Text Localization in Natural Images using Stroke Feature Transform and Text Covariance Descriptors

Weilin Huang[§,‡,†], Zhe Lin[†], Jianchao Yang[†], and Jue Wang[†]

[†]Adobe Research
[§]Shenzhen Key Lab of Comp. Vis and Pat. Rec., Shenzhen Institutes of Advanced Technology, CAS, China
[‡]Department of Information Engineering, The Chinese University of Hong Kong
wl.huang@siat.ac.cn;{zlin,jiayang,juewang}@adobe.com

Abstract

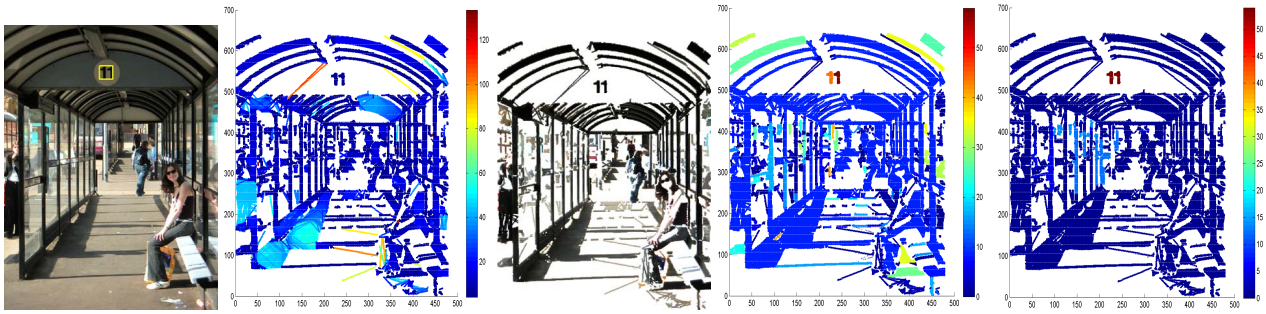
In this paper, we present a new approach for text localization in natural images, by discriminating text and non-text regions at three levels: pixel, component and text-line levels. Firstly, a powerful low-level filter called the Stroke Feature Transform (SFT) is proposed, which extends the widely-used Stroke Width Transform (SWT) by incorporating color cues of text pixels, leading to significantly enhanced performance on inter-component separation and intra-component connection. Secondly, based on the output of SFT, we apply two classifiers, a text component classifier and a text-line classifier, sequentially to extract text regions, eliminating the heuristic procedures that are commonly used in previous approaches. The two classifiers are built upon two novel Text Covariance Descriptors (TCDs) that encode both the heuristic properties and the statistical characteristics of text stokes. Finally, text regions are located by simply thresholding the text-line confident map. Our method was evaluated on two benchmark datasets: ICDAR 2005 and ICDAR 2011, and the corresponding F-measure values are 0.72 and 0.73, respectively, surpassing previous methods in accuracy by a large margin.

1. Introduction

Text detection and localization in natural images serves as a crucial component for content-based information retrieval, as textual information often provides important clues for understanding the high-level semantics of multimedia content. It has gained considerable attention in both academia and industry in the last decade. Despite the tremendous effort devoted to solving this problem, text localization remains to be challenging. The difficulties mainly lie in the diversity of text patterns and the complexity of scenes in natural images. For instance, texts in images often vary dramatically in font, size, and shape, and can be distorted easily by illumination or occlusion. Furthermore, text-like background objects, such as bricks, windows and leaves, often lead to many false alarms in text detection.

Previous text localization methods can be roughly divided into two categories: texture-based and component-based approaches. Texture-based methods scan the image at different scales using sliding windows, and classify text and non-text regions based on extracted window descriptors [10, 4, 9, 25]. Designing proper descriptors for specific visual objects plays a crucial role in these classification approaches. For instance, histogram-based local descriptors, such as Histogram of Oriented Gradients (HOG), have achieved great success for face detection [24] and pedestrian localization [5]. However, compared with faces and pedestrians, text-lines in natural images have a much larger layout variation (*e.g.* rotation, perspective distortion, aspect ratio, etc.) that cannot be well captured by generic descriptors. Besides, using sliding windows is computationally expensive. In general, the number of windows grows to N^2 for an image of N pixels, making it less practical [11].

