bin until the first maximal value is encountered).[3] The first valley (local minimum point) in front of the peak is then located. The bin, k, corresponds to this valley is used as a hint for global thresholding. The threshold value is set to $16 \times (k-1)$, where k is the bin number (or level) and 16 is the size of the bin. If an image contains normal text (i.e., dark text on a bright background), the statistical method described in [13] is used to invert the text box first.[4]

The intuition behind this algorithm is to locate a suitable value for binarization based on the global distribution of pixel values in a text box. Since the color of foreground text is normally similar, the first peak that is located ideally should correspond to the major text color. By using the threshold value that is computed based on the first minimum point in front of the peak for binarization, most of the regions that belong to background scenes can be removed.

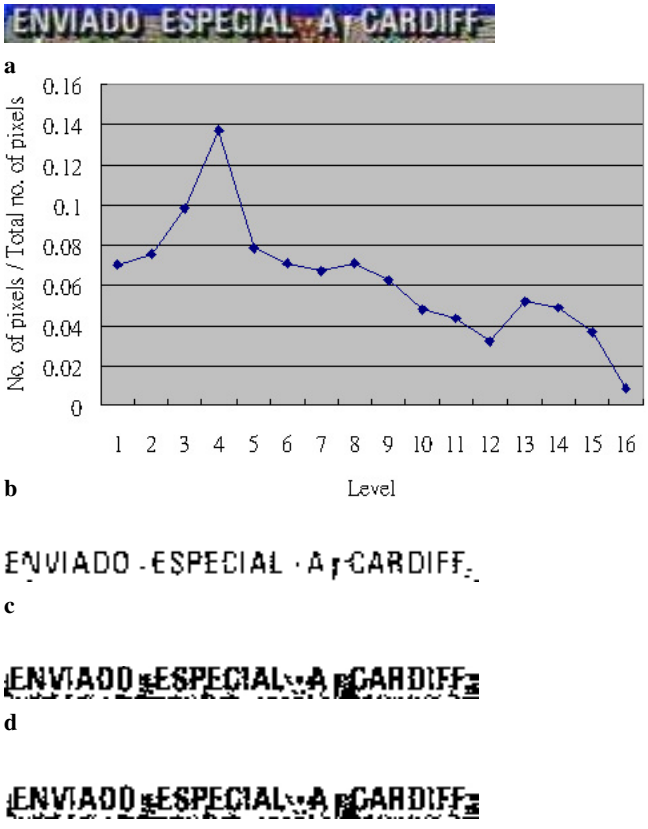


Fig. 7a–e. Results of text segmentation by different approaches. **a** Detected text box. **b** Normalized histogram. **c** Text segmentation by our approach. **d** Text segmentation by Niblack's method. **e** Text segmentation by Wolf's method

Table 3. OCR results of the text box in Fig. 7a with different text-segmentation methods

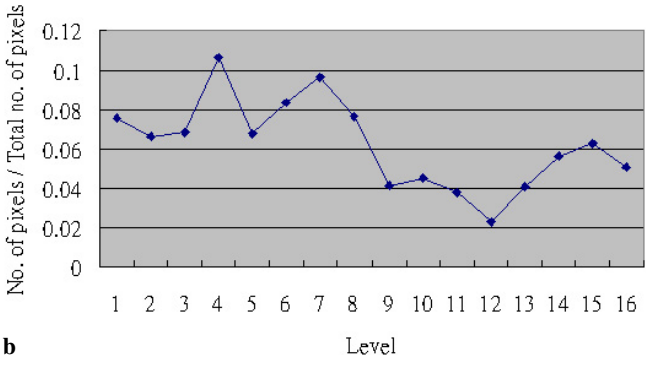
Method	Correct characters	Incorrect characters	OCR total output characters
Our approach	21	5	26
Niblack [15]	0	0	0
C. Wolf [21]	12	12	24

To illustrate the effectiveness of our approach, Fig. 7 shows the results of segmenting an extracted text box. In Fig. 7b, the normalized histogram of the text box in Fig. 7a is shown. The first detected peak is bin 13, while the first valley that is encountered after detection of the peak is bin 12. As a result, the value 176 ($(12-1) \times 16$) is used for segmenting the foreground text and background scene. To show the effectiveness, we compare the result with two other approaches – Niblack's method [15] and Wolf's method [21]. The results are illustrated in Fig. 7. The approach proposed by Wolf [21] is basically an improved version of Niblack's approach [15]. Since the major drawback of Niblack's approach is that noise can be created in background scenes, Wolf added new hypotheses on the gray level of foreground text and background scenes to tackle this problem. The results indicated in Fig. 7 show that the segmented text by our approach is not as noisy as by the other

[3] To ensure robustness, the maximal value is required to contain at least 5% of total pixels in the textbox.

