Scene Text Detection Using Superpixel-Based Stroke Feature Transform and Deep Learning Based Region Classification

Youbao Tang and Xiangqian Wu, *Member, IEEE*

***Abstract*—Scene text detection is a crucial step in end-to-end scene text recognition, a greatly challenging problem in computer vision. This paper proposes a novel scene text detection method that involves superpixel-based stroke feature transform (SSFT) and deep learning based region classification (DLRC). The SSFT is developed for candidate character region (CCR) extraction, which consists in partitioning an input image into several regions via superpixel-based clustering, removing most regions based on predefined criteria satisfied by the characters, and refining the remaining regions to obtain CCRs by computing a stroke width map. The character regions are identified from the CCRs using DLRC, in which several hand-crafted low-level features, i.e., color, texture, and geometric features, and some deep convolution neural network (CNN) based high-level features are first extracted from the regions, and then these features are fused by using two fully connected networks (FCNs) for region classification. In the DLRC step, the deep feature extraction CNN and the feature fusion FCNs are jointly trained. Next, the extracted character regions are merged to form candidate text regions, from which the final scene texts are detected. The proposed method is evaluated on three publicly available datasets: ICDAR2011, ICDAR2013, and street view text. It achieves *F*-measures of 0.876, 0.885, and 0.631, respectively, which demonstrate the effectiveness of the proposed scene text detection method.**

***Index Terms*—Scene text detection, superpixel based stroke feature transform, deep learning, convolutional neural network, fully connected network, region classification.**

I. Introduction

SCENE text detection in natural images is a crucial step in end-to-end scene text recognition. Designed to locate text in various scenes (e.g., street marks and road signs), it is useful in many applications. For instance, it can be used to assist blind individuals and to highlight warning signs for pedestrians and drivers. Effective scene text detection can also enhance the performance of numerous multimedia applications, such as mobile visual search [1], content-based image retrieval [2], and semantic event detection [3]. Despite the development of numerous scene text detection methods [4]–[50] in recent years, successful text detection in different scenarios remains a challenging problem in computer vision community, due to the large variability in scene attributes (e.g., text format and color, complex and heterogeneous background).

One important step of scene text detection is candidate character region (CCR) extraction, which directly affects the final performance of the text detection. Popular existing CCR extraction approaches are based on stroke width transform (SWT) [7], maximally stable extremal region (MSER) [51] and their variants (e.g., stroke feature transform (SFT) [18]). These methods, however, remain limited by their tendency to yield false negatives, failing to detect true text regions. For this reason, the best reported recall of CCR extraction-based scene detection methods is 0.857 [49] on ICDAR (International Conference on Document Analysis and Recognition) 2013 competition dataset [52]. Hence, to improve upon this performance, it is necessary to directly extract a maximal amount of true text regions at the CCR extraction stage. In this paper, we propose a novel CCR extraction method, called superpixel based stroke feature transform (SSFT), extended from SFT [18], to achieve this goal.

Another important step of scene text detection is region classification, which aims to filter out non-text regions from the extracted CCRs. Hence, recall performance may also be improved by extracting highly discriminative features to classify text regions and non-text regions from CCRs with maximal accuracy. Many previous approaches focus on developing hand-crafted features or CNN based features to classify text and non-text CCRs. In this paper, we leverage the complementary nature of both types of features: both are extracted and fused for region classification.

In light of these two aspects, this paper proposes a novel method using superpixel based stroke feature transform (SSFT) and deep learning based region classification (DLRC) for scene text detection. This method can detect scene text in various, complex scenarios such as those illustrated in Fig. 1. The proposed method consists of two stages: character region extraction and text region detection (cf., Fig. 2). Here, a text region usually contains multiple character regions, just as a word generally contains multiple letters. In the character region extraction stage, the CCRs are extracted and classified to get the final character regions. In the text region detection stage, the extracted

Manuscript received November 30, 2016; revised April 30, 2017, October 3, 2017, and November 29, 2017; accepted January 15, 2018. Date of publication February 5, 2018; date of current version August 14, 2018. This work was supported in part by the Natural Science Foundation of China under Grants 61472102 and 61672194 and in part by ShanDong Provincial Natural Science Foundation, China (ZR2016FM04). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Leonel Sousa. *(Corresponding author: Xiangqian Wu.)*

The authors are with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: tangyoubao@hit.edu.cn; xqwu@hit.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2018.2802644

(a) (b) (c)

Fig. 1. Detection results of our method (indicated by blue rectangle) on several challenging scenarios, which nearly overlap with the ground truths (indicated by red and green rectangles). (a) Single character. (b) Text-like outliers. (c) Low contrast.

character regions are merged to form candidate text regions (CTRs), from which, the final text regions are detected.

The main contributions of this work are the following: 1) a novel superpixel based stroke feature transform (SSFT) is proposed for candidate character region extraction. Compared with traditional methods (e.g., those based on SWT or MSER), the proposed method effectively extracts most true character regions and excludes most background regions; 2) a deep learning based region classification (DLRC) algorithm is designed for character and text region classification, which extracts and fuses a set of highly discriminative hand-crafted features and deep CNN based features using a deep learning framework; 3) the proposed method achieves good performance on three standard benchmark datasets. Experimental results reveal that the hand-crafted features and CNN-based features are indeed truly complementary, and that their fusion can boost the performance.

II. Related Work

Ye and Doermann [53] summarize most existing scene text detection approaches, which fall under three main categories: sliding window, connected component, and proposal based approaches.

Sliding window based scene text detection approaches [4]–[6], [10], [13], [25], [31], [34], [43] detect text in scene images by sliding numerous windows of varying scales across all positions of an image. Various types of features are extracted from the window regions, which are then labeled as text or non-text by a trained classifier. Most of these approaches [4]–[6], [10], [13] extract hand-crafted low-level features, such as SIFT and HOG. Convolutional neural networks (CNNs), known to be very successful in many computer vision tasks, are also successfully employed in scene text detection applications [25], [31] to extract deep high-level features. The main advantage of sliding window based approaches is that the candidate regions can be easily extracted. However, it faces the following challenges: 1) the classification of large numbers of sliding windows requires extensive computation capacity, and 2) pixel-level texts cannot be directly obtained.

Connected component based scene text detection approaches [7], [11], [12], [14], [15], [18], [20], [21], [23], [24], [27]–[30], [36], [38]–[40], [45], [49] first group similar pixels (based on attributes such as intensity, color, stroke width, and texture) into connected components,, and later classify them as text or non-text based on the extracted hand-crafted or deep CNN based features. Stroke width transform (SWT) [7] and maximally stable extremal regions (MSER) [51] are two popular connected component generators successfully applied to scene text detection. Compared with sliding window based approaches, connected component based approaches are more efficient (since much less candidate regions need to be classified), robust to scale variation and effective for pixel-level text segmentation. However, the performance of SWT [7], [15], [18], [20], [29] and MSER based approaches [12], [14], [21], [24], [27], [28], [30], [36], [38]–[40], [45] are directly affected by the edge detectors and MSER extractors they use, which are sensitive to noise.

Proposal based scene text detection approaches [37], [41], [42], [44], [50] first use CNNs or some detection mechanisms (e.g., edge boxes [54]) to extract a number of region proposals, and then filter out non-text proposals. The proposals are extracted as candidate regions using a reliable and robust technique such as CNN. Therefore, proposal based scene text detection approaches are robust to variable scenarios and achieve state-of-the-art performance. These methods require a large number of samples to train their proposal extractors. However, the training samples of scene text images are too small and cannot represent a wide enough range of examples. Therefore, these approaches may fall short in the face of new, unseen scenarios.

III. Character Region Extraction

A. Superpixel Based Stroke Feature Transform

For candidate character region (CCR) extraction, Huang *et al.* [18] propose a stroke feature transform (SFT) algorithm, which first detects edges in the input image and then generates a stroke width map and a stroke color map with pixel-wise comparisons. Although CCR extraction is effective in text detection, it is time consuming and is sensitive to noises. To address these problems, this paper proposes a superpixel based stroke feature transform (SSFT) method for CCR extraction. The SSFT is composed of three steps: superpixel segmentation and clustering, background region removal, and region refinement.

1) Superpixel Segmentation and Clustering: In this work, the input image $I \in \mathbb{R}^{M \times N}$ (as shown in Fig. 3(a)) is first resized to a fixed height of 960 and a width that preserves the original aspect ratio when its height is larger than 960, and then smoothed with an edge preserving filter [55]. Next, it is over segmented into K superpixels (as shown in Fig. 3(b)) using the simple linear iterative clustering (SLIC) algorithm [56], which clusters pixels in the combined color and image plane space to efficiently generate compact, nearly uniform superpixels. For some images with very low resolution and low contrast, under segmentation of some characters still occurs. Here, instead of fixing superpixel size, we simply fix the number of superpixels per image. Due to the variable size and content of the images, superpixel size may vary.

