Multimed Tools Appl (2015) 74:859–884
DOI 10.1007/s11042-013-1702-7

A method for text line detection in natural images

Jie Yuan · Baogang Wei · Yonghuai Liu · Yin Zhang · Lidong Wang

Published online: 27 September 2013

Abstract Text information in natural images is very important to cross-media retrieval, index and understanding. However, its detection is challenging due to varying backgrounds, low contrast between text and non-text regions, perspective distortion and other disturbing factors. In this paper, we propose a novel text line detection method which can detect text line aligned with a straight line in any direction. It is mainly composed of three steps. In the first step, we use the maximal stable extremal region detector with dam line constraint to detect candidate text regions, we then define a similarity measurement between two regions which combines sizes, absolute distance, relative distance, contextual information and color histograms. In the second step, we propose a text line identification algorithm based on the defined similarity measurement. The algorithm firstly searches three regions as the seeds of a line, and then expands to obtain all regions in the line. In the last step, we develop a filter to remove non-text lines. The filter uses a sparse classifier based on two dictionaries which are learned from feature vectors extracted from morphological skeletons of those candidate text lines. A comparative study using two datasets shows the excellent performance of the proposed method for accurate text line detection with horizontal or arbitrary consistent orientation.

Keywords Text detection · Text line · Maximal stable extremal regions · Sparse classifier

J. Yuan (✉)
Jiangsu Electric Power Information Technology Co. Ltd., Nanjing 210029, China
e-mail: java_mc@163.com

B. Wei · Y. Zhang
College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

B. Wei
e-mail: wbg@zju.edu.cn

Y. Zhang
e-mail: yinzh@zju.edu.cn

Y. Liu
Department of Computer Science, Aberystwyth University, Wales, UK SY23 3DB
e-mail: yyl@aber.ac.uk

L. Wang
Qianjiang College, Hangzhou Normal University, Hangzhou 310027, China
e-mail: violet_wld@163.com

1 Introduction

With the development of multimedia, internet, and electronic technologies, images can easily be captured and transmitted at high speeds. Images contain rich information that is usually embedded into complex structures. How to effectively organize, retrieve and understand them has become an imperative task. Many images contain meaningful text information which is very rich in semantics, for example, the name on the front cover of a book, the trademark of a computer, some instruction on a road board and so on. Text information such as program name, time and so on is also usually added into video frame images. However, text information contained in images is usually hard to acquire by other ways than text extraction and recognition. Text information from images also enables useful real-world applications such as image annotation and retrieval [6, 32].

Some researchers have managed to utilize text information to organize and retrieve images or videos [10, 18, 21, 22, 25, 26, 31]. There are several advantages to do so. On one hand, most text contained in images is relevant to the semantic contents of those images, and text needs only limited spaces for storage. On the other hand, text in images is likely to be recognized with a high accuracy. Besides, text is naturally a means for content expression. Text detection manages to find out the accurate areas of text pixels in a image or video frame[5, 22, 31]. Text in images can be classified into two main categories as in Fig. 1: (1) scene text [29]: text that appears in natural scenes, and (2) artificial text [4, 12]: text that is added in the post-production stage. Generally speaking, with the variation of background, view perspective, illumination condition, text location and other aspects, scene texts are more challenging to detect than artificial texts. In this paper, we mainly focus on the detection of scene text.

Text detection methods can be classified into three main categories [11]: gradient based methods [13, 18, 24, 28, 31, 33], texture based methods [1, 23] and color clustering based methods [7, 30]. Gradient based methods assume that text exhibits strong edges against background, and therefore those pixels with high gradient values are regarded as good candidates for text regions. [31] and [18] both detect text strokes by searching stroke paths, in which two end point pairs on the edge mask have approximately opposite gradient directions, and then use clustering and other heuristic rules to classify those strokes into different text lines. They share

(a)

(b)

Fig. 1 Examples for: **a** Scene text, **b** Artificial text

the advancement that both of them can detect text lines with arbitrary consistent orientation. However, based on stroke detection, they work not well on images with complex background. [24] employed the Fourier-Laplacian filter to enhance the difference between text and non-text regions, and then used the K-means clustering to identify candidate text regions based on the maximum difference, finally used skeletons to split candidate regions and heuristic rules to identify text regions. While their method can also detect non-horizontal text lines, it sometimes detects broken text regions. [13] used contour and temporal features to enhance the accuracy of text caption localization in videos, But the method could not be used in scene text detection directly for the lack of temporal features. Gradient based methods become less reliable when the scene also contains many edges in the background. Texture based methods extract texture by Gabor filters, wavelet transforms, fast Fourier transforms (FFT), spatial variance multichannel processing and so on. By means of texture features, texts can be detected by machine learning methods, such as neural networks and support vector machines (SVM). [1] used discrete wavelet transform to detect three kinds of edges and it used neural network to obtained text regions. The method could detect text embeded in complex background, But neural network takes a lot of time in training the weighting values and it could only detect horizontal text. [23] presented new statistic features based on Fourier transform in RGB space (FSF) and then used the K-means clustering to separate text pixels from the background, the projection profiles of text pixels were analyzed and some heuristics were finally used to identify text blocks. Texture based methods usually have two shortcomings. On one hand, they need a set of representative images for training which is not easily obtained generlly; on the other hand, because of the signal response on few directions, they can only detect text in horizontal or vertical orientation. This is not sufficient for text detection in natural images. Color-based approaches assume that the text in images possesses a uniform color. [30] firstly detected the accumulated edge map of an image, and then colorized and decomposed it using the affinity propagation (AP) clustering algorithm [3]. Finally, a projection algorithm was employed to detect text regions in each decomposed edge map. Their method can make text detection and recognition more accurate, however, it is barely true that text appears in a uniform color in images. [7] proposed a split-and-merge segmentation method using colour perception to extract text from web images. Their method can detect text lines in arbitrary orientation, but it did not work well on scene images which usually have more complex color distribution than that in web images. There are also some methods using other features to detect texts in images recently. For example, [34] proposed a corner based approach to detect text and captions. But in natural images, only corners are not sufficient for text detection because of the low contrast between text and non-text regions in many natural images. [8] used the existing transient colors between inserted text and its adjacent background to generate a transition map. While the transient color is usually hard to detect especially in images containing scene text only, it performs poorly on natural images.

In this paper, we propose a novel text line detection method which can detect scene text in natural images. Firstly, maximally stable extremal regions (MSER) [9, 17] are detected. A MSER is a region which keeps stable in image binarization when modifying the threshold in a certain range. To prevent unwanted regions from being merged, the canny edge of the image is used to serve as dam line. The connected components (CCs) are identified by using 4 neighbor connections on the MSER mask image with dam lines. We then define enhanced geometry distance (similarity) and color distance (similarity) between CC pairs. Based on the distance (similarity) measurement, those CCs with center points in the same line are organized into candidate text lines. Finally, all candidate text lines are transformed into horizontal or vertical ones by a rotation operation, and a sparse classifier is used to identify real text lines. Our method uses the edge constrained MSER method to detect text regions, so it can detect more stable regions compared to general MSERs, yet overcoming the shortcomings of gradient-based

methods. By using text line detection and rotation transformation, our method can detect text lines aligned with a straight line in any direction. By using the sparse classifier as a filter, our method can obtain a higher accuracy than the existing methods.

The contributions of this paper can be summarized mainly as three aspects as follows:

(1) A similarity measurement between any two connected components is developed. It integrates region sizes, absolute distance, relative distance, contextual information and color information into a single value. It is powerful in characterizing text regions.
(2) A text line identification algorithm is proposed. Based on the similarity measurement, the text line identification algorithm firstly searches three CCs as the seed of a line, and then expands to obtain all other CCs in the line. The method is effective for candidate text line extraction.
(3) A new filter is developed to remove those non-text lines. To this end, a sparse classifier based on two dictionaries learned from a feature vector extracted from morphological skeletons of the MSERs has been developed. With the sparse filter, our method can obtain a higher precision than other methods.

To validate our proposed method for text line detection in natural images, two datasets were used. For a comparative study, several art methods were used. Experimental results show that our method outperforms significantly the selected state of the art ones.

The remaining of this paper is organized as follows: Section 2 briefly describes the framework of our novel text line detection method. Candidate text region detection and similarity measurement are described in Section 3. Section 4 details the proposed text line detection method. The sparse filter is developed in Section 5. Experimental results and analyses are presented in Section 6. Finally, we draw some conclusions and indicate our future work in Section 7.

2 The framework of text detection

Our text line detection method consists of 3 steps as shown in Fig. 2. The first step mainly uses the MSER detector to detect all candidate text regions. Though the detected MSERs have consistent intensity in themselves, they are isolate from each other and un-structured. In the second step, nearby regions are merged into candidate text lines based on the similarity and angles among them. Candidate text lines contain not only text lines but also some non-text lines. Finally, a sparse filter is used to get rid of non-text lines. The sparse filter uses reconstruction error by learned dictionaries of sparse classifier to work.

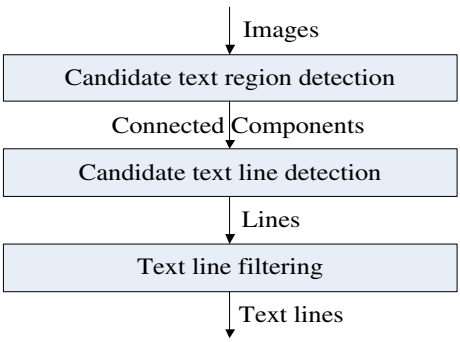


Fig. 2 The procedure of text line detection operation

3 Region detection and similarity definition

In this section we detail the candidate text region detection and measurement of similarity between two candidate text regions.