[4] The method for estimating the text color and background color of text regions is described on page 262 of [13].

a



b

c

d

e

Fig. 8a–e. Results of text segmentation. **a** Detected text box. **b** Normalized histogram. **c** Our approach. **d** Niblack's method. **e** Wolf's method

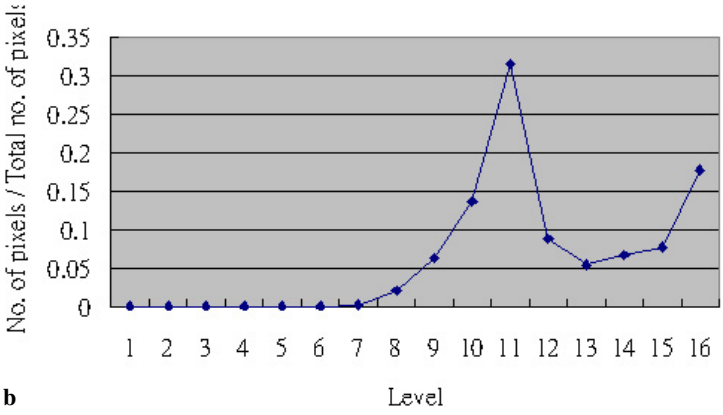
Table 4. OCR results of the text boxes in Figs. 8 and 9

	Fig. 8		Fig. 9	
Method	Correct	Incorrect	Correct	Incorrect
Our approach	13	4	15	1
Niblack [15]	2	12	14	2
C. Wolf [21]	7	5	16	0

two approaches. To further test whether the segmented text is appropriate for recognition, we use the commercial OCR package from [24] for character recognition. The results are shown in Table 3. As indicated in this table, our approach obtains the highest recognition rate.

Two more examples are shown in Figs. 8 and 9. In Fig. 8, the background is composed of three major colors, and the first valley that is located for thresholding is at bin 12 in the normalized histogram. Compared with Niblack's and Wolf's approaches, our approach achieves better segmentation quality since a significant amount of background noise is removed. Figure 9 shows the results of segmenting a scene text. The scene text looks blurred and unclear compared with normal artificial text. By our approach, the first valley located in the histogram is bin 13. In this example, the results of segmentation are similar for the three tested approaches. Table 4 shows the characters that are correctly and incorrectly recognized by the commercial OCR package. When the background is complex (as in Figs. 7 and 8), our approach usually yields much better results.

a



b

c

d

e

Fig. 9a–e. Results of text segmentation. **a** Detected text box. **b** Normalized histogram. **c** Our approach. **d** Niblack's method. **e** Wolf's method

4 Experiments

4.1 Groundtruth data

Three datasets that contain English and Chinese characters are used for our experiments. The first dataset, which is obtained from [6], contains 45 video frames, the second dataset contains 68 frames, while the last dataset contains 93 frames. All the frames are extracted from MPEG1 videos. The size of each frame is 352×288. The embedded video text includes the superimposed captions and scene text. Some video frames are rich in texture but do not contain any text. The groundtruth text regions are manually labeled by human subjects. We use the first dataset as the training set. All the parameters used in our approach are empirically determined from this dataset.

4.2 Performance evaluation for video text detection

The performance evaluation is based on the protocol proposed by [6,7]. This protocol takes into account detection difficulty such as image contrast, background complexity, and string density. The performance is measured by the total detection index (TDI), which is defined as