As in [57], a color descriptor for a superpixel S is obtained by concatenating the colors in several color spaces. In this work, the color descriptor is defined as follows:

$$f = (r, g, b, l, a, b, h, s, v) \tag{1}$$

where (r, g, b), (l, a, b) and (h, s, v) are the means of the colors of the pixels in S in the RGB, CIELab and HSV color spaces,

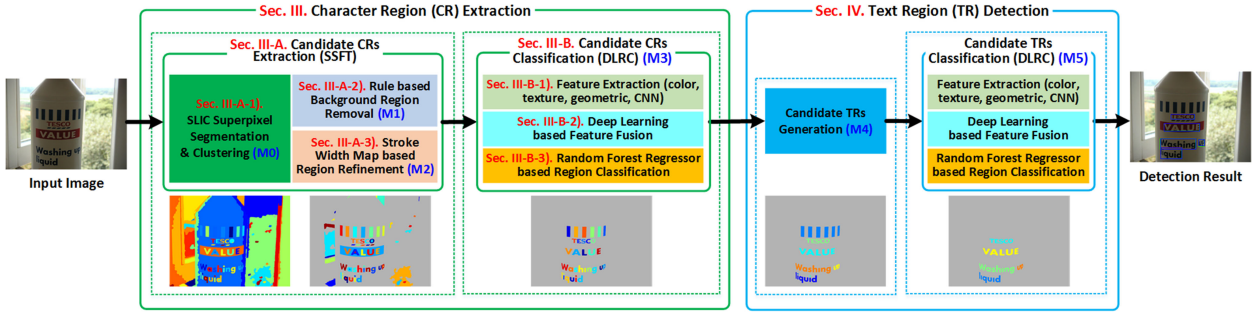


Fig. 2. The framework of the proposed method.

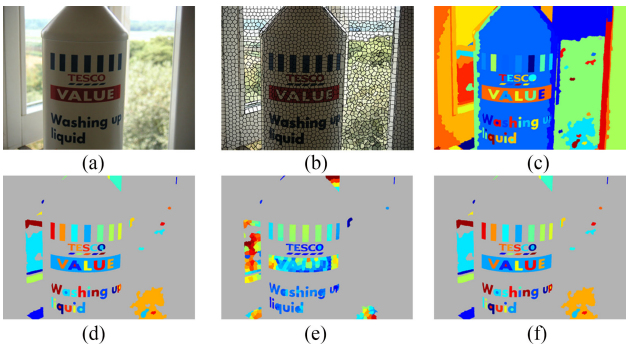


Fig. 3. The process of SSFT. (a) Original image. (b) Superpixel segmentation result. (c) The regions generated by superpixel clustering. (d) Background regions removed image. (e) Stroke width map. (f) Candidate character regions.

respectively. Each component of f is linearly normalized into the interval $[0, 1]$.

A distance matrix is constructed by computing the Euclidean distance between each pair of superpixels. Based on this distance matrix, the superpixels are clustered using the average linkage hierarchical clustering algorithm [58]. The clustering process is much faster for superpixels than for ordinary pixels, as they are much fewer in number in the image. Further details on this clustering algorithm can be found in [58]. The number of classes is automatically determined by grouping all superpixels of similar colors into clusters. In this work, the hierarchical clustering procedure terminates when the inconsistency between the classes falls below a predetermined threshold. As a result, the number of clusters differs for each image. Superpixels within the same class may form multiple subregions. Overall, the superpixel clusters partition the original image into different regions. Fig. 3(c) illustrates the image in Fig. 3(b), partitioned via superpixel clustering. Different colors correspond to different regions.

2) Background Region Removal: Given an image $I \in \mathbb{R}^{M\times N}$, a fast edge detection algorithm [59], which predicts local edges by applying structured random decision forests, is used to directly extract an edge probability map $EPM \in [0, 1]^{M\times N}$ and a gradient orientation map $GOM \in [-\pi, \pi]^{M\times N}$, in which the value associated with each pixel represents its probability of being be an edge point and its gradient orientation in the original image, respectively. The edge image E of I is extracted by applying non-maximum suppression to the EPM along the pixels' gradient orientation followed by binarization with a low threshold (here the threshold is set as 0.1). Next, the distance map of I, denoted as DM, is generated by applying a distance transformation on the edge image E based on the Euclidean distance. Let R and (w, h) denote a region and the size of its bounding rectangle, $\#(R)$ the number of the pixels in R, $sum(DM_P)$ the sum of the distance values of the pixels on the boundary of R obtained from the distance map DM, and $\#(P)$ the number of the pixels on the boundary of R. The expression $sum(DM_P)/\#(P)$ is the average distance from the pixels on the boundary of R to the edges.

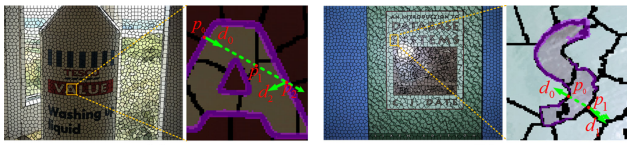
With these definitions, the following criteria for the characters in the scene can be used to remove most of the background regions:

1) The text in the scene image should not be too large. More precisely, if R is a character region, both w/M and h/N should not exceed a given threshold (here set to 0.85).
2) A character region R should be large enough to be seen, i.e., $\frac{\#(R)}{w\times h}$ it should exceed a threshold (here 0.2).
3) The contrast between the characters and background should be strong enough to distinguish text from the background, which means that the contours of the characters should be extracted in the edge image E. Considering that some unexpected interferences such as noise may result in the shift of the boundary of R from the true contour of the characters, we set the maximum value of the average distance of the pixels on the boundary of R to 6, i.e., $\frac{\mathrm{sum}(DM_P)}{\#(P)} \leq 6$.

A region R that fails to satisfy any of the 3 criteria is considered as background. We note that the thresholds provided in above have been selected to preserve approximately 98% of true character pixels in the remaining regions. Application of these criteria effectively remove most background regions, as shown in Fig. 3(c) and (d).

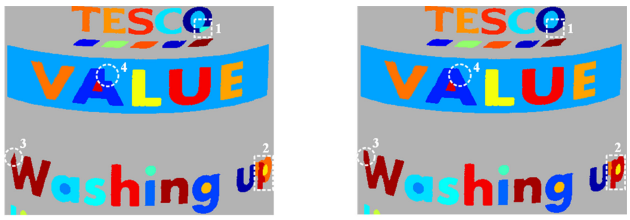
3) Region Refinement: After the background region removal phase, there still remain some characters and background regions which may be overly or incorrectly segmented (e.g., the character "O" in the word "TESCO" and part of the character "A" in the word "VALUE", in Fig. 3(d)). Since the stroke width and color of the same character generally remain consistent in scene images, we can use this information to further refine the regions.

For a contour point p_0 (see the red point in Fig. 4) of the remaining regions, we draw a ray L (see the green dotted arrow in Fig. 4) along its gradient orientation or the opposite gradient orientation to pass through the region. Denote the intersection



(a) (b)

Fig. 4. Two examples of mate-points searching by SSFT. The red points are the searching pixels, the green solid arrows illustrate the gradient orientations, the green dotted arrows are the directions for searching, the black lines are the contours of the superpixels, and the purple pixels are the ones whose edge probability is equal or larger than 0.1 in EPM.



(a) (b)

Fig. 5. The examples of region refinement. (a) The original regions. (b) The refined regions.

points of L and the boundaries of superpixels as $p_1, p_2, \ldots, p_n$, listed in ascending order according to their distances to p_0. We search the corresponding point of p_0 on the same stroke (called the mate-point of p_0) from p_1 to p_n. If a given point $p_i (i = 1, \ldots, n)$ satisfies one of the following constraints which extend the ones in SFT [18], we end our search and take p_i as the mate-point of p_0:

a) Edge constraint:

$$m_i > 0.1 \quad \text{and} \quad ||d_i - d_0| - \pi| < \frac{\pi}{2} \tag{2}$$

where m_i is the edge probability of p_i, which can be obtained from the edge probability map EPM. d_0 and d_i are the gradient orientations of p_0 and p_i, which can be obtained from the gradient orientation map GOM. This constraint ensures that the mate-point of p_0 has a high probability of being an edge point and has gradient direction nearly opposing that of p_0.

b) Color constraint:

$$f_i - f_0 > t_C \quad \text{and} \quad ||d_i - d_0| - \pi| < \frac{\pi}{6} \tag{3}$$

where f_i and f_0 are the 9-D color descriptors defined by (1) of the superpixels to which p_i and p_0 belong, and t_C is a threshold to control the color difference between two superpixels. d_0 and d_i are the gradient orientations of p_0 and p_i obtained from the gradient orientation map GOM. This constraint makes sure that the pixels' colors in the same stroke do not change too much. If a color discontinuity is detected, we consider the current pixel as an edge pixel and check its gradient orientation, as in the Step 1, with a stricter threshold.