3.1 Maximally stable extremal regions

Definition [17]

Image $\boldsymbol{I}$ is a mapping $\boldsymbol{I}:\boldsymbol{D}\subset\mathbb{Z}^2\rightarrow\boldsymbol{SI}$. Two conditions should be met under which Extremal regions are well defined:

(1) $\boldsymbol{S}$ is totally ordered.
(2) An adjacency (neighbourhood) relation $\boldsymbol{A}\subset\boldsymbol{D}\times\boldsymbol{D}$ is defined.

Region $\boldsymbol{Q}$ is a contiguous subset of $\boldsymbol{D}$

(Outer) Region Boundary $\partial\boldsymbol{Q}=\{q\in\boldsymbol{D}\setminus\boldsymbol{Q}:\exists p\in\boldsymbol{Q}:qAp\}$

Extremal Region $\boldsymbol{Q}\subset\boldsymbol{D}$ is a region such that for all $p\in\boldsymbol{Q}$, $q\in\partial\boldsymbol{Q}$:$\boldsymbol{I}(p)>I(q)$ (maximum intensity region) or $\boldsymbol{I}(p)<I(q)$ (minimum intensity region).

Maximally Stable Extremal Region (MSER). Let $\boldsymbol{Q}_1,\boldsymbol{Q}_2,\ldots\ldots\boldsymbol{Q}_{i-1},\boldsymbol{Q}_i,\ldots\ldots\boldsymbol{b}$e a sequence of nested extremal regions, Extremal region $\boldsymbol{Q}_{i^*}$ is maximally stable iff $\boldsymbol{q}(\boldsymbol{i})=|\boldsymbol{Q}_{\boldsymbol{i}+\Delta}\backslash\boldsymbol{Q}_{\boldsymbol{i}-\Delta}|/|\boldsymbol{Q}_{\boldsymbol{i}}|$ has a local minimum at $\boldsymbol{i}^*$(|.| denotes cardinality). $\Delta\in\boldsymbol{S}$ is a parameter of the method.

There are mainly three reasons for us to select MSER as our candidate region detector: firstly it is invariant to affine transformation of image intensities, secondly it is very stable and lastly it can detect regions at different scales. Readers can refer to [17] for more details about MSER.

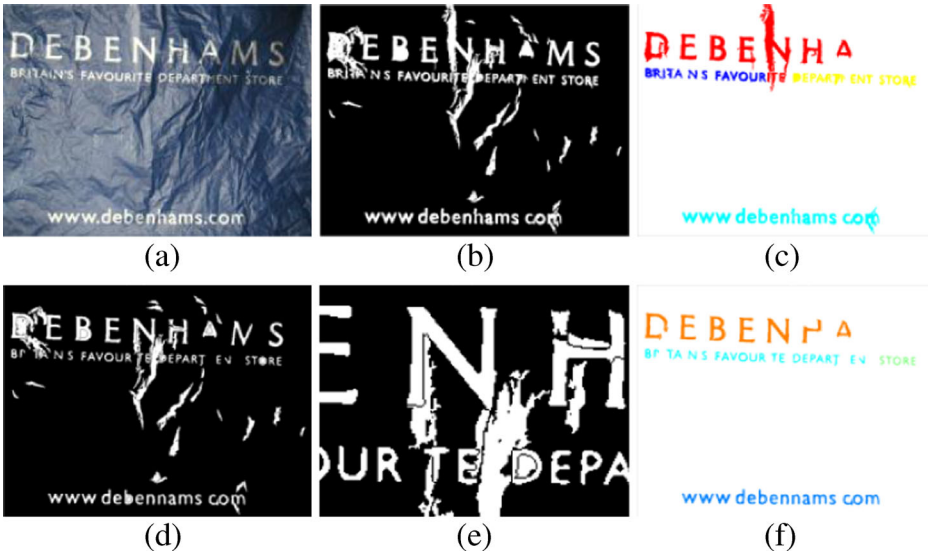


Fig. 3 **a** The original image. **b** Detected MSERs. **c** Detected text lines by using MSERs as in (**b**). **d** Image mask integrating MSERs and canny edges. **e** Magnification of part of (**d**). **f** Detected text lines by using MSERs as in (**d**)

3.2 Connected components collection

Although MSERs are very stable, their detection pays particular attention to consistent gray intensities while neglects the gradient features to some extent. Under some conditions, it may link noisy pixels to text ones with similar gray intensity. For example in Fig. 3b, many noisy pixels are connected to pixels of character "N" in the first text row in MSERs, and the noisy pixels even connect characters in the first row to those in the second row. These incorrect connections will lead to difficulties for later operations. To overcome this shortcoming, we use the canny edge in the image to serve as dam lines of those MSERs. Those pixels which are not only in the canny edge mask but also in detected MSERs are removed, and connected components (CCs) are collected in the detected MSERs with the canny edge as dam lines. In the collecting process, a pixel can only connect to its directly nearby 4 pixels which are above, below, left to and right to it respectively. This can prevent two pixels in different sides of an edge pixel from being connected. It can also be seen from Fig. 3 that our text detection method indeed benefits from the removal of pixels on the edge.

3.3 Similarity measurement

Next we want to merge similar text regions into text lines. To this end, we need to define a similarity between any two connected components. A good similarity measurement should satisfy two properties: one is that the similarity between CCs in the same text line should be large enough; the other is that the similarity between CCs in different text lines should be small enough. Supposing that $V=\{CC_1,CC_2,\ldots CC_n\}$ containing all CCs in a image, $y=[y_1,y_2,\ldots y_n]$ in which y_i denotes the text line No. of CC_i, we want to define a similarity function $\mathbb{f}$ that satisfies the condition as follows: $\forall CC_i, CC_j, CC_k \in V, (y_i = y_j \wedge y_i \neq y_k) \Rightarrow \mathbb{f}(CC_i, CC_j) > \mathbb{f}(CC_i, CC_k)$. Because CCs in the same text line usually have similar size and colour, we define two different similarities: geometry similarity and colour similarity.

3.3.1 Geometry similarity

We firstly introduce geometry distance between two CCs, and then describe geometry similarity based on the former geometry distance.

The geometry distance integrates normalized absolute distance, relevant distance and size ratio between two CCs into a single measurement. The normalized absolute distance between CC_i and CC_j is defined as follows:

$$dis_{ij}^{1} = \frac{\left|C_{x_i} - C_{x_j}\right| + k_1 \cdot \left|C_{y_i} - C_{y_j}\right|}{k_1 \cdot h_{im} + w_{im} - \left(k_1 \cdot \left(h_i + h_j\right) + w_i + w_j\right)/2} \tag{1}$$

Where C_{x_i}, C_{y_i}, C_{x_j} and C_{y_j} respectively denote the horizontal and vertical coordinates of the centroids of CC_i and CC_j. h_i, h_j, w_i, and w_j respectively denote the height and width of CC_i and CC_j, and h_{im} and w_{im} denote the height and width of the current image respectively. k_1 is a constant which controls the contribution of the horizontal distance compared to the vertical distance. In this paper, its value is 2. The numerator in Eq. 1 denotes the weighted L_1 absolute distance between two centroids, while the denominator mainly serves as a normalization factor. The value of dis_{ij}^{1} ranges from 0 to 1. When two regions are at the opposite corners of the image, it reaches the largest value of 1. Text lines are generally aligned horizontally, so the vertical distance plays a more important role than horizontal distance in distinguishing different text lines. While this observation

is reflected in Eq. 1, it can be concluded that, when the centre points of two CCs CC_i and CC_j are of the same distance from another centre point of component CC_k, except that the centres of CC_j and CC_k have the same horizontal coordinate, while those of CC_i and CC_k have the same vertical coordinate, we would consider that CC_i are farther than CC_j from CC_k. It can also be seen from Fig. 4 that characters "A" and "B" are in the same row, so the distance between them should be shorter than that between "A" and "D". However, if k_1 is set as a large value, it will prevent texts in vertical line from being clustered into the same text line, so 2 is a compromised value.

To enlarge the difference of distances between different CC pairs, the distance measurement in Eq. 1 can be modified as follows:

$$dis_{ij}^{2} = dis_{ij}^{1}\sqrt{\aleph_i(j)} \tag{2}$$

Where $\aleph_i(j)$ denotes the index number of dis_{ij}^{1} among all distances $dis_{i.}^{1}$ from the center of CC_i. For example, if the distance between the centre of CC_i and all other CC centres are 1.5, 1.7, 2.3, 4.5, and 5.6 while that of CC_j from the centre of CC_i is 4.5, then $\aleph_i(j)=4$. It can be seen from Eq. 2 that dis_{ij}^{2} has powerful distinguishing ability by setting a larger distance from CC_i to distant CCs. It can also be concluded that dis_{ij}^{2} is not equal to dis_{ji}^{2}. The motivation of Eq. 2 is that the No. in the sorted distance list brings context information for distance calculation.

