A laplacian approach to multi-oriented text detection in video. IEEE Trans Pattern Anal Mach Intell (TPAMI)

Article *in* IEEE Transactions on Software Engineering · February 2011

DOI: 10.1109/TPAMI.2010.166 · Source: PubMed

CITATIONS
219

READS
500

3 authors:

Palaiahnakote Shivakumara
National University of Singapore
149 PUBLICATIONS **1,941** CITATIONS
SEE PROFILE

Trung Quy Phan
National University of Singapore
25 PUBLICATIONS **1,014** CITATIONS
SEE PROFILE

Chew Lim Tan
National University of Singapore
425 PUBLICATIONS **10,137** CITATIONS
SEE PROFILE

A Laplacian Approach to Multi-Oriented Text Detection in Video

Palaiahnakote Shivakumara, Trung Quy Phan and Chew Lim Tan, *Senior Member, IEEE*

Abstract—In this paper, we propose a method based on the Laplacian in the frequency domain for video text detection. Unlike many other approaches which assume that text is horizontally-oriented, our method is able to handle text of arbitrary orientation. The input image is first filtered with Fourier-Laplacian. K-means clustering is then used to identify candidate text regions based on the maximum difference. The skeleton of each connected component helps to separate the different text strings from each other. Finally, text string straightness and edge density are used for false positive elimination. Experimental results show that the proposed method is able to handle graphics text and scene text of both horizontal and non-horizontal orientation.

Index Terms—Connected component analysis, Frequency domain processing, Text detection, Text orientation

1 Introduction

In video databases, each video is manually tagged with a few keywords to allow for searching and retrieval. However, this process is laborious and inconsistent, i.e. two users may choose different keywords for the same video. An alternative approach is to generate the keywords from the text that appears in the frames. Video text can be classified into *graphics text*, which is artificially added to the video during the editing process, and *scene text*, which appears naturally in the scene captured by the camera. With the rapid growth of the Internet, there is an increasing demand for video text detection.

Although many methods have been proposed over the past years, text detection is still a challenging problem because of the unconstrained colors, sizes, and alignments of the characters. Moreover, scene text is affected by lighting conditions and perspective distortions [1], [2].

Text detection methods can be classified into three approaches: connected component-based, edge-based and texture-based. The first approach employs color quantization and region growing (or splitting) to group neighboring pixels of similar colors into connected components (CC) [3], [4]. However, these CCs may not preserve the full shape of the characters due to color bleeding and the low contrast of the text lines. Therefore, these methods do not work well for video images.

To overcome the problem of low contrast, edge-based methods are proposed. Liu et al. [5] extract statistical features from the Sobel edge maps of four directions and use K-means to classify pixels into the text and non-text clusters. Although this method is robust against complex background, it fails to detect low contrast text and text of small font sizes. It is also computationally expensive due to the large feature set. Cai et al. [6] design two filters to enhance the edges in text areas. This method uses various threshold values to decide whether to enhance the edges in a certain region and thus may not generalize well for different datasets. Wong et al. [7] compute the maximum gradient difference to identify the potential line segments. These segments are then extended to neighboring top and bottom rows to form candidate text regions. This method has a low false positive rate but it uses many heuristic rules and is sensitive to threshold values. Edge-based methods are fast but they produce many false positives for images with complex backgrounds.

To overcome the problem of complex background, the texture-based approach considers text as a special texture. These methods apply Fast Fourier Transform, discrete cosine transform, wavelet decomposition and Gabor filters for feature extraction. Ye et al. [8] compute the wavelet energy features at different scales and perform adaptive thresholding to find candidate text pixels, which are then merged into candidate text lines. Lee et al. [9] use support vector machines (SVM) to classify every pixel as text or non-text. For each M × M window, the feature vector of the center pixel consists of 4M – 3 neighboring grayscale values. Texture-based methods usually involve the use of classifiers such as SVM and neural networks and thus, they are trainable for different databases. However, these classifiers require a large training set of text and non-text samples [10]. It is especially hard to ensure that the non-text samples are representative.

From our survey, we have noticed that most papers address the detection of horizontal text but not multi-oriented text. This is because most of the non-horizontal text lines are scene text, which is much more difficult to detect due to varying lighting and complex transformations [1], [2]. For some existing methods, extension to multi-oriented text is no trivial matter. For example, the uniform-color method [11] performs color clustering on each row, while the gradient-based method [7] identifies candidate text segments row-wise. The edge-based method [6] analyzes horizontal and vertical profiles of the edge map. Thus many existing methods rely heavily on the horizontal text assumption and hence break down on multi-oriented text. Only a few papers consider scene text in video images [12], [13] under the assumption that text is of large font size and high contrast. Crandall et al. [14] propose a method for extracting multi-oriented special

- *Palaiahnakote Shivakumara is with the Department of Computer Science, School of Computing, National University of Singapore, Computing 1, 13 Computing Drive, Singapore 117417. Telephone: +65 6516 6903, Fax: +65 6779 4580, E-mail: shiva@comp.nus.edu.sg.*
- *T.Q. Phan and C.L. Tan are with the Department of Computer Science, School of Computing, National University of Singapore, Computing 1, 13 Computing Drive, Singapore 117417. E-mail: {phanquyt, tancl}@comp.nus.edu.sg.*

Manuscript received (insert date of submission if desired).