Component-based methods first discard the majority of background pixels using low-level filters, and then construct component candidates from remaining pixels using a set of heuristic properties, *e.g.* consistency of stroke width [6, 26] and color homogeneity [27, 16, 3, 15]. Connected component analysis is further applied for filtering out outliers. One advantage of these methods is that they reduce the computational complexity substantially to $O(N)$. Another advantage is that the detected components provide a direct segmentation of the text letters, which benefits future applications such as recognition. However, previous component-based methods often suffer from three problems. Firstly, low-level operations are sensitive to image noise and distor-



(a) Input & result (b) Stroke width map (c) Stroke color map (d) Component conf. map (e) Text-line conf. map

Figure 1. The text localization pipeline of our method. The input image is shown in (a). We first apply the SFT filter on the input image to generate a stroke width map (b) and a stroke color map (c). We then apply the component classifier to generate a component confidence map (d), which is then fed into the text-line classifier to generate a text-line confidence map (e). The final detection result is generated by a simple thresholding on (e), which is overlaid on (a) as the yellow bounding box.

tions, leading to incorrect component grouping. Secondly, using heuristic rules to filter out outliers at the component and tex-line levels involve a set of manually tuned parameters, which may not generalize well to different datasets. Thirdly, heuristic filters are often not discriminative enough for distinguishing between true texts and text-like outliers, resulting in many false alarms (see Fig. 3).

Our goal is to develop a text localization system that combines the advantages of both classification-based and component-based methods, while overcoming their inherent limitations. The pipeline of our approach, dubbed as SFT-TCD, is shown in Fig. 1. It makes the following major contributions:

1. We propose a new low-level filter called *Stroke Feature Transform* (SFT), which extends the original Stroke Width Transform (SWT) [6] with color information. By using both color and edge orientation information (see Fig. 1(b,c)), the SFT filter effectively mitigates inter-component connections while enhancing intra-component connections, leading to a significantly better component candidate detection than SWT.

2. Building upon insights gained from heuristic properties and statistical characteristics of textual regions, we propose a component-level and a text-line-level Text Covariance Descriptor (TCD), which capture the inherent correlations between multiple features and effectively encode spatial information of text strokes.

3. Using the two TCDs, we build two classifiers instead of the commonly-used heuristic filtering methods for robust component and text-line classification, as shown in Fig. 1(d,e).

4. Our system achieved state-of-the-art results on two standard benchmarks and outperformed existing methods by a significant margin. We also experimentally validated that our method generalizes well to different datasets.

2. Related Work

Various heuristic properties of textual regions have been explored for pixel-level filtering. Based on the color uniformity of characters in a text string, Yi and Tian [27] proposed a color-based partition scheme, which applies weighted k-means clustering in the RGB space for separating text and background pixels. Neumann and Matas assumed each component as an independent extremal region (ER), and exploited maximally stable extremal regions (MSERs) to extract possible components in multiple channels [16, 15, 14]. Shivakumara *et al*. [20] first filtered the image in frequency domain by using Fourier-Laplacian transform, and then applied K-means clustering to identify candidate components.

A unique feature of a textual region is that the text strokes inside it usually have consistent, uniform width. Motivated by this observation, Epshtein *et al.* [6] proposed the SWT to detect stroke pixels by measuring the orientation difference between pairs of edge pixels, and grouping stroke pixels with similar widths as connected components. This method has been shown to be effective and is heavily used in many recent approaches. It also serves as the foundation for the SFT filter proposed in this work.

On the other hand, a number of histogram-statistics-based text descriptors have been proposed in texture-based methods. Chen and Yuille [4] adopted multiple features, including intensity means and standard deviations, histogram of intensities, gradients and gradient orientations, and edge information, as sub-window descriptors for AdaBoost classifiers. Hanif and Prevost [9] extracted three different features: mean difference feature (MDF), standard deviation (SD) and HOG, for constrained AdaBoost classifiers. In [25], a simple HOG descriptor was applied to Random

Ferns [1] classifiers for both text detection and recognition. Similarly, Pan *et al.* [17] computed text confidence values of sub-windows by using the HOG feature and a boosted cascade classifier: WaldBoost [21].