When there are many characters in a text line, the distance between characters on both ends calculated by Eq. 2 will be large, which may prevent them to merge into a line. To solve this problem, we modify Eq. 2 as follows:

$$dis_{ij}^{3} = \min_{p_k}\left\{dis_{ip_kj}^{2}, |p_k| \in \{0, 1, 2, \ldots\ldots n-2\}\right\} \tag{3}$$

Where p_k denotes a path from CC_i to CC_j and the length of p_k is between 0 and n-2. dis_{ij}^{3} denotes the shortest path between CC_i and CC_j and it can be obtained by Floyd algorithm or other algorithms with similar functions. The motivation of Eq. 3 is that a good distance measurement between two objects should describe the relationship between the two objects in the context of all objects. On the other hand, the distance defined in Eq. 3 fulfils triangle inequality which is very important for a distance measurement. An example is shown in Fig. 4. Supposing that $dis_{ik}^{2} = 0.5$, while $dis_{is}^{2} = 0.2$, $dis_{st}^{2} = 0.2$ and $dis_{it}^{2} = 0.9$, it can be concluded from above that $dis_{it}^{2} > dis_{ik}^{2}$. However, CC_i and CC_t are in the same text line while CC_i and CC_k are not. So this may result in wrong text line detection if

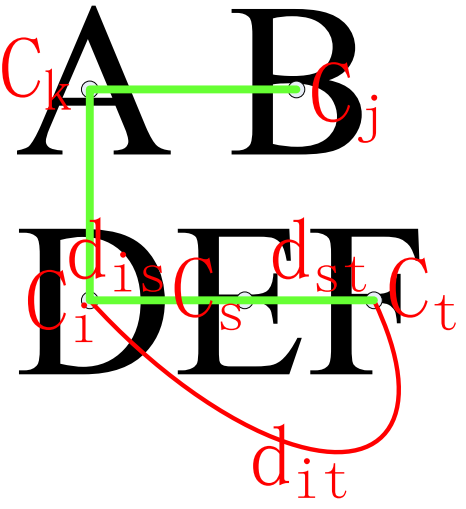


Fig. 4 Distance measurement

we use only dis^2. By using dis^3, we can get $dis^3_{it} = 0.4$ (the shortest path between CC_i and CC_t is CC_i-CC_s-CC_t) while $dis^3_{ik} = 0.5$ and $dis^2_{it} < dis^2_{ik}$ which correctly reflects the inner distance between the two pairs of CCs. Because the distance between adjacent CCs in the same text line is usually small, the distances among CCs in the same line will be further reduced by using Eq. 3.

The shape distance between two regions is defined as follows:

$$dis^4_{ij} = \sqrt{\frac{\max(h_i, h_j)\cdot\max(w_i, w_j)}{\min(h_i, h_j)\cdot\min(w_i, w_j)}} \tag{4}$$

It can be seen from Eq. 4 that two CCs with similar size will obtain a small value of dis^4_{ij}. CCs in the same text lines usually share similar height and width, so they will produce a small dissimilarity value.

At last all distance measurements defined above are integrated into a single similarity measurement as follows:

$$simi_{geometry}(i, j) = \exp\left(-\sqrt{\max\left(dis^3_{ij}, dis^3_{ji}\right)\cdot dis^4_{ij}}\right) \tag{5}$$

The similarity ranges from 0 to 1 while it can not achieve 0 and 1. The greater these distance values, the smaller the similarity. It can be seen from the above 5 equations that our geometry similarity combines sizes, normalized absolute distance, relative distance and contextual information, so it is expected that it will be powerful in characterizing text regions.

3.3.2 Colour similarity

Text CCs in the same text line usually share the same colour, so colour feature is also an important factor that should be taken into account. In this paper, we firstly convert images from RGB colour space into HSV colour space, and then H, S, and V components are quantified respectively into 8, 3, and 3 bins, leading the dimension of the colour histogram to be 72. Supposing that the color feature vector of CC_i and CC_j are C_i=[$C_{i,1}$, $C_{i,2}$,… $C_{i,t}$,… $C_{i,n}$] and C_j=[$C_{j,1}$, $C_{j,2}$,… $C_{j,t}$,… $C_{j,n}$] respectively, then the color similarity can be calculated as below:

$$simi_{color}(i, j) = \sum_{t=1}^{n} \min\left(C_{i,t}, C_{j,t}\right) \tag{6}$$

Where n=72 in this paper.

The similarity between two CCs can be finally estimated as:

$$simi(i, j) = \left(simi_{geometry}(i, j) + simi_{color}(i, j)\right)/2 \tag{7}$$

4 Candidate text line detection

Since texts are almost written in lines, so we can use the contextual information of CCs to merge similar CCs into text lines. Text line detection can be divided into two steps: sibling identification and text line identification. Sibling identification uses some heuristic rules to decide whether two adjacent CCs can be merged together, if can, we call them siblings. The heuristic rules mainly decide whether their sizes are similar and whether their absolute distance is small enough to merge. If two CCs are siblings, text line identification manages to decide whether they are in the same line. We detail the two steps in the following sections.

4.1 Sibling identification

Partly based on [31], three constraints are defined to decide whether two connected components are siblings of each other.

(1) The height ratio and width ratio of two adjacent CCs should fall between two thresholds T_1 and T_2.
(2) Two adjacent characters should not be too far away from each other in spite of various heights and widths, so the distance between two connected components should not be greater than T_3 times the width or height of the larger one.
(3) Two adjacent characters should share similar colour feature, so their colour similarity should be above a threshold T_4.

The three constraints can be formalized as follows:

$$S_{ij} = S_{ij}^1 \wedge S_{ij}^2 \wedge S_{ij}^3 \tag{8}$$

S_{ij} denotes whether two connected components CC_i and CC_j are siblings of each other. If the value of S_{ij} is 1, it denotes that CC_i and CC_j are siblings and may be in the same text line, otherwise they cannot lie in the same text line. S_{ij}^1, S_{ij}^2 and S_{ij}^3 represent the above three constraints respectively. In this paper, we set T_1=2, T_2=4, T_3=3 and T_4=0.4. It should be noticed that although our constraints are similar to those in [31], they differ in many aspects. In [31], their constraint rules work generally under an assumption that text lines are of horizontal direction and their rules were only used in their adjacent character grouping stage, but ours deal with text lines in arbitrary directions. To this end, our rules have to firstly estimate the text line orientation. Even though they also proposed methods to deal with text lines in arbitrary directions, they (text line grouping) didn't use the constraint rules. For the first constraint, we can represent it as follows in Eqs. 9 and 10:

$$\begin{aligned} h_r &= \max(h_i, h_j) / min(h_i, h_j) \\ w_r &= \max(w_i, w_j) / min(w_i, w_j) \end{aligned} \tag{9}$$

$$S_{ij}^1 = \begin{cases} 1 & \text{if } (h_r \le T_1 \&\& w_r \le T_2) \text{ and } |tg\theta| \le 1 \\ 1 & \text{if } (h_r \le T_2 \&\& w_r \le T_1) \text{and} |tg\theta| > 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

In Eq. 10, θ denotes the angle between the positive X axis and the line segment that connects centres of the two connected components CC_i and CC_j.

If the absolute value of the line slope is smaller than 1, the orientation of the candidate text line is treated to be roughly horizontal, otherwise it is roughly vertical. For the horizontal text line, the height ratio should be less than T_1 and width ratio less than T_2. In contrast, for the vertical text line, the height ratio should be less than T_2 and width ratio less than T_1.

For the second constraint, we can represent it as follows in Eq. 11:

$$S_{ij}^2 = \begin{cases} 1 & \text{if } \left(|tg\theta| > 1 \& \ \& dis_{ij} \le T_3 \cdot \max(h_i, h_j)\right) \\ 1 & \text{if } \left(|tg\theta| \le 1 \& \ \& dis_{ij} \le T_3 \cdot \max(w_i, w_j)\right) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

It means that if CC_i and CC_j are in a horizontal text line, their distance should be shorter than T_3 times the larger width, otherwise it should be shorter than T_3 times the larger height.

4.2 Text line identification

If two connected components are siblings, it only means that the two CCs have similar size, colour and a small distance between them. However, sibling CCs still have an ambiguity whether they lie in the same text line. So the next step is to find out all candidate text lines. Because the sibling identification step has considered the size, distance and colour feature of connected components, in this step, we mainly consider the central locations of CCs. It holds true that if a number of points p_1, p_2,…, p_n are in the same line l, then any line l_t created by randomly linking two points $p_i,p_j \in \{p_1,p_2,\ldots\ldots p_n\}$ has the same slope as that of the whole line l. and the reverse is also true. So we can use this property to detect candidate text lines. Given a set of centroids of connected components S_{cc}, groups of collinear character centroids are computed as:

$$C = \left\{ c \middle| c = \text{centroid}(CC) \wedge CC \in S_{cc} \right\} \tag{12}$$

$$L = L_1 \cup L_2 \tag{13}$$

$$L_1 = \left\{ G \middle| G \subseteq C, |G| \geq 3, \forall c_i, c_j, c_k \in G, l(c_i, c_j) = l(c_i, c_k) = l(c_j, c_k) \right\} \tag{14}$$

$$L_2 = \left\{ G' \middle| G' = (c_i, c_j), \exists G \in L_1, slope(G) = slope(G') \right\} \tag{15}$$

In Eq. 12, C denotes the set of centroids of all the connected components S_{cc}. In Eq. 13, L denotes the set of all candidate text lines. It includes lines from two categories: lines that contain at least three CCs and lines that contain only two CCs. L_1 in Eq. 14 denotes the set of text lines which are composed of at least 3 CCs. $1(c_i, c_j)$ denotes the line passing through c_i and c_j. L_2 in Eq. 15 denotes the set of text lines in which every text line is composed of two CCs but must be parallel to at least one line in set L_1. *slope* (G) denotes the slope of a line identified by points in set G.

To identify L_1, for every line, we firstly search its three seed CCs, and then expand it to contain more CCs. In the seed CCs searching phrase, if three lines created by linking any two from among three components CC_i, CC_j, and CC_k have the same slope, we think they are in the same line and constitute the seed CCs of the current line. After having obtained the seed CCs of a line l_i, we keep an average angle of it. Then for every remained CC_u, we obtain its K-NN(K nearest neighbouring) CCs CCS_K in the current line. If the slope angle of the line segment through it and any $CC_v \in CCS_k$ is close enough to the average angle of the current line, it is also in the current line. It should be noticed that when we calculate the K-NN CCs of a CC we use L2 distance instead of geometry distance mentioned before. The angle between two line segments c_ic_j and c_jc_k is calculated as follows:

$$\Delta\theta_{ijk} = \min\left\{ \arccos\left(\frac{v(c_ic_j)\cdot v(c_jc_k)}{\|v(c_ic_j)\|\cdot\|v(c_ic_j)\|} \right), \pi - \arccos\left(\frac{v(c_ic_j)\cdot v(c_jc_k)}{\|v(c_ic_j)\|\cdot\|v(c_ic_j)\|} \right) \right\} \tag{16}$$

where $v(c_i\ c_j)$ and $v(c_j\ c_k)$ denote the vectors of c_ic_j and c_jc_k respectively.