$$TDI = \beta D + (1-\beta)(1-F)\,, \tag{9}$$

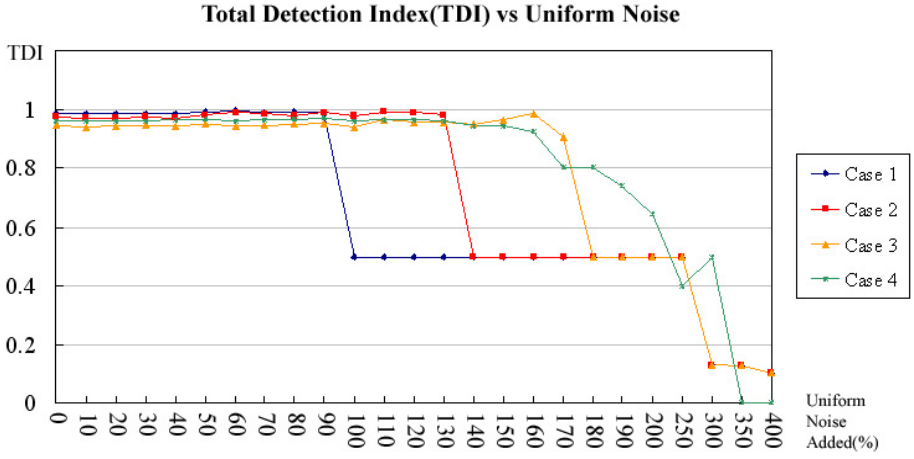


Fig. 10. Comparison of different independent cases when uniform noise is added

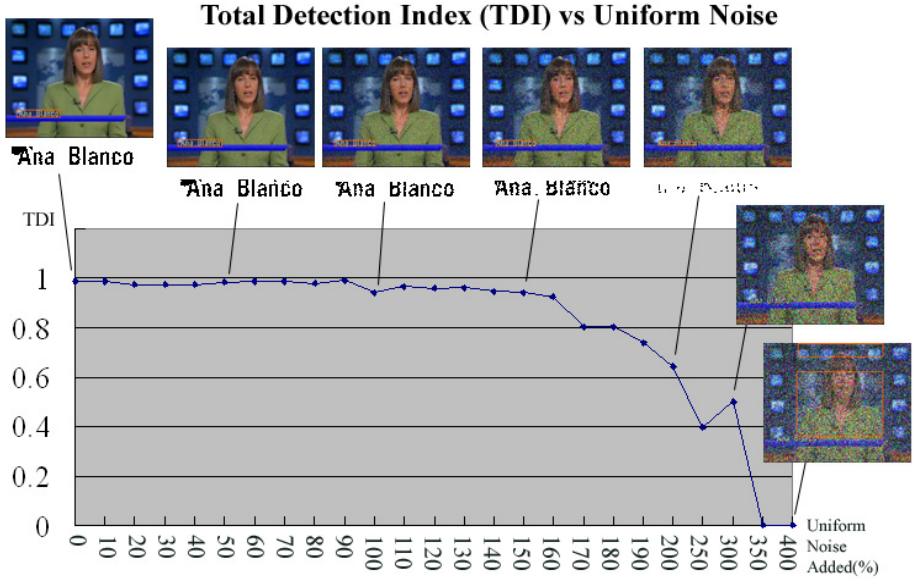


Fig. 11. Performance of scene-dependent analysis when uniform noise is added

where D is the detection rate and F is the false-alarm rate. The weight β is set to 0.5 in our experiments. The detection rate is jointly determined by the degree of overlap between the detected and groundtruth text boxes, the fragmentation quality,[5] and the detection difficulty. The false-alarm rate is determined in a similar way except that, instead of the overlapped regions, the falsely detected (nonoverlapped) regions are taken into account. When a detected text box includes more than one groundtruth text line, the detection rate will be lower while the false-alarm rate will be higher compared with the case when multiple groundtruth text lines are detected separately.

4.3 Effectiveness of scene-dependent analysis

To demonstrate the effectiveness of the proposed approach, we apply the operators for four different cases independently to a video frame with gradually increasing edge density by adding uniform noise. The experimental results are shown in Fig. 10 and Table 5. In Fig. 10, it is worth noting that the TDIs of the four case operators drop sharply at different levels of scene complexity (reflected by the percentage of added noise). Furthermore, the TDI of case 1 operators drops the earliest, followed by case 2, 3, and 4 operators. Even though the performance of case 4 operators is the best overall, its TDI is no better than the simpler approach, such as case 1 operators when the edge density is low.