The most related work to ours is the recently proposed approach by Yao *et al.* [26]. They followed Epshtein *et al.*'s work [6] in pixel-level filtering and grouping based on the SWT filter. Then, after running heuristic filtering, two classifiers were trained and applied to remove the outliers in components and text-lines. Our method improves upon this approach by developing a novel SFT filter and two TCDs, which together lead to a significant performance boost. Furthermore, benefiting from the high discriminative power of the proposed TCDs, our system no longer contains heuristic filters for outlier rejection, thus is a more principled approach that generalizes better.

3. Proposed Approach

The proposed text localization system contains four main parts: (1) component[1] detection, (2) component filtering, (3) text-line construction and (4) text-line filtering. In this section, we discuss parts (1), (2) and (4) in details, and for completeness, we briefly describe part (3), which mainly follows previous work in [6, 26].

3.1. Component Detection using Stroke Feature Transform

Component detection involves two pixel-level operation steps. Firstly, we use a new stroke filtering method called Stroke Feature Transform (SFT) to identify text stroke pixels. Secondly, we use the two maps generated from SFT, a stroke width map and a stroke color map, to perform robust text pixel grouping (component generation).

3.1.1 The Stroke Feature Transform

The recently introduced Stroke Width Transform (SWT) [6] has been shown to be effective for text detection in the wild. It detects stroke pixels by shooting a pixel ray from an edge pixel ($\mathbf{p}_x$) to its opposite edge pixel ($\mathbf{p}_y$) along the gradient direction $\mathbf{d}_x$. The ray is considered as valid only if the gradient orientations of the pair of edge pixels are roughly opposite to each other. Otherwise, the ray is considered as invalid. All pixels covered by a valid ray are labeled by the same stroke width, which is the distance between the pair of edge pixels. In this way, SWT filters out the background pixels and assigns text pixels with stroke widths. However, only gradient orientation and edge information are used for ray tracking, and each ray is handled independently. In real cases, there often exist a large number of edge pixels that have irregular gradient orientations, which are not perpendicular to the correct stroke edge directions. As shown in the examples in Fig. 2, these irregular orientations would cause two major problems: (1) multiple letters can be accidentally merged into one component if the irregular orientations point to the outside of the stokes; and (2) a single character can be split into multiple components due to misrejection of ray candidates.

These problems are fundamental for SWT, and are critical for the overall system performance since the latter parts solely rely on the output of this step. To remedy these problems, we extend SWT by leveraging two additional cues during ray tracking: color uniformity and local relationships of edge pixels, and generate two maps, a stroke width map and a stroke color map jointly. We refer to this new filtering method as Stroke Feature Transform (SFT).

Assuming that the color inside a letter generally varies smoothly, the computation of SFT proceeds as follows. Firstly, Canny edge detector is applied to detect edge pixels from the input image. Secondly, for each edge pixel $\mathbf{p}_x$ on the canny edge map, we shoot a ray along its gradient direction $\mathbf{d}_x$ and check the pixels it encounters along the way. We end this ray at the current pixel $\mathbf{p}_{cur}$ and set it as a valid ray if $\mathbf{p}_{cur}$ satisfies either of the following two constraints:

1. *Stroke width constraint*: $\mathbf{p}_{cur}$ is an edge pixel and its gradient orientation $\mathbf{d}_{cur}$ is roughly opposite to $\mathbf{d}_x$ as: $\mid |\mathbf{d}_{cur} - \mathbf{d}_x| - \pi \mid < \frac{\pi}{2}$.

2. *Stroke color constraint*: the distance between the current pixel's color $\mathbf{p}_{cur}$ (denoted as C_{cur}) and the median ray color $C_{\bar{r}}$ (computed as median R, G, B of pixels on the ray) satisfies $\| C_{cur} - C_{\bar{r}} \| > \lambda_c$. λ_c is computed by a linearly decreasing function from 200 to 100 with respect to the number of pixels in the current ray. If this color discontinuity is detected, we reconsider the current pixel as an edge pixel and check it's orientation as in the Step 1 using a more strict threshold, $\mid |\mathbf{d}_{cur} - \mathbf{d}_x| - \pi \mid < \frac{\pi}{6}$.