The average slope angle between any two line segments c_ic_j and c_mc_n are defined as follows:

$$\overline{\theta} = \begin{cases} \dfrac{\theta i_j + \theta_{mn} + \pi}{2} & \text{if}\left(\theta_{ij}\cdot\theta_{mn} \le 0 \&\& \max\left(\left|\theta_{ij}\right|, \left|\theta_{mn}\right|\right) \ge \dfrac{\pi}{4}\right) \\ \dfrac{\theta_{ij} + \theta_{mn}}{2} & \text{otherwise} \end{cases} \tag{17}$$

In the above Equation, θ_{ij} and θ_{mn} ranges in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. The angle difference between a line segment c_ic_j and a line with an average slope angle of $\overline{\theta}$ is defined as follows:

$$\Delta\theta = \min\left\{\left|-\theta_{ij} - \overline{\theta}\right|, \pi - \left|-\theta_{ij} - \overline{\theta}\right|\right\} \tag{18}$$

The reason why the minus sign appears in front of θ_{ij} is that the positive orientation of Y axis in the screen is opposite to that in the geometric coordinate system. The three centroids are approximately collinear if $\Delta\theta \le T_5$. The value of T_5 is determined as follows:

$$T_5 = \max\left(\frac{\pi}{36}, \min\left(\frac{\pi}{10}, \frac{\pi}{12} \Big/ \sqrt{\frac{d_{ij}}{\overline{d}_l}}\right)\right) \tag{19}$$

Where d_{ij} denotes the distance between c_i and c_j while $\overline{d}_l$ denotes the average distance between all centroid of CCs on line l on which all centroids are ordered from left to right or top to bottom. It can be concluded from Eq. 19 that if two CCs is far away from each other compared to the average interval distance in the line, the corresponding angle threshold should be smaller than $\frac{\pi}{12}$, while if they are very close, the threshold could be larger than $\frac{\pi}{12}$. It can also be illustrated by Fig. 5. In Fig. 5, the upper line denotes the average angle $\overline{\theta}$ of the text line. Though θ_3 is smaller than θ_2 (θ_2 and θ_3 are respectively represented as $\angle 2$ and $\angle 3$ in Fig. 5), it corresponds to a smaller Threshold than that of θ_2, so it can be correctly removed from the text line "jungle".

The framework of text line identification is shown in Fig. 6. We here describe it in details. We initialized set UL_{cc} by adding all connected components. We also initially set a label array LA_{cc} with all values as 0 in which LA_{cc}^{i} denotes whether CC_i has been merged into a line or not.. Every component of UL_{cc} denotes a CC that is not in any text line so far. For every connected component CC_i, we calculate the similarity simi(i,*) between it and all other CCs by using Eq. 7, and then the two maximum similarities are picked up and their sum is obtained and represented as $partSimi(CC_i)$. Then all $partSimi$ values are sorted into list PSL in descend order. A CC CC_i is picked squentially from list PSL, and all CCs with label value as 0 are sorted into list SCL in descend order according to the distance between themselves and CC_i For any two CCs CC_j, and CC_k (j≠k)picked from list SCL which fulfill $S_{ij}=1 \wedge S_{jk}=1$, calculate the angle difference $\Delta\theta_{ijk}$,$\Delta\theta_{jik}$ and $\Delta\theta_{ikj}$. If $\Delta\theta_{ijk} <= \frac{\pi}{12} \wedge \Delta\theta_{jik} <= \frac{\pi}{12} \wedge \Delta\theta_{ikj} <= \frac{\pi}{12}$, then create a new text line L_t, record its components $S_{cc}(L_t)=\{CC_i,CC_j,CC_k\}$ and calculate the average angle $\overline{\theta}$ by using Eq. 17, and remove the three components from set UL_{cc}. Then the three CCs serve as the seed of the current line and their label values in LA_{cc} are set as 1. For

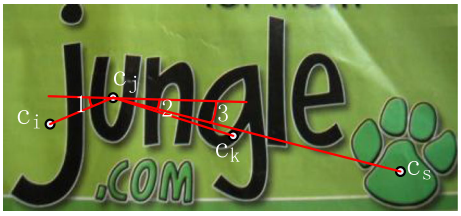


Fig. 5 Angle threshold decision

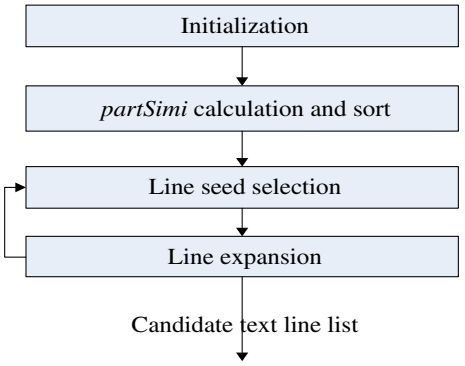


Fig. 6 The framework of text line identification

every remaining CC_m in UL_{cc}, calculate its similarity to line L_t by using $simi(CC_m, L_t) = \sum_{CC_n \in S_{cc}(L_t)} simi(CC_m, CC_n)$, and sort the *simi* in descend order. Orderly picks up a CC (CC_t) from UL_{cc}, CC_t is added to $S_{cc}(L_t)$ only if it meets the following 3 conditions:

(1) There at least exists one $CC_k \in S_{cc}(L_t)$ among the K-NN CCs $CC_1(L_t), CC_2(L_t), \ldots CC_R(L_t) \in S_{cc}(L_t)$ of CC_t which is not only the sibling of CC_t but also the angle difference $\Delta\theta$ between the two CCs and the average angle $\overline{\theta}$ of current line L_t is under T_5.
(2) CC_t is also among the K-NN CCs of the founded CC_k in Condition 1.
(3) The distance between the center of CC_t and that of the current line L_t is less than T_6.

In this paper we set the value of K as 3. T_6 is obtained as follows:

$$T_6 = \begin{cases} k' \cdot h_t & |tg\theta| \le 1 \\ k' \cdot w_t & |tg\theta| > 1 \end{cases} \tag{20}$$

In the above equation, h_t and w_t denote the height and width of CC_t respectively, θ is the angle between the positive X axis and the line that links the centers of CC_t and CC_k and k'= 1/3. The first and third conditions ensure that CC_t is in the current line while the second condition ensures that CCs in the same line are mutually similar. If the *current* CC is added to $S_{cc}(L_t)$, then the set UL_{cc}, the array LA_{cc} and the average angle of the current line are updated. We repeat the procedure until all components in UL_{cc} are processed. Then repeat to find another group of line seeds until there does not exist any group of line seeds.

For the detection of line in set L_2 in Eq. 15, if the line connecting the centres of two CCs CC_i and CC_j fulfils the following two conditions, we consider it as a candidate text line and added it into L_2: (1) It is parallel to at least one line detected in the former operation; (2) CC_i is among the K-NN CCs of CC_j and CC_j is also among the K-NN CCs of CC_i. Eventually, we obtain all candidate text lines. An example of the full candidate text line identification procedure is illustrated in Fig. 7.

5 Text line filtering

Candidate text lines obtained by the former operation may not be the real text line. So we need to filter out those false ones. To this end, we use a sparse filter to be detailed below.

[33] uses a sparse classifier in [16] to classify regular image blocks into two categories: text blocks and non-text blocks. Non-text blocks are then removed. However, the method suffered from many shortcomings as follows:

Fig. 7 Candidate text line identification. **a** The original image. **b** Detected MSERs. **c** The selected line seeds. **d** The detected candidate text lines

(1) They divided images into blocks with fixed size. In this case, their method could not work well when text sizes and intensities vary in a large range.
(2) Their method can only detect text line in horizontal direction because the method cannot obtain the slant angles of the text lines by using only a regular block.
(3) The classifier of [16] they used is not convex and does not explore the discrimination capability of sparse coding coefficients. So their method could not work well for text fonts with similar edge properties to those of non-text blocks, for example, Mistral font style.

In this paper, we develop a more powerful sparse classifier to overcome all these shortcomings: (1) we convert MSERs instead of the original images into regular blocks to overcome the first shortcoming; (2) To overcome the second shortcoming, we transform text lines in arbitrary direction into horizontal or vertical ones; and (3) To overcome the third shortcoming, we use skeleton instead of edge and a more powerful classifier. We will detail all the techniques in the following sections.

5.1 Sparse classifier

We here use the Fisher discrimination dictionary learning (FDDL) schema [27] as our sparse classifier. The reason for us to select it is that both the reconstruction error and the coding coefficient are discriminative. Compared to other state-of-the-art methods, it has competitive performance in various pattern recognition tasks.

Given the learned structured dictionary $D=[D_1,D_2,\ldots,D_c]$, where D_i is the class-specified sub-dictionary associated with class i, and c is the total number of classes. Denote by $A=[A_1,A_2,\ldots A_c]$ the set of training samples, where A_i is the sub-set of the training samples from class i. Denote by X the coding coefficient matrix of A over D, i.e. $A\approx DX$. We can write X as $X=[X_1, X_2, \ldots, X_c]$, where X_i is the sub-matrix containing the coding coefficients of A_i over D. Apart from requiring that D should have powerful reconstruction capability of A, we also require that D should have powerful discriminative capability of images in A.