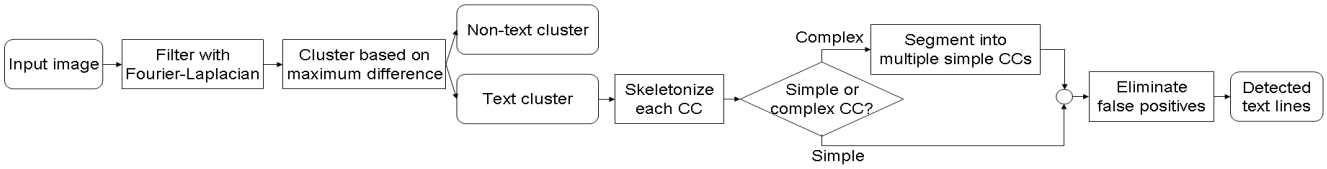
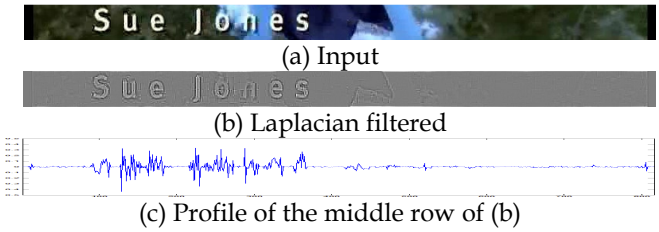


Fig. 1. Flowchart of the proposed method.



(a) Input

(b) Laplacian filtered

(c) Profile of the middle row of (b)

Fig. 2. Profiles of text and non-text regions. In (c), the x-axis shows the column numbers while the y-axis shows the pixel values.

effects text but this method is limited to graphics text of a few directions (0, 15, 30 degrees and so on). Related works for scene text in camera-based images are [15], [16] and [17], [18] (which focus specifically on text orientation). These methods are good for camera-based images but they do not work well for video images because they require text to be of high resolution in order to fully extract the clear shape of each single character [19], [20].

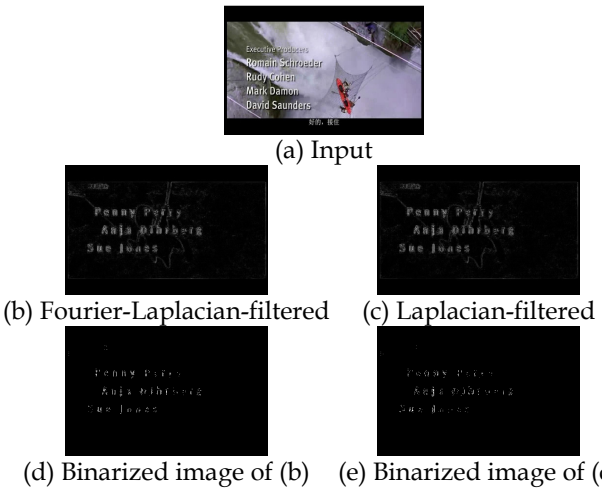
Multi-oriented text detection without any constraints on background, contrast and orientation, and with high precision and recall is still a challenging problem [14], [21], [22]. Therefore, in this paper, we propose a method which is able to handle both graphics text and scene text of arbitrary orientation under the assumption that the characters are aligned to a straight line.

2 Proposed Approach

The proposed method consists of four steps: text detection, connected component classification, connected component segmentation and false positive elimination. In the first step, we identify candidate text regions by using Fourier-Laplacian filtering. The second step uses skeletonization to analyze each CC in the text regions. Simple CCs are retained while complex CCs are segmented in the third step. False positives are removed in the last step. Fig. 1 shows the flowchart of the proposed method.

2.1 Text Detection

Because video text can have a very low contrast against complex local backgrounds, it is important to preprocess the input image to highlight the difference between text and non-text regions. We propose a two-step process called Fourier-Laplacian filtering to smooth noise in the image and detect candidate text regions. The first step uses an ideal lowpass filter in the frequency domain to smooth noise. The rationale is that the high frequency components of the Fourier transform contain information about noise and thus should be cut off. The second step is motivated by our earlier work [23], which uses the Laplacian mask in the spatial domain to detect candidate text regions. After the image is filtered, text regions have many positive and negative peaks of large magnitudes.



(a) Input

(b) Fourier-Laplacian-filtered (c) Laplacian-filtered

(d) Binarized image of (b) (e) Binarized image of (c)

Fig. 3. Fourier-Laplacian filtering produces a clearer separation between text and non-text than Laplacian filtering.

The reverse is true for non-text regions, which makes it possible to differentiate between text and non-text regions (Fig. 2c). To combine this step with the previous step, we use the Laplacian operator in the frequency domain (instead of the spatial domain):

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$
$$\Im\left[\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}\right] = -(u^2+v^2)F(u,v) \quad (1)$$

The combined Fourier-Laplacian filtering is as follows:

$$F(u,v) = \Im[f(x,y)]$$
$$G(u,v) = H_2(u,v)[H_1(u,v)F(u,v)]$$
$$H_1(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$
$$H_2(u,v) = -(u^2+v^2)$$
$$g(x,y) = \Im^{-1}[G(u,v)]$$

Equations (1) and (2) are based on the equations in [24]. $f(x, y)$ and $g(x, y)$ are the original image (converted to grayscale) and the filtered image. $H_1(u, v)$ is the ideal lowpass filter while $H_2(u, v)$ is the Laplacian operator. $D(u, v)$ is defined as the distance from (u, v) to the origin of the Fourier transform and D_0 is the cutoff distance.