If no $p_i (i = 1, \ldots, n)$ satisfies these constraints, we no longer consider p_0 as a contour point. According to the above descriptions, as in SFT, we end the search after finding the first point which satisfies the constraints. It is preferable not to continue the search until the best point, the one that most perfectly meets all criteria, for the following reasons: 1) Finding the best point requires comparison of all intersection points along the ray, which is far more time intensive than finding the first one. 2) In general, the color inside a letter varies smoothly and the strokes are axisymmetric, therefore the first one usually is the best (correct) one.

Fig. 4 provides two examples that illustrate how these two constraints are applied to search for the mate-points. The purple color in the right enlarged parts indicates that the corresponding pixels' edge probability is equal or larger than 0.1. As shown in Fig. 4(a), a ray is drawn from a region contour pixel p_0 along the direction d_0. The first intersection point of the ray and the superpixel's contour is p_1. Since p_1 is not colored in purple, which means that $m_1 < 0.1$, it does not meet the first constraint. Furthermore $f_1 - f_0 = 0.07 < 0.3$, hence it also violates the second constraint. Therefore, we continue the search and consider the next intersection point p_2. Since p_2 is purple, which means that $m_2 > 0.1$, and $|(\,|d_2 - d_0\,| - \pi)| = \frac{\pi}{3} < \frac{\pi}{2}$, it satisfies the first constraint. We thus end the search and retain p_2 as the mate-point of p_0. In Fig. 4(b), for p_0, the first intersecting point is p_1. Since p_1 is not colored in purple, which means that $m_1 < 0.1$, p_1 doesn't satisfy the first constraint. However, $f_1 - f_0 = 0.37 > 0.3$ and $||d_1 - d_0| - \pi| = \frac{\pi}{12} < \frac{\pi}{6}$, i.e., p_1 satisfies the second constraint. Therefore, p_1 is the mate-point of p_0.

After finding the corresponding point p_i, we obtain a line segment $p_0 p_i$. We set the stroke width values of all pixels on this line segment as the length of the line segment $p_0 p_i$. If a pixel belongs to more than one line segment, its value is set as the shortest length of these line segments.

After repeating the above process for all remaining regions' contour pixels, we compute the mean of stroke widths in a superpixel, set it as the stroke width value of all pixels in this superpixel, and form a stroke width map, as shown in Fig. 3(e). Comparing Fig. 3(d) and (e), we see that the over segmented part of character "O" and the incorrectly segmented part of character "A" have stroke width values similar to those of the other parts of the same characters in the stroke width map. Therefore, the stroke width information is beneficial for these region refinements.

Based on the extracted stroke width map, we construct a 10-D feature vector to represent a region or superpixel, which is composed of the 9-D color descriptor and the average stroke width of the pixels in this region or superpixel. The feature vector is normalized to have magnitude 1. The distance between two regions or superpixels is defined as weighted Euclidean distance between their feature vectors. Through extensive experiments on the training dataset, we find that the best performance is obtained by setting the weight as 0.7 for the stroke width component and 0.3 for the color components when computing the weighted Euclidean distance. We refine the remaining regions based on their feature vectors as follows:

Given a region R, for each adjacent region R', if the distance between R and R' falls below a threshold t, we merge R and R'. This process is iterated until all possible regions have been merged. Regions 1 and 2, indicated by white rectangles in Fig. 5, are two examples of region merging. Next, we apply a similar

process to move adjacent superpixels within a region. If the distance between a region and its adjacent superpixel is less than a threshold t, move the superpixel from its original region to this region. Two examples of this superpixel displacement are illustrated in Fig. 5 (encircled regions 3 and 4). The final candidate character regions (CCRs) of Fig. 3(a) are shown in Fig. 3(f). One can see, upon comparison with Fig. 3(c) that contains all background regions, that some background regions are removed and some regions are merged in Fig. 3(f).

For each CCR, we also keep the whole region enclosed by its bounding rectangle in the original image, called CCR bounding rectangular region, for subsequent feature extraction and classification.

The main difference between SFT and SSFT lies in their computing elements. The computing element of SSFT is a superpixel while that of SFT is a pixel, which confers the following advantages on SSFT:

1) Lower computational cost: 1) SSFT performs the searching process on the contour pixels of a small number of remaining regions (see Fig. 3(d)), while SFT searches a large number of edge pixels detected by canny operator on the original image. 2) For each pixel, SSFT searches in one direction, while SFT searches in two. This is due to the fact that in SFT, the text location is unknown, while for SSFT, it is sufficient to search along the direction pointing inside of the regions. 3) When searching for the mate-point of a given point, SSFT only considers the intersecting points of the line segment determined by these two points and the superpixel boundaries, while SFT considers all pixels on the line segment.
2) Greater robustness to noise: In SSFT, the color comparison in the second constraint is performed based on the superpixel's mean color, while in SFT, it is based on each individual pixel's color. As expected, the average color of a superpixel or region is far more robust to the noises than the color of a single pixel.

B. Deep Learning Based Region Classification

As shown in Fig. 3(f), the candidate character regions still contain background regions. To further remove these background regions, we propose a deep learning based region classification (DLRC) algorithm, in which hand-crafted features and deep CNN based features are extracted and fused for region classification.

1) Feature Extraction: Four types of features are extracted for region classification: color features, texture features, geometric features and deep features. The first three are hand-crafted low-level features and the fourth consists of deep CNN based high-level features.

1) *Color features:* Color is one of the most important cues in distinguishing text from background. We compute three color feature vectors according to a candidate region and its bounding rectangular region.

The first color feature vector is composed of the average colors of the candidate region in RGB, CIELab, and HSV color spaces. It is a 9-D color feature vector.

The second color feature vector is the histogram of the color distribution computed from the CCR in CIELab color space. For each color component, we quantize its range into 16 intervals, and count the occurrence frequency of each interval by quantizing color values of all pixels in this region. Hence, by processing all three color components, we obtain a 48-D histogram, i.e., a 48-D color feature vector.

The third color feature vector is the histogram of the color distribution computed from the background region in the bounding rectangular region of the CCR by the same method as the second one. This color histogram also forms a 48-D color feature vector.

1) *Texture features:* Sometimes, the texture of the bounding rectangular regions of true characters differ from those of the background regions, especially for the complex scene images. We can thus extract the texture features to distinguish the character regions from background regions. The histogram of gradients (HOG) [60] is an effective texture descriptor to capture the appearance characteristic of a region. In this work, we use the HOG version implemented by Felzenszwalb *et al.* [61] to extract texture features from the bounding rectangular region of CCR. A 31-D texture feature vector is extracted.
2) *Geometric features:* Compared to other objects, text has distinct geometric properties. Therefore, in this work, we extract the following geometric features to distinguish character regions from background. 1) the ratio of the number of pixels in candidate region to the area of the candidate region's boundary box; 2) the aspect ratio of the candidate region's boundary box; 3) the ratios of the width and height of the candidate region's boundary box to the width and height of the input image (since the texts usually are not too large or too small in scene images); 4) the ratio of the number of pixels of the CCR on the image boundaries to the total number of the pixels on the contour of the CCR (since text portions are seldom connected to image boundaries); 5) the mean and variance of the stroke widths of the pixels in the CCR. This yields a 7-D geometric feature vector.
3) *Deep features:* Inspired by successful deep learning methods in scene text detection [25], [27], [31], [50], we use a convolution neural network (CNN) to learn deep high-level features from the bounding rectangular region of CCRs. As in [50], this work uses a CNN model adapted from VGGNet-16 [62] by only keeping its first three blocks for deep feature extraction. The first block includes two convolutional layers (denoted conv1-1 and conv1-2), the second block includes one max pooling layer (denoted pooling1) and two convolutional layers (denoted conv2-1 and conv2-2), and the third block includes one max pooling layer (denoted pooling2) and three convolutional layers (denoted conv3-1, conv3-2 and conv3-3). The max pooling (i.e., pooling1 and pooling2) operation is performed over a 2×2 window with stride 2. All convolutions are operated with 3×3 kernels, stride 1 and pad 1. The numbers of convolutional kernels in the three blocks are 64, 128 and 256, respectively. Instead of fully

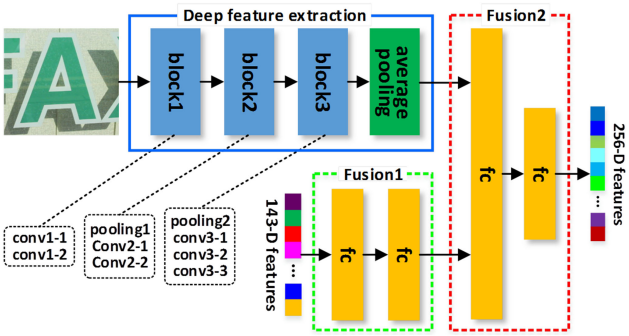


Fig. 6. The process of feature extraction and fusion with deep learning framework (Strategy1).