The FDDL model is defined as follows:

$$J_{(D,X)} = \mathrm{argmin}_{(D,X)}\left\{\sum_{i=1}^{c} r(A_i, D, X_i) + \lambda_1\|X\|_1 + \lambda_2(\mathrm{tr}(S_W(X) - S_B(X)) + \eta\|X\|_F^2\right\} \quad (21)$$

where the first term describes the discriminative fidelity; the second and fourth terms impose the sparsity constraint, and the third term is a discrimination constraint imposed on the coefficient

matrix X; and λ_1, λ_2 and η are scalar parameters. S_W (X) and S_B (X) in the above equation are calculated as follows:

$$S_W(X) = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in X_i} (\mathbf{x}_k - \mathbf{m}_i)(\mathbf{x}_k - \mathbf{m}_i)^T \qquad S_b(X) = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T \tag{22}$$

where $\mathbf{m}_i$ and $\mathbf{m}$ are the mean vectors of X_i and X respectively, and n_i is the number of samples in class A_i.

In [27], they proposed two classification schemes, the global classifier (GC) and local classifier (LC). Because there are only two categories (text region and non-text region) and we have lots of training samples of each class, we here use local classifier.

Denote by $m_i=[\mathbf{m}_i^1;\ldots;\mathbf{m}_i^k;\ldots;\mathbf{m}_i^c]$, where $\mathbf{m}_i^k$ is the sub-vector associated with sub-dictionary D_k. Denote by $\boldsymbol{y}$ a testing sample. The coding coefficients α associated with D_i are obtained by minimizing:

$$\widehat{\alpha} = argmin_\alpha \left\{ ||\boldsymbol{y} - D_i\alpha||_2^2 + \gamma_1 ||\alpha||_1 + \gamma_2 ||\alpha - \mathbf{m}_i^i||_2^2 \right\} \tag{23}$$

where γ_1 and γ_2 are constants. The final classification rule is shown as follows:

$$C(y) = argmin_i \left(\left\| y - D_i\widehat{\alpha} \right\|_2^2 + \gamma_1 \left\| \widehat{\alpha} \right\|_1 + \gamma_2 \left\| \widehat{\alpha} - m_i^i \right\|_2^2 \right) \tag{24}$$

where $C(\boldsymbol{y})$ denotes the category label of $\boldsymbol{y}$.

5.2 Feature extraction and dictionary learning

To deal with text lines in arbitrary direction, before extracting features for sparse filtering, we firstly uniformly convert arbitrary direction text lines into horizontal or vertical ones.

For a candidate text line, its slope angle was estimated in the last section. Then we rotate the text line about its center by θ_r degrees so that it will be aligned with either the horizontal or vertical axis. The value of θ_r is set as follows:

$$\theta_r = \begin{cases} -\overline{\theta} & |\overline{\theta}| \le \dfrac{\pi}{4} \\ \mathrm{sign}(\overline{\theta}) \cdot \left(\dfrac{\pi}{2} - |\overline{\theta}| \right) & |\overline{\theta}| > \dfrac{\pi}{4} \end{cases} \tag{25}$$

where $\overline{\theta}$ denotes the average slope angle of the current line as in Eq. 17. That is to say, if the absolute value of the slant angle is lower than 45°, then the text line is rotated as a horizontal one, otherwise it is rotated as a vertical one. The idea is illustrated in Fig. 8. Figure 8a shows the original text line mask. The transformed result of Fig. 8a is shown in Fig. 8b while the skeleton of Fig. 8a is shown in Fig. 8c. Then the abundant blank row and column around the

Fig. 8 Text line transformation. **a** The original candidate text line mask. **b** Rotated text line. **c** Extracted skeletons

text line in the rotated image are removed. Finally we obtain a horizontal or vertical candidate text line.

Then for every MSER CC, we resize it to $s_{rh} \times s_{rw}$ in which $\max(s_{rh}, s_{rw}) = s_{rg}$. In this paper, $s_{rg}=32$. That is to say, we resize the CC to make the length of its maximal side as s_{rg} while keeping the height/width ratio unchanged. Then its skeleton is extracted and also enlarged so that the longer side is s_{rg}, then the skeleton is extracted again on the enlarged skeleton block and located at the centre of the regular block. Finally the regular block is transformed into a vector with a dimensionality of $32 \times 32 = 1024$, which serves as input of the sparse filter. The skeleton extraction procedure is illustrated in Fig. 9. It can be seen from above that by converting a MSER into a regular block while keeping the layout of its skeleton and alignment of centre, our method is robust to various text sizes.

In this paper, we train two discriminative dictionaries. The first dictionary provides a sparse representation for the text while the second one provides a sparse representation for the background, each of which is composed of 512 base vectors. To train the text dictionary, we choose as training samples mainly isolate machine-printed characters extracted from 36 synthesized document images. These images contain 26 English letters and 10 Arabic numbers in various fonts. We also take Chinese characters into account. For Chinese characters, the components of a character are not always connected, so the problem is more complex. To deal with this problem, we select 412 Chinese character components and 1,500 common Chinese characters for training. The 412 components contain 212 Chinese character components and radicals and other 200 commonly used components, all of which does not have isolate parts. An example is shown in Fig. 10. The components in Fig. 10b, c and f are also Chinese characters. It should be noticed that even though the character in Fig. 10b is the same as the left part of that in Fig. 10a, their layout is not the same, and our training set contains both of them.

Even though the orientation of candidate text line can be obtained, the orientation of texts in the line is challenging to estimate. It can be seen in Fig. 11a that, both the left two text lines can be rotated into horizontal ones, and the letters in them are the same. However, the orientations of texts in them are opposite. The orientation of texts in the right text line in Fig. 11a cannot be estimated easily either. To overcome this difficulty, we rotate the original image by 90, 180, or 270° for training as in Fig. 11b. We only rotated those training images which contain English letters or Arabic numbers, while keep those training images with Chinese characters un-rotated for simplicity. That is to say, in this paper, we assume that all Chinese characters stand upward.

In total, we collect 15,375 images containing English letters or Arabic numbers, 15,000 images containing both Chinese characters and components and 30,354 non-text images, each of which is the skeleton of a MSER. For Chinese character, a MSER may correspond to a part of a character, so we use a projecting method to obtain the whole character. The accuracy of the learned sparse classifier on the training set is 97.4 %.

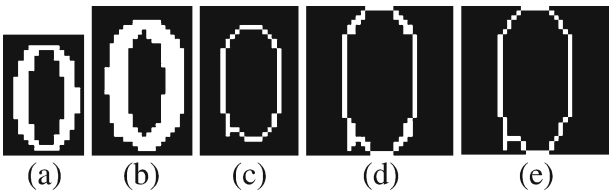


Fig. 9 Feature extraction from skeleton of MSER. **a** The original binary image. **b** Converted binary image with the maximum side resized as 32. **c** The skeleton of (**b**). **d** Enlarged skeleton. **e** The skeleton of (**d**)

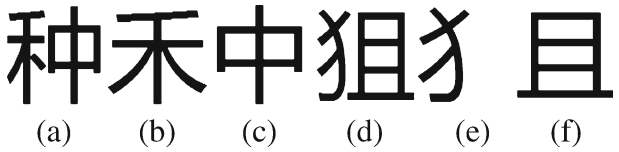


Fig. 10 Chinese characters and its components. **a**, **d** Chinese characters. **b**, **c** The components of (**a**). **e**, **f** The components of (**d**)

5.3 Filter design

The simplest filtering way is to use the sparse classifier to identify every candidate CC and remove those CCs which are identified as non-text regions. But there are three main difficulties in doing so: the first one is that the direction of current CC cannot be obtained before text line identification which may seriously influence the accuracy of the sparse classifier; the second one is that false classification of CC will seriously influence the later sibling identification and text line identification; and finally, the third one is that it is very time-consuming to identify every CC in which many MSERs are detected. So in this paper we propose to apply the sparse classifier to the identified candidate text lines only. For a candidate text line L_t and its component CCs $S_{cc}(L_t)=\{CC^t_1,CC^t_2,\ldots,CC^t_n\}$, supposing $C(y^t_i)$=identity(f^t_i) denotes the category label of the feature vector f^t_i of component CC^t_i, Then the label of the whole candidate text line can be identified as:

$$C(L_t)=\begin{cases}1 & \sum_i C\left(y^t_i\right)\geq C_T\\ 0 & \text{otherwise}\end{cases} \tag{26}$$

$$C_T = k_2 \cdot n \tag{27}$$

where k_2 is a controling parameter and n is the number of CCs in the current candidate text line L_t. In ideal condition the label of every CC can be correctly identified, so k_2 can be assumed as 1. However, in most condition the accuracy of sparse classifier can not achieve 100 %, so generally the value of k_2 is less than 1 and it can be interpreted as a lossen coefficient. The assignment of k_2 will be discussed in the next section.

6 Experimental results

In this section, we use experiments to validate our proposed method for text line detection. To evaluate our method comprehensively, we use two datasets: ICDAR and the Oriented Scene Text dataset (OSTD) which is collected by Yi et al. in [31]. The OSTD dataset can be accessed from the website of [19]. In the first dataset, there are 509 images in total, 258 images for training and 251 images for testing. In the second dataset, there are 89 scene images which contain text lines with arbitrary orientations.

Fig. 11 English letter G in different directions

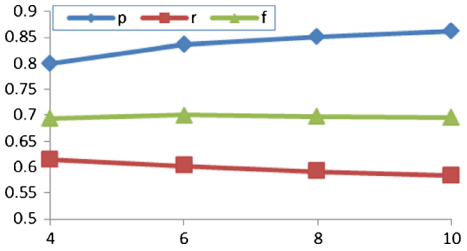


Fig. 12 Variation of p, r, and f-measure obtained under different values of Δ for MSER

For a comparative study of our method for text line detection, we use several state-of-art methods. All the experiments were implemented in MATLAB 2007 on a general purpose PC (Core 2 Duo 2.66GHZ, 4GB memory).