Fig. 3 compares the effects of Fourier-Laplacian filtering and Laplacian filtering alone. Images (b) and (c) are the absolute images of $g(x, y)$ and the pixel values are normalized to the range [0, 1]. In these images, bright pixels correspond to text pixels while dark pixels correspond to non-text pixels. It is observed that image (b) is brighter than image (c), which means that Fourier-Laplacian filter-

(a) Input (b) Fourier-Laplacian filtered

Fig. 4. Result of Fourier-Laplacian filtering for non-horizontal text.

(a) MD map of Fig. 4b (b) Text cluster after *opening*

Fig. 5. Maximum difference map and the corresponding text cluster. In the map, brighter colors represent larger values.

(a) Connected component (b) Skeleton

Fig. 6. Skeleton of a connected component from Fig. 5b.

ing produces a clearer separation between text and non-text than Laplacian filtering. In order to show the difference in brightness more clearly, we binarize (b) and (c) with a fixed threshold of 0.5, i.e. a pixel is considered as a bright pixel if its grayscale value is greater than 0.5. Again, image (d) contains more white pixels than image (e). Therefore, we conclude that Fourier-Laplacian filtering is better than Laplacian filtering alone. The advantage of the former over the latter is that the ideal lowpass filter helps to smooth noise in video images and reduce the problem of noise sensitivity of the Laplacian operator.

Note that the proposed Fourier-Laplacian filter is similar to the Laplacian of Gaussian (LoG) filter because both filters consist of a smoothing filter and the Laplacian filter. The difference is that the former employs an ideal lowpass filter while the latter makes use of the Gaussian filter. The ideal lowpass filter is chosen because it has simpler parameters than the (spatial) LoG, which makes it easier to tune and generalize better to different datasets.

Fig. 4 shows a sample result of Fourier-Laplacian filtering for non-horizontal text. Its row profiles look similar to Fig. 2c, i.e. text regions have many positive and negative peaks while non-text regions do not. It is observed that the zero crossings correspond to the transitions between text and background. Ideally, there should be the same number of text-to-background and background-to-text transitions. This condition, however, does not hold for low contrast text on complex background so we use a weaker condition to ensure that the low contrast text is not missed. Maximum difference (MD) [7], defined as the difference between the maximum value and the minimum value within a local 1 × N window, is computed from the filtered image:

$$\begin{aligned} Max(x,y) &= \max_{\forall t\in[-N/2,N/2]} g(x,y-t) \\ Min(x,y) &= \min_{\forall t\in[-N/2,N/2]} g(x,y-t) \\ MD(x,y) &= Max(x,y) - Min(x,y) \end{aligned} \tag{3}$$

The MD map is obtained by moving the window over the image (Fig. 5a). Text regions have larger MD values than non-text regions due to the larger magnitudes of the positive and negative peaks. We use K-means to classify all pixels into two clusters, text and non-text, based on the Euclidean distance of MD values. Let the clusters returned by K-means be C_1 (cluster mean M_1) and C_2 (cluster mean M_2). Since the cluster order varies for different runs, the following rule is used to identify the text cluster:

$$Text\_Cluster = \begin{cases} C_1 & \text{if } M_1 > M_2 \\ C_2 & \text{otherwise} \end{cases} \tag{4}$$

The clustering step can also be thought of as binarizing the MD map. K-means is chosen for simplicity and it does not require any threshold values. The morphological operation *opening* is used to remove small artifacts (Fig. 5b).

The end result of this step, the text cluster, plays an important role in the proposed method because if it misses low contrast text lines, the subsequent steps will not be able to recover those lines. We have employed three main techniques in this step. First, because video images are noisy due to compression, an ideal lowpass filter is used to smooth noise. Second, the Laplacian operator is a second-order derivative operator which gives stronger response to fine detail than first-order derivative operators and thus ensures that low contrast text lines are not missed. Its disadvantage of noise sensitivity is largely reduced because the image has been smoothed. Finally, we employ the maximum difference (instead of the normal difference between corresponding positive and negative peaks) to further increase the gap between text and non-text. Hence, our text detection step combines the above techniques in a coherent way which has not been explored in earlier work.

2.2 Connected Component Classification

Traditionally, bounding boxes are used for displaying the detected text blocks. This is fine for horizontal text lines; however, for skewed text lines, rectangular boxes will enclose many unnecessary background pixels. Neighboring skewed text lines will also lead to overlapping bounding boxes. Hence we propose to use CCs as a unique way for displaying text lines. We further propose *skeletonization* to segment CCs into separate text lines.

There are two types of CCs: *simple* and *complex*. A simple CC is either a single text string or a false positive, e.g. the CCs at the bottom of Fig. 5b. On the other hand, a complex CC contains multiple text strings which are connected to each other and to false positives in the background. For example, the CC in the middle of Fig. 5b contains three text strings and a false positive (the logo). High contrast text often appears as simple CCs while low contrast text often appears as complex CCs.

In the first case (simple CCs), the whole component is displayed in the result (if it is a text CC) while in the second case (complex CCs), we want to output the text part and suppress the non-text part of the CC. In order to do so, we need to segment a complex CC into multiple simple CCs and retain only the text CCs.

The segmentation step will be described in detail in the

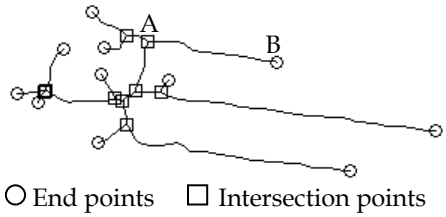
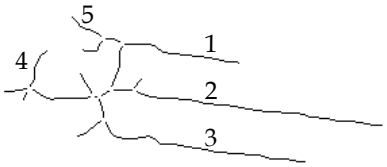


Fig. 7. End points and intersection points of Fig. 6b.