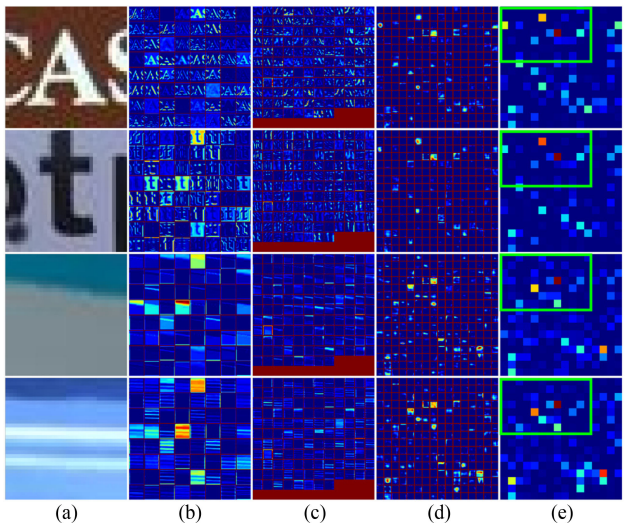


Fig. 7. Four examples of visual illustration of different feature maps. (a) Input images. (b)–(e) The feature maps of conv1-2, conv2-2, conv3-3, and average pooling in Fig. 6.

connected layers (fc), a global average pooling layer following the third block is used to generate a feature vector with fixed length. Since the third block outputs 256 feature maps and each feature map generates a value with the global average pooling, the length of the feature vector is 256. Finally, a softmax loss layer with two outputs is used to compute the loss during training. The whole architecture of the deep feature extraction CNN network is shown in the blue rectangle of Fig. 6. This CNN, is first trained on an image dataset. Next, the given region of interest is fed into the trained CNN, and the 256 outputs of the global average pooling layer are taken as components of its 256-D deep feature vector. The bounding rectangular region of CCR is first enlarged to a square area with size 32×32, in accordance with the region constructing scheme proposed by Tang *et al.* [50]. Next, the enlarged region is fed to the CNN for training and feature extraction.

To visualize the feature extraction process, we provide four examples that display different feature maps of different layers in the CNN model, i.e., conv1-2, conv2-2, conv3-3 and global average pooling, as shown in Fig. 7. From Fig. 7, we can see that 1) many details of input images are reflected in the features extracted by shallow layers (e.g., conv1-2, as shown in Fig. 7(b)). In the deeper layers of the model, these details are disappeared, and the extracted features become more and more similar for different text regions or background regions (e.g., compared with Fig. 7(c), the feature maps in Fig. 7(d) are more similar in the first two rows and the last two rows). 2) The features extracted by the global average pooling layer are similar for different text regions or different non-text regions, but significantly different for text and non-text regions, e.g., the places illustrated by green rectangles in Fig. 7(e). Therefore, the features extracted by the global average pooling layer are highly discriminative for text region classification.

2) Deep Learning Based Feature Fusion: The extracted high-level deep features effectively capture the global and contextual information of texts, instead of the low-level information (e.g., color and texture). However, this low-level information is crucial to distinguish text from background. Hence, in this work, low-level hand-crafted features also are extracted. Next, these features are fused with high-level deep learning features for region classification. The simplest way to use the information from all features is to directly concatenate all of the features to form a vector for region classification, in which the components of the feature vector are almost considered independently. In this work, we not only consider the relationship between components of hand-crafted features but also automatically learn the complex dependencies of hand-crafted features and CNN-based features by using fully connected layers and a supervised learning scheme. For hand-crafted feature fusion, we first normalize the hand-crafted feature vectors to have magnitude 1, and then feed them into a neural network consisting of two fully connected layers (fc) with 256 and 128 nodes respectively, and a softmax loss layer with two outputs for training, as shown in the green dot rectangle of Fig. 6. After training this network, the 128 outputs of the last fully connected layer are used as the initial fused features. To learn the dependencies between CNN-based features and the initial fused features, we concatenate them, normalize them to unit magnitude, and then feed them into a network with two fully connected layers (fc) with 512 and 256 nodes respectively and a softmax loss layer with two outputs for training, as shown in the red dot rectangle of Fig. 6. The whole network architecture of the feature fusion framework (Fusion1 and Fusion2) is shown in Fig. 6. This network is used to learn the deep relationship of the features and trained to maximize its regression accuracy. The fusion framework described above is denoted as Strategy1. After training this network, we use the outputs of the last fully connected layer as a 256-D feature vector (final fused features) to represent the property of each input image during the testing stage.

In our preliminary experiments, we also tested two other fusion strategies: 1) Unlike the method in Strategy 1, the deep features are directly concatenated with the hand-crafted features instead of the initial fused features (denoted as Strategy2, as shown in Fig. 8(a)). 2) Building upon Strategy 1, an additional fully connected layer with 512 nodes is added to learn more complex feature transformation (denoted as Strategy3, as shown in Fig. 8(b)). We found that Strategy3 outperforms Strategy2, but fails to meet Strategy1's performance level, even with its

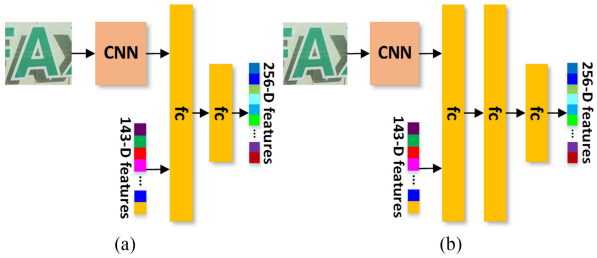


Fig. 8. Another two fusion strategies. (a) Strategy2, (b) Strategy3.

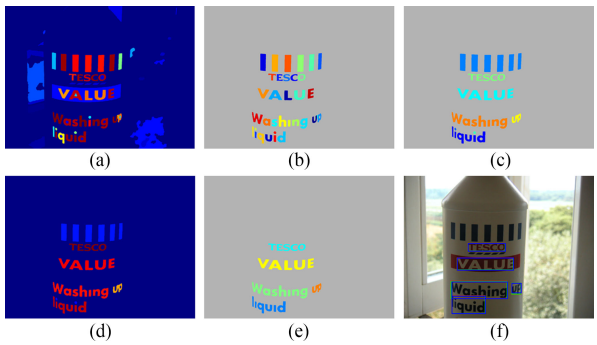


Fig. 9. The process of text detection. (a) Character confident map. (b) Extracted character region. (c) Candidate text regions. (d) Text confident map. (e) Detected text. (f) The image with the detected texts (indicated by blue rectangle) and the ground truths (indicated by red rectangle for ICDAR2011 and green rectangle for ICDAR2013). In (a) and (d), the colors of pixels are more red means that they have higher confidences of being text.

usage of more parameters. Therefore, we choose Strategy1 as the fusion framework shown in Fig. 6 for feature fusion in this work.

There are three networks: the CNN used for deep feature extraction, the fully connected network used for hand-crafted feature fusion and the fully connected network used for fusion of the initial fused features and deep features, as shown in Fig. 6. After training them separately, we use the resulting model parameters as initial weights and jointly train the three networks to improve the final performance.

3) Region Classification: After performing the above processes, a 256-D feature vector is extracted for each region. To further improve the regression accuracy, a random forest regressor [63] is trained on the training dataset. We initially train a softmax regressor jointly with the network. Next, we use the fused (low level hand-crafted + high level deep learning) features to train other regressors, including random forest and SVM. Through several experiments, we find that the random forest regressor yields the best performance, and use it for our study. The trained random forest regressor is used to compute a confidence score for each region, as shown in Fig. 9(a). From Fig. 9(a), most character regions receive much larger confidence scores (redder) than background regions. Finally, we segment the confidence map to obtain the character regions as shown in Fig. 9(b), by using multiple-level Otsu's algorithm [64].

At present, the two main parts (i.e., candidate region extraction and classification) are disjoint, and cannot be optimized in an end-to-end fashion. Indeed, the hand-crafted operations cannot be easily replaced with CNN-based computations. In the future, we hope to resolve these issues by designing a CNN model that simultaneously performs text region detection and classification.

IV. Text Region Detection

In the candidate character region extraction stage, only a single character's property is considered, which may lead to errors. For example, the regions at the top row of Fig. 9(b) are mistakenly extracted as character regions since they resemble the characters 'l' and 'I'. We thus address this problem by exploiting such relationships between characters. In this section, we generate candidate text regions and classify them as text or non-text.

Given a set of character regions extracted from an input image, we obtain their bounding boxes denoted as $B'_i = (x'_i, y'_i, w'_i, h'_i), i = 1, 2, \ldots, m$, where m is the number of boxes, (x'_i, y'_i) and (w'_i, h'_i) are the coordinates of left-top point and the size of B'_i. Here, the coordinate's origin is the top-left point in an image. The candidate text regions are generated by the following process:

1) Find the most left unprocessed box B'_l.
2) Obtain a set of boxes B based on $B_1 = B'_l$ by iteratively looking for the box $B_j = (x_j, y_j, w_j, h_j)$ which is the closest one to B_{j-1} and satisfies following conditions:

$$y_{j-1} - h_j < y_j < y_{j-1} + h_{j-1} \tag{4}$$

$$\frac{1}{2} < \frac{h_j}{h_{j-1}} < 2 \tag{5}$$

This step is based on an assumption that the texts nearly horizontally appear in scene images [7], [18], [25], [27], [31], which is ensured by (4). The height constraint, i.e., (5), is used to ensure that the heights of the adjacent boxes are not too different because the size of the characters in the same word should be similar.