6.1 Performance evaluation

To evaluate the performance, we use two metrics, precision p and recall r as in [14, 15, 20]. Here, precision is the ratio of the area of the successfully extracted text regions to the area of the whole detected text region, and recall is the ratio of the area of the successfully extracted text regions to the area of the ground truth text regions. The area of a region is the number of pixels inside it. A region is not necessarily a rectangle; we use a rectangle to describe a region. A low precision means overestimate while a low recall means underestimate. To combine p and r into a single measurement, we use a standard f as follows:

$$f = \frac{2 \cdot p \cdot r}{p + r} \tag{28}$$

For a non-horizontal text line, we use the measure in the Eqs. 12 and 13 in [31]. The basic unit in these two dataset is text line rather word, because it is non-trivial to perform word partition without high level information.

6.2 Parameter selection

Two parameters mainly influence the performance significantly: parameter Δ in MSER detection and k_2 in Eq. 27. Because there is no straightforward relationship between them, in this paper, we change the value of one parameter while keeping another fixed. In MSER detection, Δ is a very important parameter which controls the step of gray threshold variation. In this paper, we perform experiments on the ICDAR dataset with different values of Δ while keep k_2=0.7. The experimental result is shown in Fig. 12. The horizontal axis denotes the value of Δ. It can be seen from Fig. 12 that when Δ is set as 6, the f-measure obtains its maximum value. So in the later experiments we set the value of Δ as 6. The conclusion in Fig. 12 can be explained as follows:

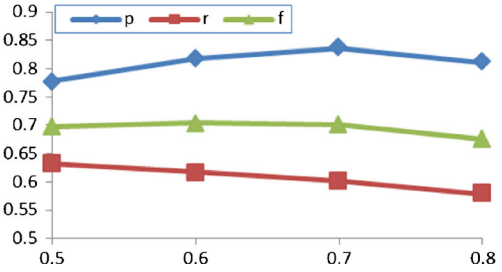


Fig. 13 Variation of p, r, and f-measure obtained under different values of k_2s for text line filtering

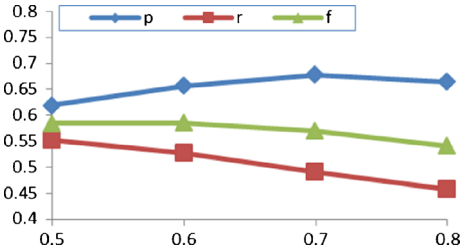


Fig. 14 Variation of p, r, and f-measure obtained under different values of k_2 by using edge feature

when Δ takes a smaller value, more candidate text regions will be detected, so the recall rate will be higher, while more non-text regions will also be collected which will reduce the precision rate. On the contrary, if Δ takes a larger value, fewer text regions will be detected, leading more non-text regions to be removed which will reduce the recall while increase the precision rate. Thus the value of 6 balances a tradeoff between precision and recall. Generally speaking, if the contrast between text and non-text pixels is low, it is better for Δ to take a smaller value. On the contrary, high contrast corresponds a higher value of Δ. To find out the best value of the parameter k_2, we also performed experiments on the ICDAR dataset with different values of k_2 while keeping Δ=6. As shown in Fig. 13, when k_2 is assigned a higher value, more text lines will be removed which will reduce the recall while increases the precision. On the contrary, more non-text regions will remain, when k_2 is smaller. In this case, the precision will be reduced while the recall will be increased. However, if k_2 is large enough, for example, 0.8 here, more text lines will be removed which will decrease both precision and recall. So k_2=0.7 is a reasonable value. To compare the edge feature with the skeleton feature, we also obtained a sparse dictionary from the edge feature of MSERs. The experimental result is shown in Fig. 14. It can be seen from Fig. 14 that the proposed method using skeleton feature can obtain higher precision and recall than those of edge feature. So the skeleton feature is better than the edge feature for the sparse classifier. There are two reasons for this conclusion: one is that for the same character, edge varies more seriously than skeleton for different font styles; the other is that skeleton is more robust than edge against the same noisy signals.

6.3 Results and discussions

The quantitative measurements of the performance of different methods for text detection are presented in Table 1. All methods are evaluated on the standard benchmark ICDAR dataset. It can be shown in Table 1 that Becker's Method [14] achieved the best recall value while our method obtains the best precison and F-measure among all methods. There are two reasons which can account for the high precision of our method. On the one hand, by using only structure or texture features, most methods did not work well when both non-text and text lines

Table 1 Performances of different text detection methods evaluated on the ICDAR test set

Algorithm	Recall	Precision	F-measure
Our method	0.60	0.84	0.70
TD-Mixture [28]	0.66	0.69	0.67
Yi's Method [31]	0.62	0.71	0.62
Epshtein's Method [18]	0.60	0.73	0.66
Becker's Method [14]	0.67	0.62	0.62
Minetto's Method [19]	0.61	0.63	0.61
Chen's Method [2]	0.60	0.60	0.58

Table 2 The performance of our mehtod on OSTD dataset

Algorithm	Recall	Precision	F-measure
Our method	0.66	0.83	0.74
Yi's method [31]	0.64	0.56	0.55
TD-Mixture [28]	0.73	0.77	0.74

have similar structures. On the other hand, our method use both the structure among CCs in the line and the intrinsic feature of every CC, so it can work well over both non-text and text lines with varying structures. Because of the executables of other methods are not available, we only evaluate three methods on the OSTD dataset by using the experiment results in their original papers respectively. The three methods are respectively our method, Yi's method[31] and TD-Mixture[28]. The performances of all three method on OSTD dataset are shown in Table 2. It

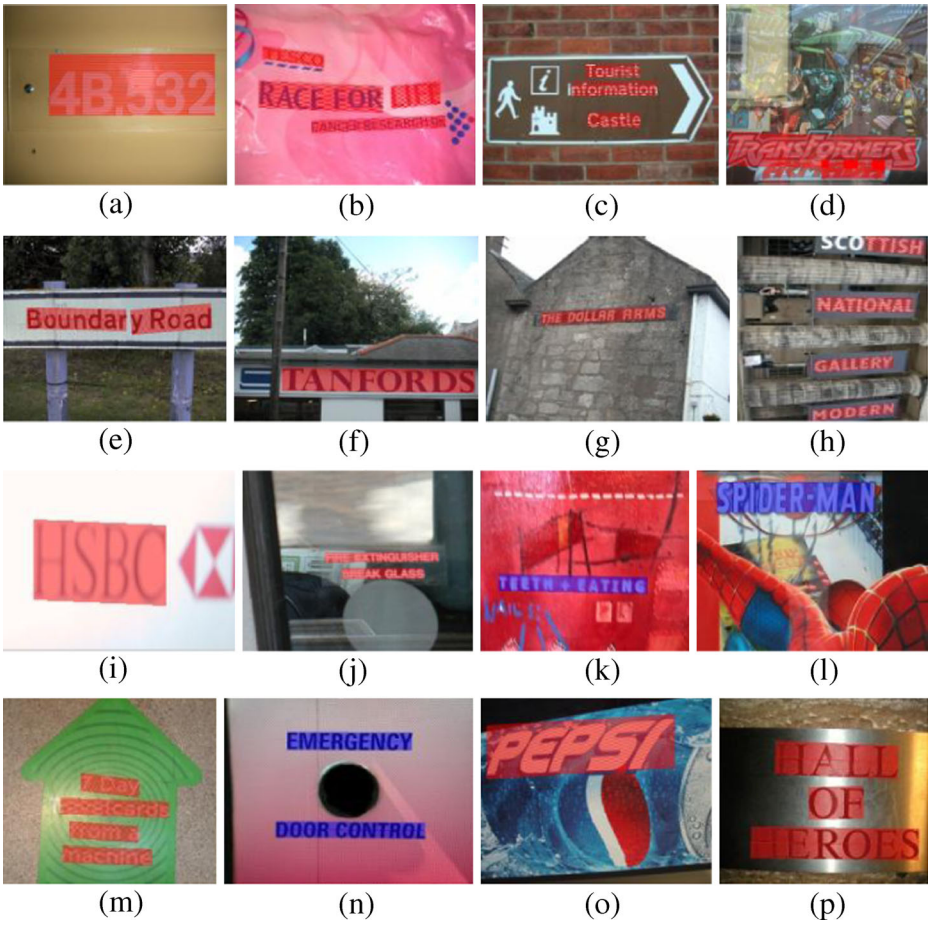


Fig. 15 Some example results of text line detection by using our method on ICDAR2003 dataset. The detected regions of text lines are marked in *red* or *blue*

can be shown from Table 2 that TD-Mixture method[28] achieve the best recall while our method can also achieve the best precision on OSTD dataset. Generally speaking, one the one hand, with several restrictive conditions, our method may discard some text lines, so our method can not obtain a relatively high recall. On the other hand, for the reason explained above, our method can get a relatively higher precision than other methods. We need to develop more flexible constraints to enhance the recall measurement in the future.

Some example results of text line detection by using our method on the ICDAR dataset are shown in Fig. 15. We use the minimum surrounding rectangle to cover the detected text regions. It can be seen from Fig. 15 that our method can detect texts varying in size, color, intensity, and orientation. In some images with strong highlights, mirror reflection or other strong disturbing factors, our method can still detect partial text lines. Our method can also detect blurred texts, which are challenging for edge-based methods such as [18].

For better comparing, we also show some results by using Yi's method [31] on the ICDAR dataset and OSTD dataset.