(a) Skeleton segments

(b) 1 (c) 2 (d) 3 (e) 4 (f) 5

Fig. 8. Skeleton segments of Fig. 6b and their corresponding sub-components (only 5 sample sub-components are shown here).

next section. In this section, we discuss how to classify every CC as either simple or complex. Skeleton is a well-defined concept in digital image processing to represent the structure of a region (Fig. 6). The intersection points (or junction points) of a skeleton show the locations where the sub-components of different orientation are connected to each other. The rule for CC classification is defined based on the number of intersection points:

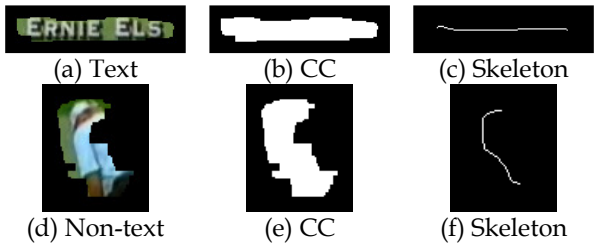
$$N\_Intersection = |Intersection(Skeleton(c_i))|$$
$$Type(c_i) = \begin{cases} Simple & \text{if } N\_Intersection = 0 \\ Complex & \text{otherwise} \end{cases} \quad (5)$$

$\{c_i\}$ is the set of CCs in the text cluster returned by the previous step. *Skeleton*(.) returns the result of skeletonization. *Intersection*(.) returns the set of intersection points. At the end of this step, simple CCs are retained while complex CCs are sent for segmentation.

2.3 Connected Component Segmentation

In order to output only the text part of a complex CC, we need to segment, or split, it into multiple simple CCs based on the intersection points. In Fig. 7, point A shows the location where the first text line of Fig. 4a connects to the logo (a false positive). By segmenting the complex CC from A to B, we are able to get back the first text line.

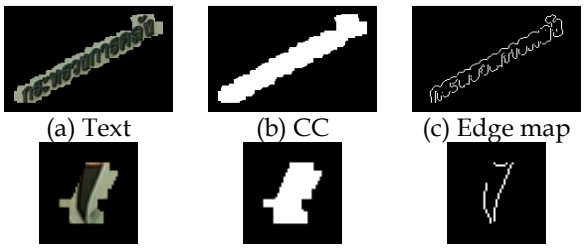
AB is called a skeleton segment, which is defined as a continuous path from an intersection point to either and end point or another intersection point. In addition, the path should not include any other point in the middle. A simple way to find the skeleton segments is to delete all the intersection points. After that, for each skeleton segment, we extract the corresponding sub-component from the complex CC. In Fig. 8, sub-components 1, 2 and 3 are the first three text lines in Fig. 4a while sub-components 4 and 5 are false positives (part of the logo).



(a) Text (b) CC (c) Skeleton

(d) Non-text (e) CC (f) Skeleton

Fig. 9. False positive elimination based on straightness.



(a) Text (b) CC (c) Edge map

(d) Non-text (e) CC (f) Edge map

Fig. 10. False positive elimination based on edge density.

2.4 False Positive Elimination

After the previous step, we have a set of simple CCs, $\{b_i\}$, each of which is either an original simple CC or a new CC segmented from a complex CC. b_i is a true text block if:

$$Straightness(b_i) < T_1 \wedge Edge\_Density(b_i) \geq T_2 \quad (6)$$

The first feature, straightness, comes from the observation that text strings appear on a straight line (our assumption) while false positives can have irregular shapes.

$$s_i = Skeleton(b_i)$$
$$Straightness(b_i) = \frac{Length(s_i)}{End\_Distance(s_i)} \quad (7)$$

Note that all b_i's are simple CCs and thus all s_i's have exactly two end points and zero intersection points. For text, *Length*(.), the length of the skeleton, is close to *End_Distance*(.), the straight line distance between the two end points while for non-text, *Length*(.) is much larger than *End_Distance*(.) (Fig. 9).

The second feature, edge density, is defined as follows:

$$e_i = Sobel(b_i)$$
$$Edge\_Density(b_i) = \frac{Edge\_Length(e_i)}{CC\_Area(b_i)} \quad (8)$$

Sobel(.) returns the binary Sobel edge map (e_i contains edge information only for the white pixels of b_i). *Edge_Length*(.) is the total length of all edges in the edge map. *CC_Area*(.) is the area of the CC. *Edge_Length*(.) and *CC_Area*(.) can be computed by counting the number of edge pixels and white pixels, respectively. This feature assumes that the edges are denser in text regions than non-text regions (Fig. 10).

3 Experimental Results

As there is no standard benchmarking dataset available, we have selected a variety of video images, extracted from news programmes, sports videos and movie clips.

(a) Input (b) Detected blocks

Fig. 11. Sample ATBs, TDBs, FDBs and MDBs.

For comparison purpose, we have implemented four existing methods. Method [5], denoted as *edge-based method 1*, extracts six statistical features from four Sobel edge maps. Method [6], denoted as *edge-based method 2*, performs Sobel edge detection in the YUV color space and applies two text area enhancement filters. Method [7], denoted as *gradient-based method*, computes the maximum gradient difference to identify candidate text regions. Method [11], denoted as *uniform-colored method*, performs hierarchical clustering in the L*a*b* color space to locate uniform-colored text strings.

The parameters in (3) and (6) are empirically determined: $N = 21$, $T_1 = 1.2$ and $T_2 = 0.1$. The parameters of the existing methods are set according to the recommended values in the respective papers. The same parameter values are used for all the experiments.