Figure 11 shows the performance of scene-dependent analysis where different case operators are applied adaptively depending on scene complexity. The detected and segmented video text is also shown in the figure. When the added noise reaches 200% (**D** is approximately 65%), the detected text box

Table 5. Effectiveness of scene-dependent analysis when uniform noise is added

Approach	D	F	TDI
Case 1	0.419	0.119	0.650
Case 2	0.573	0.124	0.724
Case 3	0.700	0.143	0.779
Case 4	0.764	0.147	0.808
Scene-dependent	0.773	0.144	0.814

[5] For the detection rate, the fragmentation quality reflects the extent to which a groundtruth text box is split by multiple detected textboxes. For the false-alarm rate, the fragmentation quality reflects the extent to which a detected text box includes multiple groundtruth text boxes.

Table 6. Performance of scene-dependent analysis on dataset 1

Approach	D	F	TDI
Case 1	0.808	0.228	0.789
Case 2	0.862	0.206	0.828
Case 3	0.843	0.123	0.860
Case 4	0.777	0.145	0.815
Scene-dependent	0.915	0.109	0.903

Table 7. Results of video text detection for dataset 1

Approach	D	F	TDI
Scene dependent	**0.915**	**0.109**	**0.903**
SVM	0.909	0.071	0.919
Neural network	0.858	0.107	0.876
Corner and edge [6]	0.890	0.087	0.902
Multiresolution	0.912	0.125	0.893
DCT-based	0.674	0.219	0.728

cannot be correctly segmented.[6] While if the added noise is above 200%, no text line can be correctly detected. The overall performance of scene-dependent analysis is given in Table 5. As indicated in the table, the proposed scene-dependent detector can comprise different cases and yield the best performance. We conduct another test on the training set (dataset 1) where the scene-dependent detector is compared with four different case operators. The experimental results, as indicated in Table 6, show that the proposed approach obtains the highest TDI.

4.4 Performance comparison of different video text detectors

We compare scene-dependent analysis with four different existing approaches based on: support vector machines (SVM) [8], neural networks (NN) [9], multiresolution approaches [4], and compressed features (DCT) [23]. For SVM and NN, a total of 36 wavelet features are extracted from each 16×16 image block for classification. These features represent the mean, second-order, and third-order center moments of the subband images. The details of implementation can be found in [9]. In SVM, a Gaussian kernel is used while iterative training is employed to train a two-class classifier.[7] In NN, a multilayer neural network with twelve hidden nodes [9] are trained for classification.[8] For the multiresolution-based approach, three levels of pyramid edges are applied to detect text with large and small fonts. For the DCT-based approach, a total of 11 AC coefficients from each DCT block are used to characterize the horizontal and vertical spatial intensities of a DCT block for text detection [23].

Dataset 1, which is obtained from [6], is used as the training set for SVM and NN. Each video frame in the dataset is divided into 16×16 text and nontext blocks. We use all the text blocks from dataset 1, while randomly selecting the equal number of nontext blocks for training. For NN, we further employ a bootstrap method as in [9] to iteratively add the new nontext blocks that are incorrectly classified to train the NN until it converges. The training dataset is also used by the multiresolution-based, DCT-based, and the proposed scene-dependent analysis for parameter settings. The last two datasets are used as testing data to evaluate the effectiveness of the five different approaches. In total, there are 45 video frames (169 text lines) used for training and 161 frames (696 text lines) for testing.

[6] The percentage of added noise is defined as power(noise)/power(image).

[7] The implementation of SVM is based on LIBSVM at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[8] The implementation of NN is based on LNKnet at `http://www.ll.mit.edu/IST/lnknet/`.

Table 8. Results of video text detection for dataset 2 (English characters)

Approach	D	F	TDI
Scene-dependent	**0.887**	**0.140**	**0.874**
SVM	0.716	0.142	0.787
Neural network	0.640	0.144	0.784
Multiresolution	0.759	0.197	0.781
DCT-based	0.591	0.321	0.635

Table 9. Results of video text detection for dataset 3 (Chinese characters)

Approach	D	F	TDI
Scene-dependent	**0.836**	**0.225**	**0.805**
SVM	0.660	0.240	0.713
Neural network	0.625	0.240	0.694
Multiresolution	0.576	0.366	0.605
DCT-based	0.490	0.428	0.531

Table 7 shows the performance of various approaches on the training dataset. In this experiment, the parameters used in different approaches are systematically adjusted so as to achieve the best TDI. Compared with the experimental results given by [6], which utilizes edge and corner visual hints for detection, our proposed approach achieves better detection rate and TDI. Overall, SVM achieves the highest TDI and the lowest false-alarm rate, while the multiresolution-based method and our approach achieve the highest detection rate.