3) Compute the distances between neighboring boxes B_j and $B_{j+1} \in B$ by

$$d_{j,j+1} = x_{j+1} - x_j - w_j, \quad \text{if } \#(B) \geq 2 \tag{6}$$

where $\#(B)$ denotes the number of boxes in B.
4) Generate the text regions according to the number of boxes in B as below. 1) When $\#(B) = 1$, directly set the corresponding CCR of B_1 as a CTR; 2) When $\#(B) = 2$, if $d_{1,2} > \frac{h_1 + h_2}{2}$, set the corresponding CCRs of B_1 and B_2 as two CTRs; otherwise consider them as a single CTR; 3) When $\#(B) > 2$, find the pairs of neighboring boxes B_j and B_{j+1} whose distances $d_{j,j+1}$ satisfies the following constraint

$$d_{j,j+1} > \frac{1.2}{\#(B) - 1} \times \sum_{k=1}^{\#(B)-1} d_{k,k+1} \tag{7}$$

and set them as points of separation in B to generate CTRs. For example, if there are 9 boxes in B and $d_{3,4}$, $d_{7,8}$ satisfy the condition, we split B between B_3 and B_4, and between B_7 and B_8 to generate three CTRs, i.e., the corresponding CCRs of $B_1B_2B_3$, $B_4B_5B_6B_7$, and B_8B_9.

5) Repeat step 1 to step 4 until all boxes are processed.

Applying this CTR generation process on all CCRs yields a large number of CTRs, as shown in Fig. 9(c).

The CTRs are then classified using the DLRC presented in Section III–B. Since the CTRs often contain different number of CCRs, their bounding boxes have different aspect ratios. Instead of directly reshaping them into a fixed size, we resize the CTRs' bounding rectangular regions to have a fixed height of 32 and a width that preserves the original aspect ratio, so as to reduce information loss. The resized regions are used as the inputs for deep feature extraction, and the original bounding rectangular regions are used for hand-crafted feature extraction. Fig. 9(d) shows the confidence map of the CTRs, in which the text regions receive much larger confidence scores (redder) than background regions. Finally, the text regions are detected by binarizing the confidence map with an adaptive threshold obtained by Otsu's algorithm, as shown in Fig. 9(e). Fig. 9(f) shows the final detected texts (indicated by the blue rectangle), which almost overlap with the ground truth (indicated by the red and the green rectangles). From Fig. 9(c) and (e), the non-text region "IIIIII" has been successfully removed after CTR classification. This is due to the fact that the positive samples (texts) used for CTR classification model training are constructed from frequently used English words. There exist no words like the CTR "IIIIII". Yet, the constructed negative samples (non-texts) contain some samples like "IIIIII", e.g., multiple adjacent blocks and fences. Therefore, the CTR classification can identify the CTR "IIIIII" as a non-text region.

V. Experimental Results and Analysis

A. Datasets and Evaluation Criteria

We test the proposed method on three benchmark datasets: the ICDAR 2011 robust reading competition dataset [65] (denoted as ICDAR2011), the ICDAR 2013 robust reading competition dataset [52] (denoted as ICDAR2013), and the street view text dataset [10] (denoted as SVT). The ICDAR2011 dataset contains 299 training images and 255 test images. The ICDAR2013 dataset contains 229 training images and 233 test images, which is a subset of ICDAR2011 created by removing the duplicated images and revising a small part of the ground truth annotations. Actually, in ICDAR 2013 [52] and 2015 [66], the Challenge of Text Localization and Text Segmentation in Robust Reading Competition use the same dataset. The SVT dataset contains 101 training images and 249 test images which are captured from Google street view and often have low resolution or quality. The background regions of these images have many patterns similar to those in text regions, and text regions are not fully annotated, which makes the SVT dataset greatly challenging. All datasets provide the bounding box ground truth for text detection task. Since the ICDAR2013 dataset also provides the pixel-level ground truth of text for the text segmentation task, we denote it as the ICDAR2013 segmentation dataset.

Due to the success of the synthetic word dataset [37], we used the same generation process to construct a larger number of synthetic scene text images, which includes font rendering, border/shadow rendering, base coloring, projective distortion, natural image blending, and noise. Further details are provided in [37]. We follow this process to construct a synthetic image training dataset, which contains approximately 500,000 (100,000) positive character (text) regions and 500,000 (100,000) negative character (text) regions.

We use the same evaluation criteria applied in the ICDAR2011 and ICDAR2013 competitions, i.e., precision, recall, and F-measure [52], [65]. The precision is the ratio of true positives to all detections, and the recall is the ratio of true positives to ground truths. The F-measure is the harmonic mean of the precision and recall.

The Caffe library [67] is used to implement the deep learning based framework, and all other parts of the proposed method are implemented using MATLAB.

B. Parameters Selection

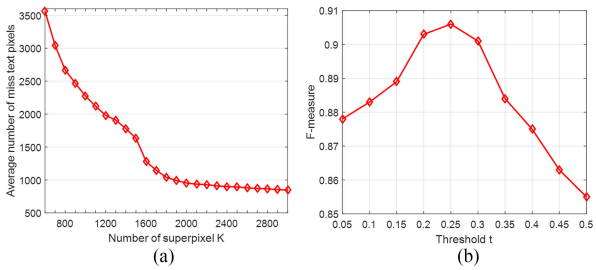
As described in Section III, there are several parameters, i.e., the number of superpixels K, the thresholds in background region removal, and the threshold t_C and t in region refinement. All of these parameters are selected by direct computation or exhaustive search on the ICDAR2013 training dataset, which contains the ground truth for pixel-level text segmentation. In this section, we describe the procedure applied to select the optimal values of the thresholds in background region removal and the threshold t_C, and investigate the sensitivity of the proposed method with different values of parameter K and t.

To obtain the final thresholds in the background region removal phase, we directly compute all possible values on all true text regions in the training dataset, and then select those which preserve all true text regions.

For threshold t_C, we directly compute the distances between the superpixel pairs (i.e., the superpixel corresponding to current text region contour pixel and the first superpixel belonging to background) of all tracking rays according to the ground truth of text segmentation in training dataset and select the value that maximizes tracking accuracy.

For the number of superpixels K, we perform an exhaustive search on the interval [600,3000], with a step size of 100. Given a value of K, the input image is first segmented into approximately K superpixels. Next, the overlapping area rate of each superpixel and the ground truth is calculated. In an ideal segmentation, the text regions would be over segmented, i.e all superpixels exclusively contain text or background. Here, the superpixels with overlapping area rates below 0.8 are considered as under segmented. Finally, the average number of pixels belonging to the ground truth in all under segmented superpixels (denoted as "missed text pixels") are computed for all training images. Fig. 10(a) shows the curve of the average number of "missed text pixels" plotted against the number of superpixels K. From Fig. 10(a), the average number of "missed text pixels" varies minimally for all values of $K \geq 2,000$. Hence, we set $K = 2000$, as additional of more superpixels requires additional time and computation power.

To obtain the optimal threshold t, we perform an exhaustive search on [0.05, 0.5] with step size 0.05. Given a value of t, the region merging process in region refinement and its



(a) (b)

Fig. 10. The evaluation results with different values of parameters, i.e., (a) the number of superpixel K and (b) the threshold t on ICDAR2013 training dataset.

TABLE I
THE PERFORMANCE OF DIFFERENT SCENE TEXT DETECTION APPROACHES ON ICDAR2011 DATASET

Approach	Year	Precision	Recall	F-measure
Our method	–	0.906	0.847	0.876
Tang *et al.* [50]	2017	0.902	0.859	0.880
Tian *et al.* [44]	2016	0.89	0.79	0.84
Gupta *et al.* [41]	2016	0.915	0.748	0.823
Qin *et al.* [38]	2016	0.876	0.772	0.821
He *et al.* [36]	2016	0.91	0.74	0.82
Jaderberg *et al.* [37]	2016	–	–	0.81
Tian *et al.* [34]	2015	0.862	0.762	0.809
Zhang *et al.* [31]	2015	0.84	0.76	0.80
Huang *et al.* [27]	2014	0.88	0.71	0.78
Yin *et al.* [30]	2014	0.863	0.683	0.762
Koo *et al.* [16]	2013	0.814	0.687	0.745
Yin *et al.* [39]	2015	0.838	0.660	0.738
Yao *et al.* [29]	2014	0.822	0.657	0.730
Huang *et al.* [18]	2013	0.82	0.75	0.73
Neumann *et al.* [20]	2013	0.793	0.664	0.723
Ren *et al.* [47]	2017	0.78	0.67	0.72

subsequent procedure of text detection are conducted on the training dataset. Next, the text detection performance is evaluated. Fig. 10(b) shows the curve of the F-measure plotted against the threshold t. Threshold $t = 0.25$ yields the best performance. The F-measure declines for $t < 0.25$ and $t > 0.25$. This is due to the fact that 1) some text regions belonging to the same text cannot be merged when $t < 0.25$, which yields results deemed incorrect by the word based evaluation. 2) Some text regions are merged with background when $t > 0.25$, which also may be considered as wrong detections. Therefore, we set $t = 0.25$.