3.1 Performance Measures

We evaluate the performance at the block level, which is a common granularity level in the literature [6], [7], [8], [10], [11], rather than the word or character level because we have not considered text recognition in this work. The following categories are defined for each detected block by a text detection method.

- *Truly Detected Block* (*TDB*): A detected block that contains at least one true character. Thus, a TDB may or may not fully enclose a text line.
- *Falsely Detected Block* (*FDB*): A detected block that does not contain text.
- *Text Block with Missing Data* (*MDB*): A detected block that misses more than 20% of the characters of a text line (MDB is a subset of TDB). The percentage is chosen according to [10], in which a text block is considered correctly detected if it overlaps at least 80% of the ground-truth block.

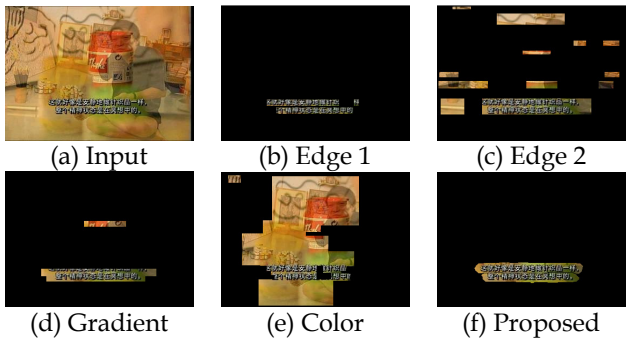
For each image in the dataset, we also manually count the number of *Actual Text Blocks* (*ATB*), i.e. the true text blocks. For example, the numbers of the different types of blocks in Fig. 11 are 3 ATBs (3 text lines), 3 TDBs (all lines are detected), 1 FDB (the eye) and 2 MDBs (more than 20% of the first two lines are missed).

The performance measures are defined as follows.

- *Recall (R)* = TDB / ATB
- *Precision (P)* = TDB / (TDB + FDB)
- *F-measure (F)* = $2 \times P \times R$ / (P + R)
- *Misdetection Rate* (*MDR*) = MDB / TDB

There are two other performance measures commonly used in the literature, *Detection Rate* and *False Positive Rate*; however, they can also be converted to Recall and Precision: Recall = Detection Rate and Precision = 1 – False Positive Rate [7]. Hence only the above four performance measures are used for evaluation.

Our definition of Recall is more forgiving than its traditional definition because the former allows both fully and partially detected text lines while the latter only considers fully detected text lines. Due to the challenges of scene text and the arbitrary orientation of the text lines, it is difficult for a method to always enclose a full text line in a block. Sometimes it misses some characters of very low contrast and detects only parts of a line. Since the goal of this paper is text detection (how well a method locates potential text blocks), partial detection (even to the extent of just a few true characters) is still acceptable because it shows that a method is able to locate a block (albeit partially). Hence our definition of Recall is more forgiving but is in line with the definition of Detection Rate in the literature. We also include MDR as a performance measure and provide discussion on partial detection in all experiments to ensure a fair comparative study.



(a) Input (b) Edge 1 (c) Edge 2

(d) Gradient (e) Color (f) Proposed

Fig. 12. The detected blocks of the four existing methods and the proposed method for a horizontal text image.

3.2 Experiment on Horizontal Text

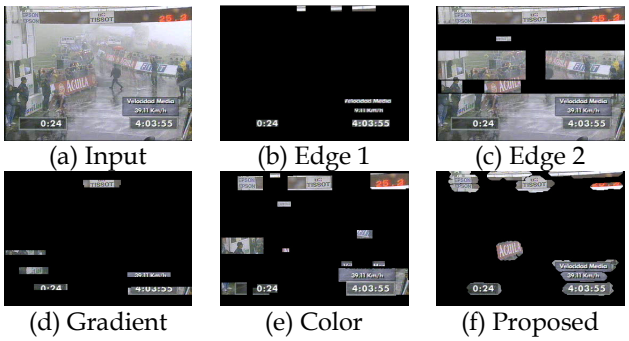
In this experiment, we have selected 960 images. The English sub-dataset contains 800 images (652 images for graphics text and 148 images for scene text) while the Chinese sub-dataset contains 160 images (153 for graphics text and 7 for scene text).

Fig. 12 shows a sample image with two horizontal Chinese text lines on a complex background. The edge-based method 1 misses some characters of the first line. The edge-based method 2 and the uniform-colored method produce many false positives while the gradient-based method fails to detect the first line. The proposed method is the only one that fully detects and separates the text lines from each other, without any false positives.

Table 1 shows the performance of the four existing methods and the proposed method on the horizontal text dataset. The proposed method has the highest recall, the second highest precision (almost the same as that of the gradient-based method) and the highest F-measure. This shows the advantage of the proposed method because it achieves good results while making fewer assumptions about text. By assuming that text appears on a horizontal straight line, the existing methods can remove false positives more easily, e.g. by using projection profile analysis.

The drawback of the proposed method is MDR, which is not as good as those of the gradient-based method and the edge-based method 2. This drawback is discussed in detail in section 3.4.