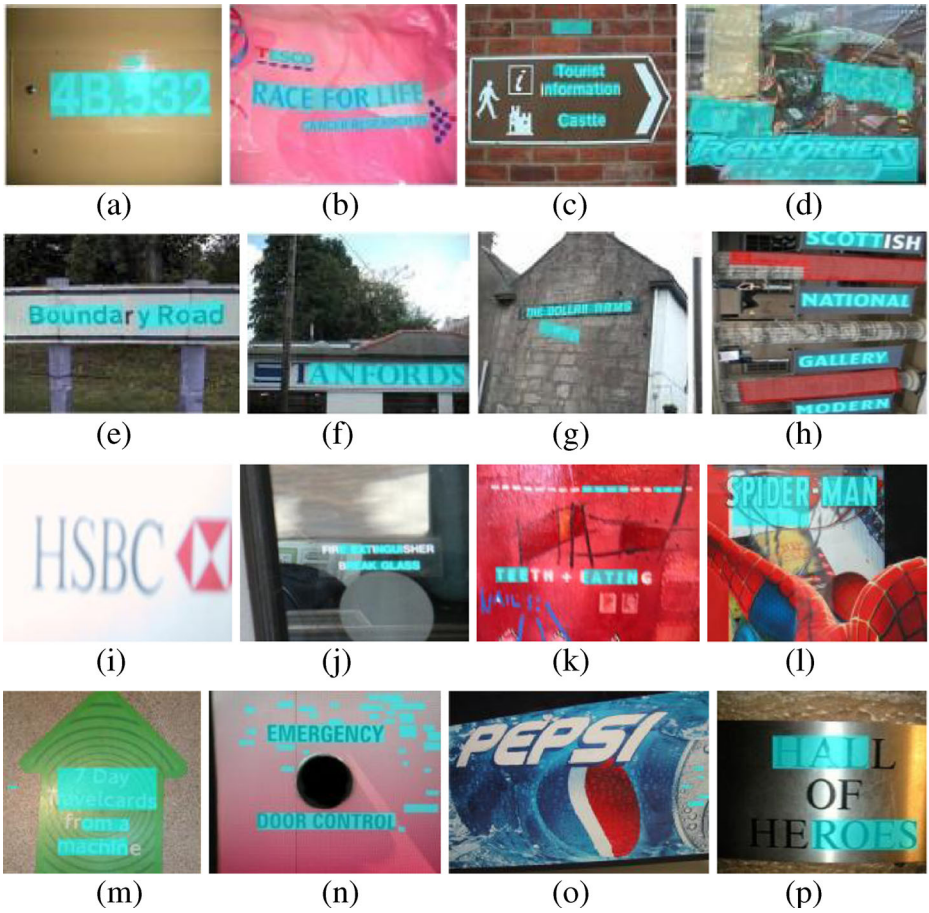


Fig. 16 Some example results of text line detection by using method in [31] on ICDAR2003 dataset. The detected regions of text lines are marked in *cyan* or *red*

Some example results of text line detection with Yi's method in [31] on the same images as in Fig. 15 are shown in Fig. 16. It can be seen from Fig. 16 that sometimes their method failed to remove non-text lines, for example, brick in Fig. 16c, toys in (d), stone in (g), pillars in (h), white blob in (k), patches of the door in (n) and the tin top in (o). It can also be seen from Fig. 16 that their method failed to detect text lines or distinguish text regions from non-text regions, for example, HSBC in Fig. 16i, SHER in (j), the spider web in (l) and PEPSI in (o), due to the similar colors in the text and non-text regions, blurred text or other disturbing factors. The reason is that the structure analysis in conjunction with the color clustering of their method is not powerful enough for removing all those non-text linear alignments of the noisy components in similar sizes.

Some example results of text line detection with our method on the OSTD dataset are shown in Fig. 17 while those with Yi's method in [31] are shown in Fig. 18. Images in Figs. 17 and 18 contain text lines with arbitrary orientations. The minimum surrounding rectangle is also used to cover the detected text regions. It can be seen from Fig. 17 that even though the orientations of text lines vary in a large range, our method still correctly identifies them in most situations. It also can be seen from Fig. 17 that under some conditions our method may miss the text line for example in the last one in the first row due to low contrast and blurred texts. The method in [31] sometimes mis-detected non-text lines as text ones, for example, the line between two timbers in Fig. 18a, the lines between two road signs in (c), and the sealing line of the chewing gum in (g). In Fig. 18b, h and l it detected a larger area than necessary.

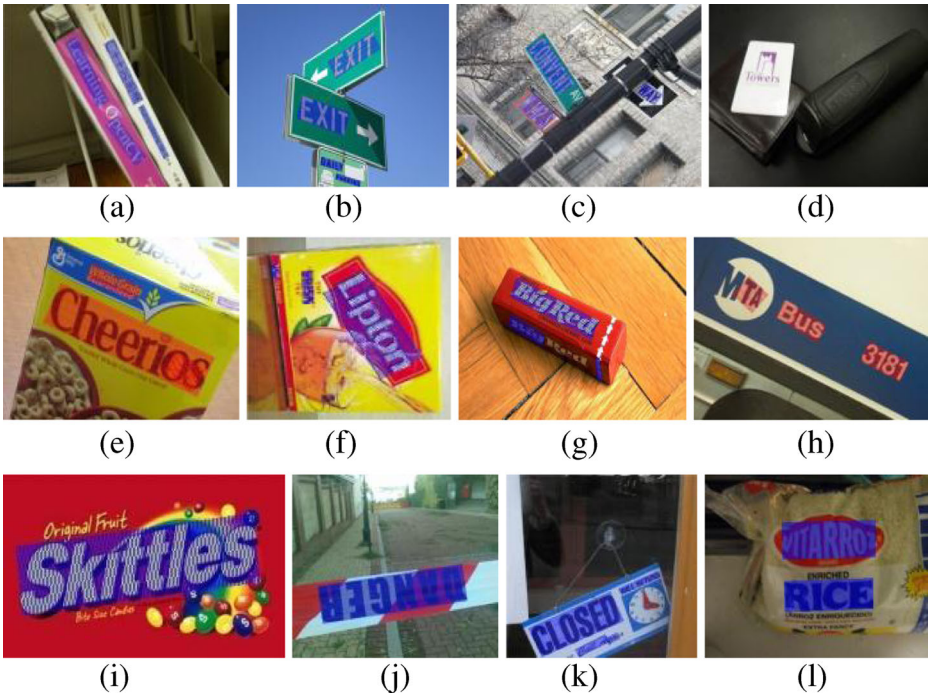


Fig. 17 Some example results of text line detection on OSTD dataset by using our method. The detected regions of text lines are marked in *red* or *blue*

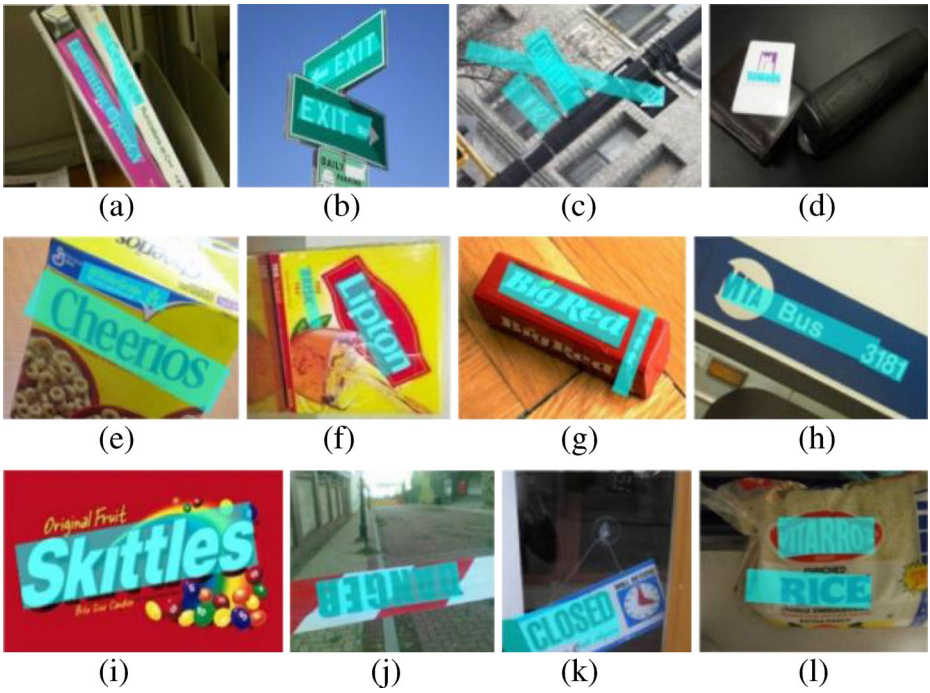


Fig. 18 Some example results of text line detection on OSTD dataset by using CT in [31]. The detected regions of text lines are marked in *cyan* or *blue*

Some poor example results of our method are shown in Fig. 19. In Fig. 19a and b, because of the transparent text or the serious light variation, our method missed many characters. In Fig. 19c, detected characters are arranged in the false text lines. The reason is that the text lines are aligned vertically, the interval between them is not large enough to separate them from each other, and their colours are very similar. In Fig. 19d and e our method failed to find text line in them due to the existence of few characters for the detection of seed CCs and low contrast between the text and its background. Clearly, these images are also challenging for the existing algorithms to detect test lines inside.