If neither constraints is met for a certain number of checked pixels on the ray, we discard the current ray, and continue to the next edge pixel and repeat the above process. Once all the edge pixels are considered on the canny edge map, we further filter out invalid rays whose median colors are significantly different from its local neighbors on the canny edge map. This is called the *neighborhood coherency constraint*. Finally, we assign a stroke width value and the median RGB color value to all pixels in a valid ray to construct the stroke width map and the stroke color map.

SFT reduces the number of incorrect connections substantially compared to the original SWT approach. Due to the stroke color constraint, SFT is very effective at discarding rays shooting towards the outside of the strokes, because color often changes dramatically in the background region. Furthermore, it can help us recover some missing

[1] Component means text character in this paper.

(a) Input (b) Canny edge map (c) Stroke color map (d) Stroke width map (e) SWT stroke width map

Figure 2. Comparing SFT with SWT. The examples show that the proposed SFT method generates fewer inter-component (top example) and more intra-component connections (middle example), and is more robust against background structures (bottom example) than the original SWT method. They also show that SFT can recover more rays even in places where canny edge detection fails (see the two "0"s in the middel example). Overall SFT leads to more accurate component grouping than SWT, as shown as the red bounding boxes.

rays caused by missing edge information to prevent inter-component separations. The neighborhood coherency constraint is used to discard occasional errors in text strokes, as well as a large amount of incorrect, scattered connections in the background.

Another important advantage of SFT is that it produces a stroke color map as a byproduct of the transform, in which the stroke pixels have better uniformity and stronger distinction from background pixels than the stroke width map. Hence, by applying the stroke width map and the stroke color map jointly for grouping, our filter can effectively identify incorrectly-connected stroke components and other outliers in the background, as shown in Fig. 2.

3.1.2 Component Generation

We apply region growing [8] for grouping the stroke pixels into different components by using both the stroke width and color maps. The values in the stroke width map are normalized to [0 255]. Then region growing is performed in a 4-dimensional space by representing each stroke pixel using a width value and R, G, B color values. It simply connects neighboring pixels whose Euclidean distances in the defined 4-D space are below a threshold (empirically set as 75). To this end, we have successfully incorporated both stroke width and color information for low-level filtering and grouping, which outperforms the original SWT approach significantly on letter candidate detection, as shown in Fig. 2.

3.2. Text Covariance Descriptors for Filtering

As discussed earlier, many previous approaches use heuristic rules for filtering out false components, and grouping true text components into text-lines. To present a more principled system, we use classification-based methods to achieve these goals. Differing from previous methods that compute local features from all pixels within a sub-window, we propose two Text Covariance Descriptors (TCDs) by deriving features just from the detected stroke pixels, which naturally enable them to be highly discriminative representations of text information, and to have good generalization capability.

3.2.1 Region Covariance Descriptor

Region covariance descriptor was originally proposed by Tuzel *et al*. for object and human detection [22, 23]. It computes a covariance matrix of multiple element-level features within a defined region. Specifically, let $I \in \mathbb{R}^{H\times W}$ be an intensity image, and $F \in \mathbb{R}^{H\times W\times d}$ be its d-dimensional feature map, where each pixel of I is represented by d features as:

$$F(x,y) = \Phi(I(x,y)) \in \mathbb{R}^d, \tag{1}$$

where the function Φ can be any mapping or feature representation of the pixel element, such as spatial coordinates, intensity, color and gradients.

For a given region $U = \{\mathbf{u}_i\}_{i=1}^n \subset F$, and $\mathbf{u}_i \in \mathbb{R}^d$ is d-dimensional feature vector of the elements inside U, covariance descriptor of region U can be computed as:

$$C_U = \frac{1}{n-1}\sum_{i=i}^{n}(\mathbf{u}_i - \bar{\mathbf{u}})(\mathbf{u}_i - \bar{\mathbf{u}})^T, \tag{2}$$

where $C_U \in \mathbb{R}^{d\times d}$ and $\bar{\mathbf{u}}$ is the mean feature vector: $\bar{\mathbf{u}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{u}_i$.