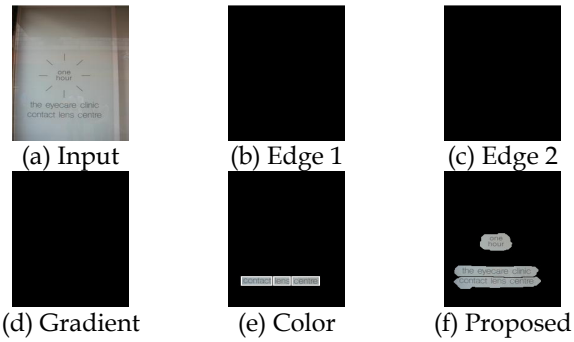
Table 1 also shows that the proposed method works slightly better for English text than Chinese text for two



(a) Input (b) Edge 1 (c) Edge 2

(d) Gradient (e) Color (f) Proposed

Fig. 13. The detected blocks of the four existing methods and the proposed method for an image from the Microsoft Asia dataset.



(a) Input (b) Edge 1 (c) Edge 2

(d) Gradient (e) Color (f) Proposed

Fig. 14. The detected blocks of the four existing methods and the proposed method for an image from the ICDAR 2003 dataset.

TABLE 1
EXPERIMENTAL RESULTS ON HORIZONTAL TEXT

Method	English				Chinese			
	R	P	F	M	R	P	F	M
Edge 1	0.58	0.68	0.63	0.22	**0.79**	0.63	0.70	0.43
Edge 2	0.58	0.39	0.47	0.12	0.61	0.36	0.45	0.18
Gradient	0.66	**0.83**	0.74	**0.03**	0.69	**0.76**	0.72	**0.10**
Color	0.55	0.45	0.50	0.35	0.69	0.51	0.59	0.56
Proposed	**0.86**	0.82	**0.84**	0.13	**0.79**	0.75	**0.77**	0.23

R – Recall (= Detection Rate), P – Precision (= 1 – False Positive Rate), F – F-measure, M – Misdetection Rate

TABLE 2
EXPERIMENTAL RESULTS ON PUBLIC DATASETS

Method	ICDAR 2003				Microsoft Asia			
	R	P	F	M	R	P	F	M
Edge 1	0.53	0.61	0.57	0.24	0.75	0.54	0.63	0.16
Edge 2	0.67	0.33	0.44	0.43	0.69	0.43	0.53	0.13
Gradient	0.52	**0.83**	0.64	**0.08**	0.51	0.75	0.61	0.13
Color	0.60	0.44	0.51	0.45	0.47	0.44	0.45	0.44
Proposed	**0.86**	0.76	**0.81**	0.13	**0.93**	**0.81**	**0.87**	**0.07**
ICDAR 1	0.62	0.67	0.62	N.A	N.A	N.A	N.A	N.A
ICDAR 2	0.60	0.60	0.58	N.A	N.A	N.A	N.A	N.A

R – Recall (= Detection Rate), P – Precision (= 1 – False Positive Rate), F – F-measure, M – Misdetection Rate

reasons. First, the spaces between Chinese characters can be larger than those of English characters. Second, Chinese characters have more complicated strokes and thus, for text lines of small font sizes, it is difficult to distinguish the strokes from complex backgrounds.

3.3 Experiment on Horizontal Text with Public Datasets

In addition to our own dataset, we consider two publicly available datasets for this experiment: the Microsoft Research Asia dataset [25] and the ICDAR 2003 Text Locating dataset (in particular, the SceneTrialTest dataset) [26]. The first dataset contains 45 images (33 for graphics text and 12 for scene text) while the second dataset contains 251 camera-based images (all images are for scene text). Since the images in the second dataset are of high resolution, they are resized to a standard width (or height, for portrait images) of 256 pixels to save computational costs.

Fig. 13 shows a sample image from the Microsoft Asia dataset. This image has both graphics text (at the bottom left and right corners) and scene text (at the top). The edge-based method 1 detects the graphics text but misses the scene text. The edge-based method 2 detects both types of text but fails to separate the text strings into individual text blocks. The gradient-based method misses some graphics text (the first graphics text line) and scene text (the two scene text blocks at the top left and right corners). The uniform-colored method produces many false positives. Finally, the proposed method detects all the text blocks correctly, except one at the top right corner. One of the billboards is also included in the result. However, there are two false positives at the top.

Fig. 14 shows sample results of the existing methods and the proposed method for the ICDAR 2003 dataset. The sample image contains four text lines of extremely low contrast. In addition, the clear reflections on the window glass make it even more difficult to detect the text lines. Therefore, three out of four existing methods fail to detect any text lines while the last method (the uniform-colored method) only detects one text line. The proposed method is able to detect all the text lines, although it fails to separate the first two lines into individual text blocks.

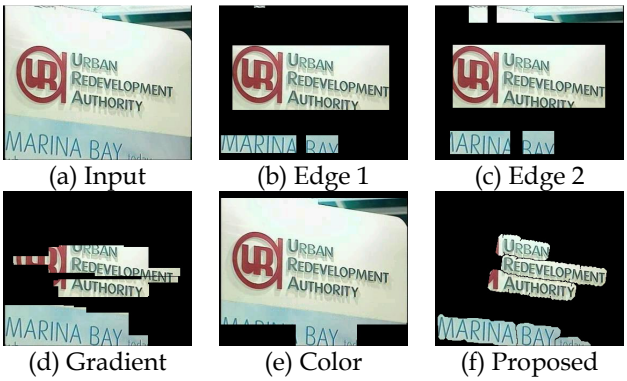
The experimental results on the ICDAR 2003 dataset are similar to those of the previous experiment: the proposed method achieves the highest recall and F-measure while the gradient-based method has the highest precision and the lowest MDR (Table 2). The performance of the top two entries (denoted as *ICDAR 1* and *ICDAR 2*) of the ICDAR 2005 Competition, which used the same dataset as the 2003 competition, is also included for reference [27]. However, the results may not be directly comparable because we use a more relaxed definition of recall than in [27] and perform evaluation at the block level instead of the word level, as explained in section 3.1.