C. Overall Performance

We fully evaluate the performance of the proposed method on three test datasets. Fig. 11 contains successful text detection results from the proposed methods, including challenging cases, e.g., single character, complex background, low contrast, etc.

The text detection performances of the proposed method and the state-of-the-art approaches are reported in Tables I–III in terms of precision, recall and F-measure. From these tables, the proposed method achieves promising results on all test datasets and outperforms the state-of-the-art approaches, except the one proposed by Tang *et al.* [50]. In [50], cascaded convolutional neural networks are designed for text detection and segmentation (denoted CCNN), and are adapted to all CCR extraction

TABLE II
THE PERFORMANCE OF DIFFERENT SCENE TEXT DETECTION APPROACHES ON ICDAR2013 DATASET

Approach	Year	Precision	Recall	F-measure
Our method	–	0.911	0.861	0.885
Tang *et al.* [50]	2017	0.919	0.871	0.895
Tian *et al.* [44]	2016	0.93	0.83	0.88
Sun *et al.* [49]	2014	0.870	0.857	0.864
Qin *et al.* [38]	2016	0.888	0.787	0.834
Zhu *et al.* [43]	2016	0.933	0.752	0.833
Zhang *et al.* [42]	2016	0.88	0.78	0.83
Gupta *et al.* [41]	2016	0.920	0.755	0.830
Cho *et al.* [40]	2016	0.863	0.785	0.822
He *et al.* [36]	2016	0.928	0.729	0.817
Tian *et al.* [34]	2015	0.852	0.759	0.803
Zhang *et al.* [31]	2015	0.88	0.74	0.80
Lu *et al.* [35]	2015	0.892	0.696	0.782
Neumann *et al.* [33]	2015	0.818	0.724	0.771
iwrr2014 [28]	2014	0.86	0.70	0.77
USTB TexStar [30]	2014	0.88	0.66	0.76
Text Spotter [14]	2012	0.88	0.65	0.74
Yin *et al.* [39]	2015	0.840	0.651	0.734
Ren *et al.* [47]	2017	0.81	0.67	0.73
Wu *et al.* [48]	2015	0.76	0.70	0.73

TABLE III
THE PERFORMANCE OF DIFFERENT SCENE TEXT DETECTION APPROACHES ON SVT DATASET

Approach	Year	Precision	Recall	F-measure
Our method	–	0.541	0.758	0.631
Tang *et al.* [50]	2017	0.588	0.762	0.664
Yi *et al.* [19]	2013	0.36	0.63	0.46
Xu *et al.* [26]	2014	0.34	0.60	0.43
Gupta *et al.* [41]	2016	0.299	0.407	0.245
Fraz *et al.* [32]	2015	–	0.516	–
Neumann *et al.* [14]	2012	0.191	0.329	0.242
Wang *et al.* [10]	2011	0.67	0.29	0.41

methods (e.g., using MSER [36], [38], [49] or CNN [42]) and types of features used to filter out non-text regions (e.g., hand-crafted [43] or CNN-based [36], [38], [41], [42], [44], [49]). For example, the F-measures are improved by 4.3% (from 0.84 [44] to 0.876), 0.6% (from 0.88 [44] to 0.885) and 37.2% (from 0.46 [19] to 0.631) on ICDAR2011, ICDAR2013 and SVT respectively. These high quantitative results demonstrate the effectiveness of the proposed method, which may be attributed to the fact that 1) the proposed SSFT is able to extract most of true character regions and 2) the proposed deep learning based region classification can effectively identify the character/text regions and background regions.

From Table I–III, the proposed method fails to outperform CCNN [50]. This is due to the fact that in the CCR extraction stage, CCNN [50] carefully designs and trains two CNN models with two large-scale datasets including 20,000 scene text images and 200,000 word images, which are constructed by considering different scenarios, while the proposed SSFT algorithm is hand-crafted and merely uses 229 images to tune the parameters. Therefore, CCNN [50] can deal with some challenging cases (e.g., low contrast, as shown in Fig. 12) which our method cannot effectively handle. However, CCNN [50] requires far

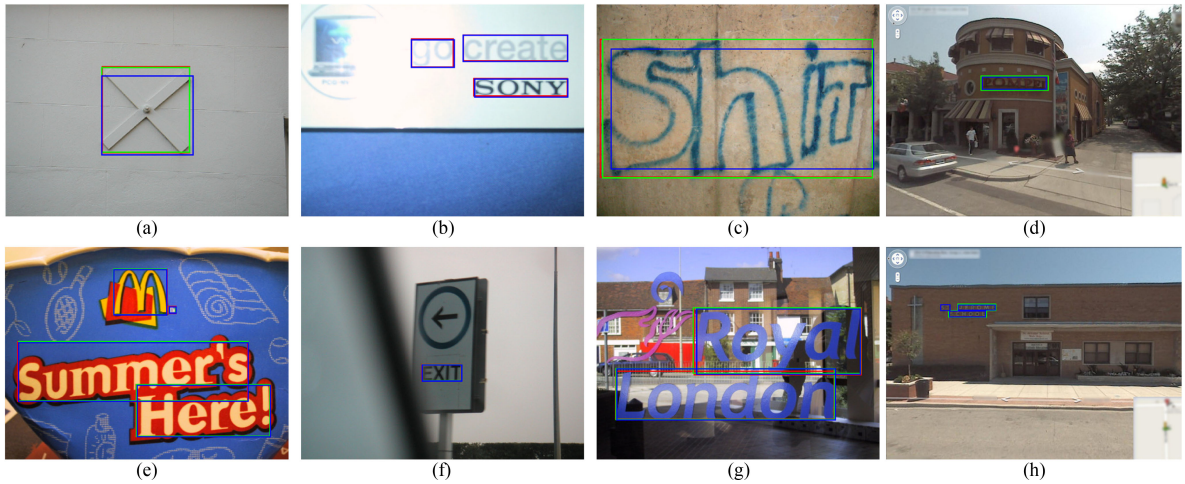


Fig. 11. Some results of the proposed method. The detected text is indicated by a blue rectangle and the ground truths of ICDAR2011 and ICDAR2013 (or SVT) are respectively indicated by a red rectangle and a green rectangle.

Fig. 12. Two examples of text detection results by the proposed method (blur rectangle) and CCNN [50] (pink rectangle).

TABLE IV
THE PERFORMANCE OF DIFFERENT APPROACHES ON ICDAR2013 SEGMENTATION DATASET IN PIXEL-LEVEL

Approach	Year	Precision	Recall	F-measure
Our method	–	0.9238	0.8312	0.8751
Tang *et al.* [50]	2017	0.9470	0.8560	0.8992
Lu *et al.* [35]	2015	0.8210	0.7727	0.7963
I2R_NUS_FAR [51]	2013	0.8170	0.7473	0.7806
I2R_NUS [51]	2013	0.7904	0.7357	0.7621
USTB_FuStar [51]	2013	0.7445	0.6958	0.7193
Text Detection [51]	2013	0.7620	0.6474	0.7001
NSTextractor [51]	2013	0.7628	0.6071	0.6761
NSTsegmentator [51]	2013	0.6395	0.6841	0.6610
OTCYMIST [51]	2013	0.5853	0.4611	0.5158

Fig. 13. Failure examples of the proposed method.

more computational resources to train the CNNs for candidate region extraction (approximately three days on a TESLA k40c GPU, vs. approximately 18 hours for the proposed method on an Intel i7-4790k CPU). In the region classification stage, the proposed method can greatly outperform CCNN [50] which will be demonstrated in next subsection.

Since the proposed method can obtain pixel-level text regions (see an example in Fig. 9(e)), we also evaluate its performance on theICDAR2013 segmentation dataset, which provides pixel-level text annotations. We directly compare the segmentation results of the proposed method with the ground truth and use the precision, recall, and F-measure as evaluation criteria. Table IV presents the performances of different approaches on the ICDAR2013 segmentation dataset. Our method greatly outperforms all mentioned approaches except CCNN [50], which demonstrates that our method can well extract the text regions and filter out the non-text regions at the pixel level. CCNN [50] outperforms the proposed method because CCNN [50] is a text detection and segmentation approach, in which a text segmentation CNN model is designed to segment the text much more precisely.