3.2.2 TCD for Components (TCD-C)

The most straight-forward method for component filtering is to perform heuristic filtering with multiple features [6, 26, 27]. Histogram statistics of various low-level image properties have also been used for classification-based filters [10, 26, 4]. These methods typically compute the statistics of each single feature separately.

In order to explore statistical feature correlations, we propose to use the covariance matrix on a basic feature vector. The diagonal entries of the covariance matrix are the variances of each feature, while the nondiagonal entries capture the correlation between features, which are also informative for discriminating text strokes from background clutter. Furthermore, by jointly computing coordinates of stroke pixels with other features, the proposed TCD-C naturally encodes local spatial information into the descriptor.

The basic elements in each text component are the stroke pixels. Although the number of stroke pixels vary significantly among different components, an important merit of the covariance descriptor is that, it is invariant to the number of elements within the regions. The size of the descriptor is determined by the number of adopted features, which is often small. Based on heuristic and geometric characteristics of text strokes, we adopt nine different basic features for computing the TCD-C. The details of these features are listed and discussed bellow.

1. Normalized pixel coordinates I'_x, I'_y in X- and Y-axis for enhanced spatial locality. The original coordinates are normalized as: $I'_x = \frac{I_x - \widetilde{I}_x^{min}}{\widetilde{I}_x^{max} - \widetilde{I}_x^{min}}$, $I'_y = \frac{I_y - \widetilde{I}_y^{min}}{\widetilde{I}_y^{max} - \widetilde{I}_y^{min}}$, where $\widetilde{I}_x^{min}$, $\widetilde{I}_x^{max}$, $\widetilde{I}_y^{min}$ and $\widetilde{I}_y^{max}$ are the minimum and maximum coordinates of the regions in X- and Y-axis. Coordinate normalization enables the TCD-C to be invariant to geometric transforms and scale changes.

2. Pixel intensities I' and RGB values I'_R, I'_G, and I'_B in the stroke color map for color uniformity. All values are linearly normalized to $[0, 1]$.

3. Stroke width values in the stroke width map S_{swm} for stroke width consistency. The values are normalized by the maximum stroke width in the region.

4. Stroke distance values in a stroke distance map S_{dist}, normalized to $[0, 1]$, which compensate the stroke width map for stroke width consistency. The stroke distance map is computed from the stroke width map using the the Euclidean distance transform, the details are described in [3].

5. Per-pixel edge labeling for describing the stroke layout. The labels are 1 for edge pixels and 0 for non-edge ones.

Combining the above features, the resulting covariance descriptor is a 9×9 matrix. We concatenate the upper triangular elements of the matrix to construct a 45-dimensional vector as a component descriptor. Besides, three additional global features are added into the descriptor: (1) the aspect ratio (i.e. ratio between the height and width of the component); (2) the occupation percentage, computed as the ratio of total number of pixels to the number of stoke pixels in the component; and (3) the ratio of the component scale (*i.e*. the larger value of the width and height of the component region) to its mean stroke width map value. These three features are added to form the final TCD-C that has 48 dimensions. We train a random forests classifier [2] and use it to generate a confident score for each text component, as shown in Fig 3.

Given the detected components, the process of text-line aggregation is straightforward. We follow similar procedure as in [6]. Firstly, two components having similar characteristics are paired together by using a set of heuristic conditions, such as similar mean stroke width, color, height and distance between them. Secondly, pairs are merged together if they share the same components and have similar directions. The text-lines are detected when no pair or chain can be merged. Thirdly, but optionally, text-lines are broken into words by computing and thresholding the horizontal distances between consecutive components.