For the Microsoft Asia dataset, the proposed method outperforms all existing methods in all performance measures. The MDR of the proposed method is especially low for this dataset compared to the ICDAR 2003 dataset as well as the previous dataset. The reason is that most of the text lines in this dataset are high contrast graphics text and thus, the proposed method misses fewer characters.

3.4 Experiment on Non-horizontal Text

241 images are used for this experiment. The English sub-dataset contains 220 images (44 for graphics text and 176 for scene text) while the Chinese sub-dataset contains 21 images (4 for graphics text and 17 for scene text).

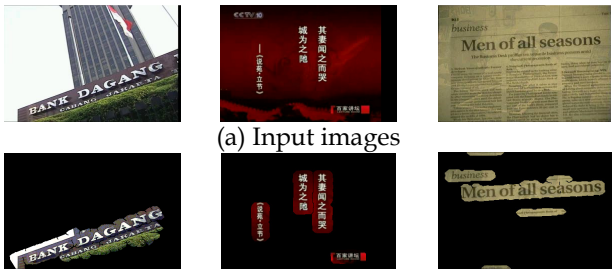
Fig. 15 shows a sample image which has four non-



(a) Input (b) Edge 1 (c) Edge 2

(d) Gradient (e) Color (f) Proposed

Fig. 15. The detected blocks of the four existing methods and the proposed method for a non-horizontal text image.



(a) Input images

(b) Detected blocks for the images in (a)

Fig. 16. Results of the proposed method for non-horizontal text.

horizontal text strings. All the existing methods are able to detect the three text strings in the middle but fail to separate them from each other. This is because if we try to enclose each string into a (non-rotated) rectangular bounding box, we would observe that the boxes overlap each other. The proposed method detects all the text strings correctly. Note that the output is wider than the text cluster in Fig. 5b because we use the morphological operation *thickening* to slightly extend each CC.

Fig. 16 shows more sample results where the proposed method works for multi-oriented text of different font sizes. The third line in the third input image can only be partially detected because the font size is too small.

It is clear from Fig. 15 that the existing methods are not designed for non-horizontal text and thus will not perform well for this dataset. For this reason, we consider only the proposed method in this experiment. Table 3 shows that this dataset is more challenging than the dataset in section 3.2. The proposed method achieves lower recall, precision and F-measure, and higher MDR for both sub-datasets. The only exception is the recall on the Chinese sub-dataset, which is slightly higher than that of the Chinese horizontal text sub-dataset.

The fact that the proposed method achieves high recall for both sub-datasets is encouraging because the scene text lines in this experiment are more difficult to detect due to their low contrast and arbitrary orientation. This experiment shows that the proposed method is able to handle both horizontal and non-horizontal text.

Similar to previous experiments, the proposed method has a high MDR due to the CC segmentation step. The intention is to segment a whole text string into a simple CC. However, sometimes a string is split into multiple

TABLE 3
EXPERIMENTAL RESULTS OF THE PROPOSED METHOD ON NON-HORIZONTAL TEXT

Method	English				Chinese			
	R	P	F	M	R	P	F	M
Proposed	0.85	0.77	0.81	0.14	0.81	0.74	0.77	0.35

R – Recall (= Detection Rate), P – Precision (= 1 – False Positive Rate), F – F-measure, M – Misdetection Rate

TABLE 4
EXPERIMENTAL RESULTS ON NON-TEXT IMAGES

Method	Images Falsely Classified as Text Frames	False Positive Blocks
Edge 1	238	953
Edge 2	290	1090
Gradient	**107**	**196**
Color	278	690
Proposed	269	776

CCs. If one of them does not satisfy the false positive elimination rule, only part of the string is extracted.

Another aspect of the proposed method which could be improved is the precision. However, it is difficult to eliminate false positives without accidentally removing true text blocks because both come from the same scene.

3.5 Experiment on Non-text Images

This experiment tests the performance of the proposed and existing methods on 300 non-text images. The motivation is that although a video typically contains more non-text frames than text frames, many methods in the literature assume that the input image contains text.

For non-text images, a text detection method should ideally detects no blocks (in other words, any blocks detected would be false positives). Therefore, the performance measures of this experiment are the total number of false positive blocks and the number of images falsely classified as text frames. The second measure is defined as the number of images where a method (falsely) detects at least one block. For both measures, lower is better.

The gradient-based method is the best method in this experiment (Table 4). The proposed method produces less false positives but wrongly classifies more images than the edge-based method 1. This experiment demonstrates the importance of text frame selection, which should be considered by all text detection methods.

3.6 Experiment on Processing Time

Table 5 shows the average processing time of the proposed method and the existing methods for a 256 × 256 image on a Core 2 Duo 2.0 GHz machine. For the existing methods, the processing time is reported only for horizontal text because they are not included in the experiment on non-horizontal text. The proposed method is slower than the gradient-based method, slightly slower than the edge-based method 2 but much faster than the edge-based method 1 and the uniform-colored method. The proposed method also takes longer to detect non-horizontal text than horizontal text because more time is required to segment complex CCs into simple CCs.