While our method performs well on all test datasets and can deal with various challenging scenarios, e.g., single character (as shown in Fig. 11(a)), blurred text (as shown in Fig. 11(b)), and complex background (as shown in Fig. 11(g)), it still fails in some cases. For example, when the text and background have very similar intensity or low contrast (as shown in Fig. 13(a)), or the scene images have low resolution or low quality (as shown in Fig. 13(b)), the proposed method cannot segment text from background after using the superpixel based clustering strategy. When the text elements which belong to different text lines are

TABLE V
THE PERFORMANCE OF TEXT DETECTION WITH DIFFERENT FEATURES AND FUSION STRATEGIES ON ICDAR2011\ICDAR2013\SVT DATASET

Features	Precision	Recall	F-measure
F1	0.858\0.862\0.484	0.769\0.778\0.702	0.811\0.818\0.573
F2	0.883\0.890\0.502	0.828\0.834\0.723	0.855\0.861\0.593
F3	0.837\0.843\0.471	0.748\0.754\0.687	0.790\0.796\0.559
Strategy2	0.890\0.895\0.514	0.837\0.842\0.736	0.863\0.868\0.605
Strategy3	0.898\0.904\0.529	0.841\0.849\0.749	0.869\0.876\0.620
Strategy1	0.906\0.911\0.541	0.847\0.861\0.758	0.876\0.885\0.631

connected (as shown in Fig. 13(c)), the proposed method cannot separate them effectively. When the scene images contain non-uniform illumination or strong reflective light on text regions (as shown in Fig. 13(d)), the proposed region classification algorithm cannot extract discriminative features to correctly classify these text regions. In the entire field of scene text detection, despite the development of a large number of successful methods in recent years, there still exist some challenging problems to be solved for practical applications. For example, incidental scene text in uncontrolled environments remains very challenging to classify. Another unsolved problem of interest in the construction of a model that can effectively discern unseen font styles and languages.

D. Performance of Different Features and Fusion Strategies

In the proposed method, two types of features are extracted for region classification: hand-crafted and CNN-based features. Three different feature fusion strategies are proposed. To evaluate their performances, we conduct experiments on the ICDAR2011 dataset with different feature settings and fusion strategies. Table V lists the experimental results in which the hand-crafted features are fused by using the deep network (denoted as initial fused (F1)) or direct concatenation (denoted as hand-crafted (F3)), and the deep features are denoted as CNN-based (F2). We can see that 1) each type of features achieves promising results, and F2 performs better than F1 and F3. This is due to the fact that CNN-based features are directly learned from images which can reflect the deep properties of the text and the background regions, while the hand-crafted features are extracted according to appearance characteristics of regions. 2) The initial fused features F1 perform much better than F3, which means the network Fusion1 can automatically learn the complex dependence of the hand-crafted features to enhance their strengths. 3) All feature fusion strategies improve the performance compared with the individual features, and feature fusion Strategy1 yields the best results. Indeed, the CNN-based features and the hand-crafted features are extracted according to different, yet complementary, types of information. Hence, effective fusion of CNN-based features and hand-crafted features by using the proposed deep learning framework (Strategy1) can greatly improve the performance.

As mentioned in Section V–C, SSFT is less effective than CCNN [50] in CCR extraction, but outperforms CCNN in region classification. This is due to the fact that CCNN exclusively extracts CNN-based features.

TABLE VI
THE PERFORMANCE OF SCENE TEXT DETECTION WITH DIFFERENT MODULE COMBINATIONS ON ICDAR2011 DATASET

Modules	Precision	Recall	F-measure
w/o (M2 & M5)	0.828	0.819	0.824
w/ M2	0.859	0.852	0.856
w/ (M2 & M5)	0.906	0.847	0.876

E. Contributions of Different Modules

The proposed method contains six modules: superpixel based clustering (M0), background region removal (M1), region refinement (M2), candidate character region classification (M3), candidate text region generation (M4) and candidate text region classification (M5) (cf. Fig. 2). M0, M1, M3 and M4 are the essential modules, while M2 and M5 are optional. To assess the contributions of these optional modules to text detection, we conduct experiments on the ICDAR2011 dataset with different module combinations: without (w/o) (M2 & M5), with (w/) M2, and with (w/) (M2 & M5). Experimental results of these strategies are reported in Table VI. Promising results are achieved w/o (M2 & M5). By adding module M2, all measures have been improved as the over segmented regions of text and background are merged, which improves the discriminability of extracted regional features and reduces the number of background regions. Next, with the addition of module M5, the proposed method attains the highest F-measure. Indeed, the precision significantly increases due to the usage of the information regarding relationships between characters, and the recall slightly drops because of the small number of text region training samples. We believe that a larger number of training samples can further improve the text detection performance.

F. Robustness Analyses of SSFT

As stated in the last paragraph of Section III–A, SSFT is more robust to noise than SFT. To validate this property, we compare the performance of both algorithms on images containing several levels of noise. We generate these noise-imbued images by adding salt and pepper noise at five density levels (0.02, 0.04, 0.06, 0.08, and 1) to images on the ICDAR2011 dataset. Performance is measured based on the F-measure metric. Fig. 14 compares SSFT's and SFT's F-measures for text detection with different levels of noise on the ICDAR2011 dataset. SSFT obtains higher F-scores than SST at all levels of noise and experiences a slower decline in performance as the noise level increases. Hence, it is a more powerful and noise-robust CCR extraction tool.

G. Time Performance

The proposed method (composed of M0∼M5 modules) is implemented in non-optimized MATLAB and C++ code, and is tested on a PC with a TESLA k40c GPU, an Intel i7 CPU and 32G RAM. The average runtimes per image in each module on the ICDAR2011 dataset are listed in Table VII.

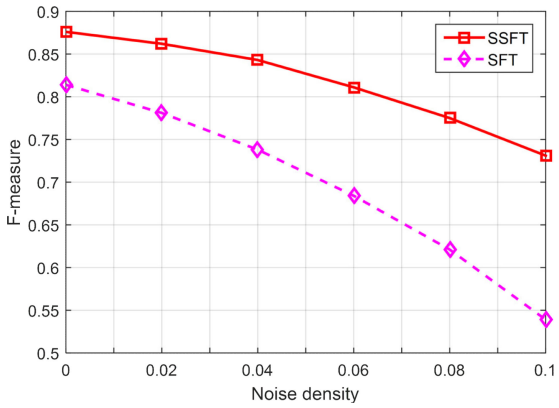


Fig. 14. The F-measures of text detection based on SSFT and SFT with different levels of noises on ICDAR2011 dataset.

TABLE VII
RUNTIME OF EACH MODULE (SECONDS)

M0	M1	M2	M3	M4	M5	Total
5.72	0.09	1.52	0.67	0.31	0.40	8.71

According to Table VII, the proposed method requires about 8.6 seconds to process a single image, well below the time required by other CNN based methods [31] (1 minute per image). From Table VII, the slowest module is M0 (5.72 s), which uses the SLIC algorithm for segmentation (5.6 s) and clustering (0.12 s). Recently, a GPU implementation of SLIC has been released, which increases speed by a factor of ∼83 times with a single GPU [68]. Hence, GPU optimization can greatly increase our method's speed. The second most time-consuming module is region refinement, i.e., M2 (1.52 s), in which the stroke width is obtained by tracking all regions' contour pixels. For different contour pixels, the tracking process is independent, and thus can be implemented in parallel to greatly accelerate M2.

The average runtime of SSFT (including M0, M1 and M2 modules) is about 7 s/image, well below that of SFT (∼24 s). Hence, our proposed SSFT outperforms SFT not only in accuracy, but also in speed.

VI. CONCLUSION

This paper proposes a novel method that involves a superpixel based stroke feature transform (SSFT) and deep learning based region classification (DLRC) for scene text detection. The SSFT effectively generates most of the true character regions and removes most of the background regions. Furthermore, SSFT can precisely obtain the character regions at the pixel level. The DLRC algorithm based on both hand-crafted and deep CNN features effectively discriminates the character and text regions from background regions. The hand-crafted and the deep CNN-based features, both of which have high discriminative power, are complementary, and can be fused via the proposed deep learning framework to greatly improve the performance. Experimental results confirm that the proposed method can effectively detect text in scenes of varying type and complexity.

ACKNOWLEDGMENT

The authors would like to thank I. Nogues, H. Pratt, and Y. Zheng for proofreading the manuscript, and NVIDIA Corporation for the GPU donation.