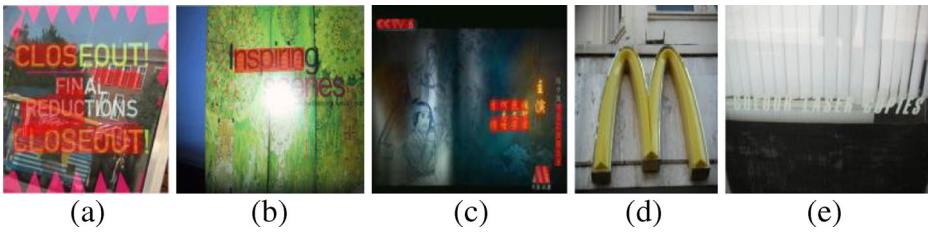


Fig. 19 Some example results of challenging text line detection. **a** Images with transparent foreground. **b** Images with highlights and low contrast. **c** Images with low contrast and dark appearance. **d** Images with few characters. **e** Images with low contrast between text and background in dark appearance

7 Conclusion

Text information in natural images is very informative for image content understanding which finds many applications in the real world. However, due to the unpredictable text appearances and complex backgrounds, it still remains a challenging problem for text detection especially in natural images. This paper proposes a novel text line detection method which is mainly based on a similarity measurement and a sparse filter. Firstly, all connected components (CC) are obtained by MSER detector with canny edges as dam lines, and then a similarity measurement is proposed to estimate the similarity between two CCs. Based on such similarity measurement, a method is developed to find out all candidate text lines. The method firstly finds three connected components (CC) as the seed CCs of a line, and then expands to contain all other CCs which are in the same line as the three seeds. Finally a sparse classifier based on the learned dictionary using a feature vector extracted from morphological skeletons in the MSERs is used to remove those non-text lines. Our method can detect text lines aligned with a straight line in any direction in natural images. A comparative study has shown that the proposed method outperformed significantly other selected state of the art methods for text line detection.

Further research will consider three different categories of text line detections: (1) those text lines which are composed by less than three text regions; (2) those text lines aligned without a straight line; and (3) text extraction from detected text regions. In this respect, the colour consistency theory may be investigated.

Acknowledgments We would like to express our sincere thanks to both the anonymous associate editor and reviewers for their constructive comments that have significantly improved the quality as well as readability of the paper.

This work is supported by the National Natural Science Foundation of China (No.60673088).

References

1. Chen T (2008) Text localization using DWT fusion algorithm. In: 11th IEEE International Conference on Communication Technology (ICCT), pp. 722–725
2. Chen X, Yuille A (2004) Detecting and reading text in natural scenes. In: the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 366–373
3. Frey BJ et al (2007) Clustering by passing messages between data points. Science 315:972–976
4. Gllavata J, Ewerth R, Freisleben B (2004) Text detection in images based on unsupervised classification of high-frequency wavelet coefficients. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), vol.1, pp. 425–428
5. Grana C, Borghesani D, Cucchiara R (2011) Automatic segmentation of digitalized historical manuscripts. Multimed Tools Appl 55(3):483–506
6. Idris F, Panchanathan S (1997) Review of image and video indexing techniques. J Vis Commun Image Represent 8(2):146–166
7. Karatzas D, Antonacopoulos A (2004) Text Extraction from Web Images Based on a Split-and-Merge segmentation Method Using Color Perception. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), vol.2, pp. 634–637
8. Kim W, Kim C (2009) A New approach for overlay text detection and extraction from complex video scene. IEEE Trans Image Process 18:401–411
9. Kimmel R, Zhang C, Bronstein AM, Bronstein MM (2011) Are MSER features really interesting? IEEE Trans Pattern Anal Mach Intell 33(11):2316–2320

10. Li Z, Liu G, Qian X, Wang C, Ma Y, Yang Y (2010) A Video Text Detection Method Based on Key Text Points. In: Processing of the 11th Pacific-Rim Conference on Advances in multimedia information processing (PCM), pp. 284–295
11. Liang J, Doermann D, Li H (2005) Camera based analysis of text and documents: a survey. Int J Doc Anal Recognit 7:84–104
12. Lienhart R (2000) Automatic text segmentation and text recognition for video indexing. Multimed Syst Mag 8:69–81
13. Liu X, Wang W (2012) Robustly extracting captions in videos based on stroke-like edges and spatio-temporal analysis. IEEE Trans Multimed 14(2):482–489
14. Lucas SM (2005) ICDAR 2005 text locating competition results. In: Proceeding of the 8th International Conference on Document Analysis Recognition, vol. 1, pp. 80–85
15. Lucas SM, Panaretos A, Sosa L, Tang A, Wong S, Young R (2003) ICDAR 2003 robust reading competitions. In: Proceeding of 7th International Conference on Document Analysis Recognition, pp. 682–687
16. Mairal J, Bach F, Ponce J, Sapiro G, Zisserman A (2008) Discriminative learned dictionaries for local image anlysis. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8
17. Matas J, Chum O, Urban M, Pajdla T (2002) Robust wide baseline stereo from maximally stable extremal regions. British Machine Vision Computing Conference, pp. 384–393
18. Ofek BEE, Wexler Y (2010) Detecting text in natural scenes with stroke width transform. IEEE Conf Comput Vis Pattern Recognit pp. 2963–2970
19. Minetto R, Thome N, Cord M, Fabrizio J, Marcotegui B (2010) "Snoopertext: A multiresolution system for text detection in complex visual scenes". In: 17th IEEE International Conference on Image Processing ICIP, pp. 3861–3864
20. Shahab A, Shafait F, Dengel A (2011) ICDAR 2011 Robust Reading Competition Challenge 2: Reading Text in Scene Images. In: Proceeding of International Conference on Document Analysis Recognition, pp. 1491–1496
21. Shivakumara P, Dutta A, Tan CL, Pal U (2010) A New Wavelet-Median-Moment based Method for Multi-Oriented Video Text Detection. In: Proceedings of the Ninth IAPR International Workshop on Document Analysis and Systems (DAS), pp. 279–288
22. Shivakumara P, Huang W, Phan TQ, Tan CL (2010) Accurate video text detection through classification of low and high contrast images. Pattern Recogn 43:2165–2185
23. Shivakumara P, Phan TQ, Tan CL (2010) New fourier-statistical features in RGB space for video text detection. IEEE Trans Circ Syst Video Technol 20(11):1520–1532
24. Shivakumara P, Phan TQ, Tan CL (2011) A laplacian approach to multi-oriented text detection in video. IEEE Trans Pattern Anal Mach Intell 33:412–419
25. Shivkumara P, Huang W, Tan CL (2008) Efficient Video Text Detection using Edge Features. In: Proceedings of 19th international Conference on Pattern Recognition (ICPR), pp. 1–4
26. Wang F, Ngo C-W, Pong T-C (2008) Structuring low-quality videotaped lectures for cross-reference browsing by video text analysis. Pattern Recogn 41:3257–3269
27. Yanga M, Zhanga L, Fengb X, Zhang D (2011) Fisher Discrimination Dictionary Learning for Sparse Representation. In: Proceeding of IEEE International Conference on Computer Vision (ICCV), pp. 543–550
28. Yao C, Bai X, Liu W, Ma Y, Tu Z (2012 June) Detecting Texts of Arbitrary Orientations in Natural Images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
29. Ye Q, Jiao J, Huang J, Hua Y (2007) Text detection and restoration in natural scene images. Vis Commun Image Represent 18:504–513
30. Yi J, Peng Y, Xiao J (2007) Color-based clustering for text detection and extraction in image. In: Proceedings of the 15th International Conference on Multimedia (MM), pp. 847–850
31. Yi C, Tian YL (2011) Text string detection from natural scenes by structure-based partition and grouping. IEEE Trans Image Process 20(9):2594–2605
32. Zhang D, Islam M, Lu G (2012) A review on automatic image annotation techniques. Pattern Recogn 45:346–362

33. Zhao M, Li S, Kwok J (2010) Text detection in images using sparse representation with discriminative dictionaries. Image Vis Comput 28:1590–1599
34. Zhao X, Lin K-H, Fu Y, Hu Y, Liu Y, Huang TS (2011) Text from corners: a novel approach to detect text and caption in videos. IEEE Trans Image Process 20(3):790–799

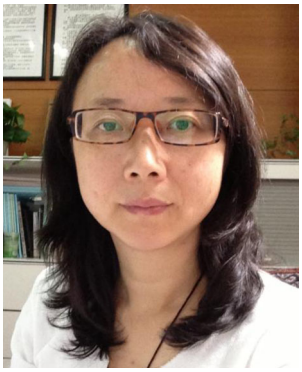
Jie Yuan received the Ph.D. degree in June 2013 in College of Computer Science at Zhejiang University (ZJU), Hangzhou, Zhejiang Province, China. Now he works in Jiangsu Electric Power Information Technology Co. Ltd., Nanjing, Jiangsu Province, China.

His current research interests are image processing, pattern recognition and information retrieval.

Baogang Wei was born in Shenyang, China. He received the M.S. degree in computer software and the Ph.D. degree in computer application from Northwestern Polytechnical University, China, in 1993 and 1997 respectively. He worked as a post-doctor at Zhejiang University, China from October 1997 to September 1999.

Since 1999, he has been being members of Chinese Association for Artificial Intelligence. He is currently a professor at college of computer science and technology of Zhejiang University. So far, he has published more than 40 papers in international conference proceedings and journals. His main research interests include artificial intelligence, pattern recognition, image processing, machine learning, digital library, and information and knowledge management.

Yonghuai Liu is now a Senior Lecturer of Department of Computer Science, Aberystwyth University. His research domain includes machine vision, object recognition, robot vision, and so on.



Yin Zhang received her Ph.D degree in computer science from Zhejiang University in 1999. Currently, she is an associate professor at the college of computer science, Zhejiang University. Her research interests mainly include multimedia information processing, pattern recognition, data mining and knowledge discovery, and knowledge engineering.

Lidong Wang was born in December 4, 1982. She received the M.S degree in computer Science from Ningbo University, and received Ph.D. degree in College of Computer Science and Technology from Zhejiang University. Her current research interests include multimedia analysis, machine learning and text mining.