Based on the trained classifiers and tuned parameters obtained from the training set, Tables 8 and 9 further show the experimental results of different approaches on the two testing datasets. Dataset 2 contains only English characters, while dataset 3 contains only Chinese characters. In general, the detection difficulties of the video frames in these two datasets are higher than in the training set. Certain video frames suffer from decoding noise, low contrast, and poor resolution. As indicated in Tables 8 and 9, the performance of our proposed approach is superior to the four other tested approaches in terms of detection rate, false alarm rate, and TDI. By taking into account the background complexity as the criterion for selecting appropriate image operations, our approach can outperform the four other approaches.

Figure 12 shows the results of detection by different approaches on one sample video frame. The SVM-, NN-, and multiresolution-based approaches cause false alarms by detecting background scenes with rich texture as text. THe DCT-based method, on the other hand, cannot precisely locate text lines. This is mainly due to the fact that the DCT-based method operates directly on the block level. The located text box is normally either larger or smaller than the correct text lines.

Fig. 12a–f. Results of text detection by different approaches. **a** Original video frame. **b** Scene-dependent. **c** SVM-based. **d** Neural-network-based. **e** Multiresolution-based. **f** DCT-based

Fig. 13. Experimental results of scene-dependent analysis

Table 10. Results of text detection in Fig. 12

Approach	D	F	TDI
Scene-dependent	**0.954**	**0.097**	**0.928**
SVM	0.949	0.280	0.834
Neural network	0.953	0.215	0.869
Multiresolution	0.939	0.220	0.859
DCT-based	0.854	0.144	0.855

In fact, this problem also happens in SVM and NN since the classification decision is made on features extracted from image blocks. Among all the tested approaches, scene-dependent analysis achieves the best results, as indicated in Table 10. Figure 13 further shows more examples of text detection by our proposed approach. Currently, the computational time of our approach for a single frame is on average 1.13 s on a Pentium IV platform.

4.5 Evaluation of text segmentation and recognition

We employ a commercial OCR package [24] for character recognition. This OCR package is designed for recognizing text in high-resolution binary document images. In this experiment, the performance of text segmentation is evaluated by assessing the recognition rate of the segmented text in all the detected text boxes.

Figure 14 shows the results of our text-segmentation approach on various kinds of text boxes. Figure 14b–e make it clear that correctly segmenting foreground text and background scenes is a difficult task; nevertheless our approach still achieves reasonably good results. Obviously, the effectiveness of text segmentation can affect considerably the performance of OCR. The performance of text segmentation is evaluated based on the results of character recognition by

Fig. 14. Text segmentation on different sample types. **a** Clean background and inverse text. **b** Complex background. **c** Similar color in foreground text and background. **d** Nonwhite text (*yellow*). **e** Normal text (dark text on bright background)

Table 11. OCR results for dataset 1

Approach	Correct characters	Incorrect characters	OCR total output characters	Precision	Recall
Our approach	**1066**	**395**	**1461**	**0.729**	**0.719**
Niblack [15]	904	504	1408	0.642	0.610
Wolf [21]	1027	364	1391	0.738	0.693

Table 12. OCR results for dataset 2

Approach	Correct characters	Incorrect characters	OCR total output characters	Precision	Recall
Our approach	**1869**	**724**	**2593**	**0.721**	**0.737**
Niblack [15]	1423	1064	2469	0.576	0.561
Wolf [21]	1760	800	2560	0.688	0.694

$$\text{Recall} = \frac{\text{Number of correctly recognized characters}}{\text{Number of characters in dataset}}$$

$$\text{Precision} = \frac{\text{Number of correctly recognized characters}}{\text{Number of characters output by OCR}}$$

We compare our approach with the methods proposed by Niblack [15] and Wolf [21]. Tables 11 and 12 show the experimental results of OCR for datasets 1 and 2. The results of the last dataset are not given since the OCR package cannot recognize Chinese characters. For the first dataset, based on our approach, the OCR recognizes 1066 out of 1481 characters (recall = 71.9% and precision = 72.9%). For the second dataset, by our approach 1869 characters out of 2536 characters (recall = 73.7% and precision = 72.1%) are correctly recognized. Compared with the methods of Niblack and Wolf, our proposed approach achieves the best recall for both datasets and the best precision for the second dataset.