REFERENCES

[1] W. Tan, B. Yan, K. Li, and Q. Tian, "Image retargeting for preserving robust local feature: Application to mobile visual search," *IEEE Trans. Multimedia*, vol. 18, no. 1, pp. 128–137, Jan. 2016.
[2] L. Y. Duan, R. Ji, Z. Chen, T. Huang, and W. Gao, "Towards mobile document image retrieval for digital library," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 346–359, Feb. 2014.
[3] C. Xu, Y. F. Zhang, G. Zhu, Y. Rui, H. Lu, and Q. Huang, "Using webcast text for semantic event detection in broadcast sports video," *IEEE Trans. Multimedia*, vol. 10, no. 7, pp. 1342–1355, Nov. 2008.
[4] K. Kim, K. Jung, and J. Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1631–1639, Dec. 2003.
[5] X. Chen and A. Yuille, "Detecting and reading text in natural scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2004, pp. 366–373.
[6] K. Wang and S. Belongie, "Word spotting in the wild," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 591–604.
[7] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2963–2970.
[8] Y. Pan, X. Hou, and C. Liu, "A hybrid approach to detect and localize texts in natural scene images," *IEEE Trans. Image Process.*, vol. 20, no. 3, pp. 800–813, Mar. 2011.
[9] C. Yi and Y. Tian, "Text detection in natural scene images by stroke gabor words," in *Proc. Int. Conf. Document Anal. Recognit.*, 2011, pp. 177–181.
[10] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1457–1464.
[11] C. Yi and Y. Tian, "Text string detection from natural scenes by structure-based partition and grouping," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2594–2605, Sep. 2011.
[12] L. Neumann and J. Matas, "A method for text localization and recognition in real-world images," in *Proc. Asian Conf. Comput. Vis.*, 2010, pp. 770–783.
[13] A. Mishra, K. Alahari, and C. Jawahar, "Top-down and bottom-up cues for scene text recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2687–2694.
[14] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3538–3545.
[15] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1083–1090.
[16] H. Koo and D. Kim, "Scene text detection via connected component clustering and nontext filtering," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2296–2305, Jun. 2013.
[17] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "PhotoOCR: Reading text in uncontrolled conditions," in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 785–792.
[18] W. Huang, Z. Lin, J. Yang, and J. Wang, "Text localization in natural images using stroke feature transform and text covariance descriptors," in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 1241–1248.
[19] C. Yi and Y. Tian, "Text extraction from scene images by character appearance and structure modeling," *Comput. Vis. Image Understanding*, vol. 117, no. 2, pp. 182–194, Feb. 2013.
[20] L. Neumann and J. Matas, "Scene text localization and recognition with oriented stroke detection," in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 97–104.
[21] C. Shi, C. Wang, B. Xiao, Y. Zhang, and S. Gao, "Scene text detection using graph model built upon maximally stable extremal regions," *Pattern Recognit. Lett.*, vol. 34, no. 2, pp. 107–116, Jan. 2013.
[22] C. Yao, X. Bai, B. Shi, and W. Liu, "Strokelets: A learned multi-scale representation for scene text recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 4042–4049.
[23] Y. Tang and X. Wu, "Scene text detection via edge cue and multi-features," in *Proc. Int. Conf. Frontiers in Handwriting Recognit.*, 2016, pp. 156–161.

[24] Y. Li, W. Jia, C. Shen, and A. Hengel, "Characterness: An indicator of text in the wild," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1666–1677, Apr. 2014.
[25] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 512–528.
[26] H. Xu, L. Xue, and F. Su, "Scene text detection based on robust stroke width transform and deep belief network," in *Proc. Asian Conf. Comput. Vis.*, 2014, pp. 195–209.
[27] W. Huang, Y. Qiao, and X. Tang, "Robust scene text detection with convolution neural network induced MSER trees," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 497–511.
[28] A. Zamberletti, L. Noce, and I. Gallo, "Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions," in *Proc. Workshop of Asian Conf. Comput. Vis.*, 2014, pp. 91–105.
[29] C. Yao, X. Bai, and W. Liu, "A unified framework for multioriented text detection and recognition," *IEEE Trans. Image Process.*, vol. 23, no. 11, pp. 4737–4749, Nov. 2014.
[30] X. Yin, X. Yin, K. Huang, and H. Hao, "Robust text detection in natural scene images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 970–983, May 2014.
[31] Z. Zhang, W. Shen, C. Yao, and X. Bai, "Symmetry-based text line detection in natural scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2558–2567.
[32] M. Fraz, M. S. Sarfraz, and E. A. Edirisinghe, "Exploiting colour information for better scene text detection and recognition," *Int. J. Document Anal. Recognit.*, vol. 18, no. 2, pp. 153–167, Jun. 2015.
[33] L. Neumann and J. Matas, "Efficient scene text localization and recognition with local character refinement," in *Proc. Int. Conf. Document Anal. Recognit.*, 2015, pp. 746–750.
[34] S. Tian *et al.*, "Text flow: A unified text detection system in natural scene images," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 4651–4659.
[35] S. Lu, T. Chen, S. Tian, J.-H. Lim, and C.-L. Tan, "Scene text extraction based on edges and support vector regression," *Int. J. Document Anal. Recognit.*, vol. 18, no. 2, pp. 125–135, Jun. 2015.
[36] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-attentional convolutional neural networks for scene text detection," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2529–2541, Jun. 2016.
[37] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 1–20, Jan. 2016.
[38] S. Qin and R. Manduchi, "A fast and robust text spotter," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2016, pp. 1–8.
[39] X. Yin, W. Pei, J. Zhang, and H. Hao, "Multi-orientation scene text detection with adaptive clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1930–1937, Sep. 2015.
[40] H. Cho, M. Sung, and B. Jun, "Canny text detector: Fast and robust scene text localization algorithm," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3566–3573.
[41] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2315–2324.
[42] Z. Zhang *et al.*, "Multi-oriented text detection with fully convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4159–4167.
[43] S. Zhu and R. Zanibbi, "A text detection system for natural scenes with convolutional feature learning and cascaded classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 625–632.
[44] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 56–72.
[45] L. Neumann and J. Matas, "Real-time lexicon-free scene text localization and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1872–1885, Sep. 2016.
[46] X. Liu and W. Wang, "Robustly extracting captions in videos based on stroke-like edges and spatio-temporal analysis," *IEEE Trans. Multimedia*, vol. 14, no. 2, pp. 482–489, Apr. 2012.
[47] X. Ren *et al.*, "A convolutional neural network based Chinese text detection algorithm via text structure modeling," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 506–518, Mar. 2017.
[48] L. Wu, P. Shivakumara, T. Lu, and C. L. Tan, "A new technique for multi-oriented scene text line detection and tracking in video," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1137–1152, Aug. 2015.
[49] L. Sun, Q. Huo, W. Jia, and K. Chen, "Robust text detection in natural scene images by generalized color-enhanced contrasting extremal region and neural networks," in *Proc. Int. Conf. Pattern Recognit.*, 2014, pp. 2715–2720.
[50] Y. Tang and X. Wu, "Scene text detection and segmentation based on cascaded convolution neural networks," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1509–1520, Mar. 2017.
[51] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image Vis. Comput.*, vol. 22, no. 10, pp. 761–767, Sep. 2004.
[52] D. Karatzas *et al.*, "ICDAR 2013 robust reading competition," in *Proc. Int. Conf. Document Anal. Recognit.*, 2013, pp. 1484–1493.
[53] Q. Ye and D. Doermann, "Text detection and recognition in Imagery: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 7, pp. 1480–1500, Jul. 2015.
[54] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 391–405.
[55] Q. Zhang, L. Xu, and J. Jia, "100+ times faster weighted median filter (WMF)," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2830–2837.
[56] R. Achanta *et al.*, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.
[57] J. Kim, D. Han, Y. Tai, and J. Kim, "Salient region detection via high-dimensional color transform and local spatial support," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 9–23 Jan. 2016.
[58] T. Hastie *et al.*, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, vol. 2, 2nd ed. Berlin, Germany: Springer-Verlag, 2009.
[59] P. Dollar and C. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, Aug. 2015.
[60] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 886–893.
[61] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[63] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[64] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.
[65] A. Shahab, F. Shafait, and A. Dengel, "ICDAR 2011 robust reading competition challenge 2: Reading text in scene images," in *Proc. Int. Conf. Document Anal. Recognit.*, 2011, pp. 1491–1496.
[66] D. Karatzas *et al.*, "ICDAR 2015 competition on Robust Reading," in *Proc. Int. Conf. Document Anal. Recognit.*, 2015, pp. 1156–1160.
[67] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Conf. MultiMedia*, 2014, pp. 675–678.
[68] C. Y. Ren, V. A. Prisacariu, and I. D. Reid, "gSLICr: SLIC superpixels at over 250 Hz," arXiv:1509.04232, 2015.

Youbao Tang received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Harbin Institute of Technology, Harbin, China, in 2009, 2011, and 2016, respectively. He is currently a Postdoctoral Researcher with National Institutes of Health, Bethesda, Maryland, USA. His research interests include image processing, computer vision, biometrics, medical image analysis, etc.

Xiangqian Wu (M'06) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Harbin Institute of Technology (HIT), Harbin, China, in 1997, 1999, and 2004, respectively. He was a Lecturer (2004–2006), was an Associate Professor (2006–2009), and is a Professor (2009–present) with the School of Computer Science and Technology, HIT. He has authored or coauthored one book and more than 100 papers in international journals and conferences. His research interests include computer vision, pattern recognition, biometrics, medical image analysis, etc.