TABLE 5
AVERAGE PROCESSING TIME (IN SECONDS)

Method	Horizontal Text	Non-horizontal Text
Edge 1	22.1	N.A
Edge 2	6.1	N.A
Gradient	**1.1**	N.A
Color	13.9	N.A
Proposed	7.8	10.3

4 CONCLUSION AND FUTURE WORK

We have proposed a method for video text detection which is able to handle both graphics text and scene text. Our focus for scene text is text orientation because traditional methods only consider horizontal text. The Laplacian in the frequency domain helps to identify the candidate text regions while our new idea of using skeleton serves to segment a complex CC into constituent parts and separate connected text strings from each other. The experimental results show that the proposed method performs well for both horizontal and non-horizontal text. It outperforms three out of four existing methods and achieves higher recall and F-measure than the last one.

In the future, we plan to address the problems of false positives and partially detected text lines. The edge map could be used to verify the intersection points found by skeletonization. We would also like to extend this work to cursive text.

ACKNOWLEDGMENT

This research is supported in part by the MDA grant R252-000-325-279 and A*STAR grant R252-000-402-305.

We would like to thank the anonymous reviewers for their constructive comments which helped to improve the paper.

REFERENCES

[1] J. Zang and R. Kasturi, "Extraction of Text Objects in Video Documents: Recent Progress", In *Proc. DAS 2008*, pp. 5-17.

[2] K. Jung, K.I. Kim and A.K. Jain, "Text information extraction in images and video: a survey", *Pattern Recognition*, 37, 2004, pp. 977-997.

[3] Y. Zhong, K. Karu and A.K. Jain, "Locating Text in Complex Color Images", In *Proc. ICDAR 1995*, pp. 146.

[4] K. Sobottka, H. Bunke and H. Kronenberg, "Identification of Text on Colored Book and Journal Covers", In *Proc. ICDAR 1999*, pp. 57.

[5] C. Liu, C. Wang and R. Dai, "Text Detection in Images Based on Unsupervised Classification of Edge-based Features", In *Proc. ICDAR 2005*, pp. 610-614.

[6] M. Cai, J. Song and M.R. Lyu, "A New Approach for Video Text Detection", In *Proc. ICIP 2002*, pp. 117–120.

[7] E.K. Wong and M. Chen, "A new robust algorithm for video text extraction", *Pattern Recognition* 36, 2003, pp. 1397-1406.

[8] Q. Ye, Q. Huang, W. Gao and D. Zhao, "Fast and robust text detection in images and video frames", *Image and Vision Computing* 23, 2005, pp. 565-576.

[9] C.W. Lee, K. Jung and H.J. Kim, "Automatic text detection and removal in video sequences", *Pattern Recognition Letters* 24, 2003, pp. 2607-2623.

[10] D. Chen, J.M. Odobez and J.P. Thiran, "A localization/verification scheme for finding text in images and video frames based on contrast independent features and machine learning", *Signal Processing: Image Communication* 19, 2004, pp. 205-217.

[11] V.Y. Mariano and R. Kasturi, "Locating Uniform-Colored Text in Video Frames", In *Proc. ICPR 2000*, pp. 539-542.

[12] K. L Kim, K. Jung and J. H. Kim, "Texture-Based Approach for Text Detection in Images using Support Vector Machines and Continuously Adaptive Mean Shift Algorithm", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 12, December 2003, pp. 1631-1639.

[13] F. Wang, C. W. Ngo and T. C. Pong, "Structuring low quality videotaped lectures for cross-reference browsing by video text analysis", *Pattern Recognition* 41, 2008, pp. 3257-3269.

[14] D. Crandall, S. Antani and R. Kasturi (2003), "Extraction of Special Effects Caption Text Events from Digital Video", *Int J Doc Anal Recog* 5(2–3):138–157, 2003.

[15] U. Bhattachatya, S. K Parui and S. Mondal, Devenagari and Bangla, "Text Extraction from Natural Scene Images", In *Proc. ICDAR 2009*, pp. 171-175.

[16] X. Chen, J. Yang, J. Zhang and A. Waibel, "Automatic Detection and Recognition of Signs from Natural Scenes", *IEEE Transactions on Image Processing*, Vol. 13, No. 1, 2004, pp. 87-99.

[17] P. P Roy, U. Pal, J. Llados and M. Delalandre, "Multi-Oriented and Multi-Sized Touching Character Segmentation using Dynamic Programming", In *Proc. ICDAR 2009*, pp. 11-15.

[18] P. P .Roy, U. Pal, J. Liados and F. Kimura, "Covex Hull based Approach for Multi-Oriented Character Recognition from Graphical Documents", In *Proc. ICPR 2008*.

[19] K. Jung, "Neural network-based text location in color images", *Pattern Recognition Letters* 22, 2001, pp. 1503-1515.

[20] X. Tang, X. Gao, J. Liu and H. Zhang, "A Spatial-Temporal Approach for Video Caption Detection and Recognition", *IEEE Transactions on Neural Networks*, Vol. 13, No. 4, July 2002, pp 961-971.

[21] M. R. Lyu, J. Song and M. Cai, "A Comprehensive Method for Multilingual Video Text Detection, Localization, and Extraction", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 2, February 2005, pp 243-255.

[22] R. Lienhart and A. Wernickle, "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.

[23] T.Q. Phan, P. Shivakumara and C.L. Tan, "A Laplacian Method for Video Text Detection", In *Proc. ICDAR 2009,* pp. 66-70.

[24] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, pp. 167-180, 2002.

[25] X.S. Hua, L. Wenyin and H.J. Zhang, "An Automatic Performance Evaluation Protocol for Video Text Detection Algorithms", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, No. 4, April 2004, pp. 498-507.

[26] http://algoval.essex.ac.uk/icdar/TextLocating.html

[27] S.M. Lucas, "ICDAR 2005 Text Locating Competition Results", In *Proc. ICDAR 2005*, Vol. 1, pp. 80–84.