5 Conclusion

We have presented our proposed approaches to video text detection and segmentation. For text detection, experimental results indicate that the proposed scene-dependent analysis can balance both the detection rate and the false-alarm rate. Of all the tested approaches, the proposed approach achieves the best performance. For text segmentation, encouraging empirical results are also obtained. Through experiments, the commercial OCR package can recognize approximately 72% of the segmented characters. While the proposed approaches have been shown to be effective, further improvement can be attained by exploiting the interframe relationship in videos such as multiframe analysis and the superresolution reconstruction of video text prior to text binarization [9, 10, 13].

Acknowledgements. The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 1072/02E (9040693)] and by a grant from City University of Hong Kong (Project No. 7001546). The authors would like to thank the anonymous reviewers for their comments in improving the content of this article.

References

1. Aradhye H, Dorai C, Shim J-C (2001) Study of Embedded font context and kernel space methods for improved videotext recognition. IBM Research Report RC 22064
2. Jähne B (2002) Digital image processing, 5th edn. Springer, Berlin Heidelberg New York
3. Cai M, Song J, Lyu MR (2002) A new approach for video text detection. In: International conference on image processing
4. Chen X, Yang J, Zhang J, Waibel A (2002) Automatic detection of signs with affine transformation. In: Proceedings of WACV
5. Flickner M, Sawhney H, Niblack W, Ashley J, Huang Q, Dom B, Gorkhani M, Hafner J, Lee D, Petkovic D, Steele D, Yanker P (1995) Query by image and video content: the QBIC system. IEEE Comput 28:23–32
6. Hua X-S, Liu W, Zhang HJ (2001) Automatic performance evaluation for video text detection. In: international conference on document analysis and recognition, pp 545–550
7. Hua X-S, Liu W, Zhang HJ (2004) Automatic performance evaluation protocol for video text detection algorithms. IEEE Trans Circuits Syst Video Technol 14(4):498–507
8. Kim KI, Jung K, Park SH, Kim HJ (2001) Support vector machine-based text detection in digital video. Int J Pattern Recog 34(2):527–529
9. Li H, Doerman D, Kia O (2000) Automatic text detection and tracking in digital video. IEEE Trans Image Process 9:147–156
10. Li H, Doerman D (2000) Superresolution-based enhancement of text in digital video. In: International conference on Image Processing
11. Lienhart R (2000) Dynamic video summarization of home video. SPIE Storage Retrieval Media Database 3972:378–389
12. Lienhart R, Effelsberg W (2000) Automatic text segmentation and text recognition for video indexing. Multimedia Syst Mag 8:69–81
13. Lienhart R, Wernicke A (2002) Localizing and segmenting text in images and videos. IEEE Trans Circuits Syst Video Technol 12:256–268
14. Ngo CW, Pong TC, Huang TS (2002) Detection of slide transition for topic indexing. In: IEEE conference on multimedia and expo
15. Niblack W (1986) An introduction to digital image processing Prentice Hall, Englewood Cliffs, NJ, pp 115–116
16. Shim JC, Dorai C, Bolle R (1998) Automatic text extraction from video for content-based annotation and retrieval. In: International conference on pattern recognition
17. Sauvola J, Seppanen T, Haapakoski S, Pietikainen M (1997) Adaotive document binarization. In: International conference on document analysis and recognition, 1:147–152
18. Sato T, Kanade T, Hughes EK, Smith MA (1997) Video OCR for digital news archives. In: ICCV workshop on image and video retrieval
19. Trier OD, Jain A (1995) Goal-directed evaluation of binarization methods. IEEE Trans Pattern Anal Mach Intell 17:1191–1201
20. Wong EK, Chen M (2003) A new robust algorithm for video text extraction. Pattern Recog 36:1397–1406
21. Wolf C, Jolion MJ, Chassaing F (2002) Text localization, enhancement and binarization in multimedia documents. In: International conference on pattern recognition, pp 1037–1040
22. Zhang J, Chen X, Hanneman A, Yang J, Waibel A (2002) A robust approach for recognizing of text embedded in natural scenes. In: International conference on pattern recognition
23. Zhong Y, Zhang HJ, Jain KA (2000) Automatic caption localization in compressed video. IEEE Trans Pattern Anal Mach Intell 22:385–392
24. OmniPage Pro 12. http://www.scansoft.com/omnipage/