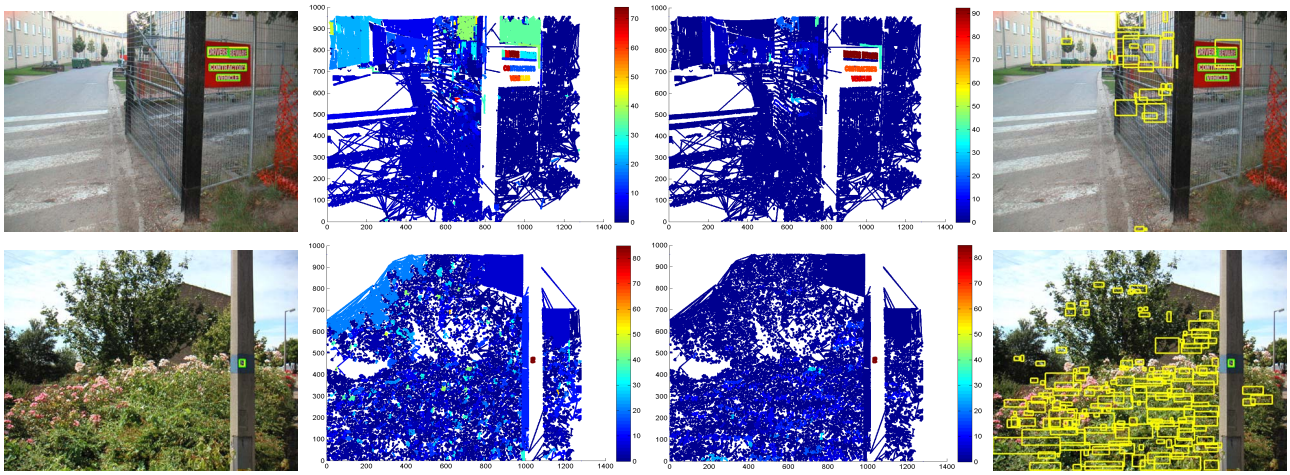
3.2.3 TCD for Text-lines (TCD-T)

Most text-like components have similar local structures to true text components, thus are difficult to be distinguished solely by component-level filtering. Here we present a text-line-level covariance descriptor, the TCD-T, to further identify these text-like outliers.

Region is defined in word or text-line level for the TCD-T filter, and elements are defined as the valid components within each region. Similar to TCD-C, heuristic properties, geometric characteristics and spatial distributions of the components are crucial information for generating high-level representations of text-lines. In addition, since each component is composed by a set of pixels, it can also provide meaningful statistical characteristics for the text-line.

In TCD-T we use two covariance matrices to compute two different types of the component features independently. The first matrix computes the correlated features between heuristic and geometric properties, as described bellow:

1. Seven heuristic features used in TCD-C, including mean values of intensities, colors, stroke widths and distances (mean[I', I'_R, I'_G, I'_B, S_{swm}, S_{dist}]). Here, S_{swm} and S_{dist} are normalized using their maximum values in the text-line. The last one is the occupation percentage of the stroke pixels in each component.



(a) Input & our result (b) Component conf. map (c) Text-line conf. map (d) Result w/o TCD-T

Figure 3. Illustrating the importance of the proposed text-line filter. Given the input image shown in (a), TCD-C generates a component confidence map in (b), which is fed into TCD-T to generate a text-line confidence map in (c). Our final text-line detection result is overlaid in (a), where green bounding boxes are ground truth and yellow ones are our detection results. If we discard TCD-T in the pipeline, the result is shown in (d), which contains many outliers.

2. The coordinates (C_x, C_y) of component centers, which are normalized with respect to the text-line's bounding box. Components within the same text-line should have similar or uniformly increasing C_y values for horizontal or slanted text-lines.

3. The heights of components normalized by the height of the text-line.

4. Cosine value of the angle between the current and the next components. The angle is measured by the orientation from the center of the current component to that of the next one. The value of last component is set as the same as the one before it. It is set to zero if only a single component is included in a text-line.

5. Horizontal distance from the current component to the next one, measured by the normalized horizontal coordinates (C_x) of two component centers. The distance of the last component is equal to the one before it, and is set to zero for single-component text-lines.

In total there are 12 component features adopted for the first covariance matrix, which in turn generates a 78-dimensional vector for text-line representation.

The second covariance matrix is computed to capture the correlation of statistical features among components. For each component, a 16-bin Histogram of Oriented Gradients (HOG) [5] is computed from its edge pixels, which carries the underlying shape information of its strokes. Therefore, a 16×16 covariance matrix is generated, resulting a 136-dimensional feature vector. We concatenate the feature vectors extracted from the two covariance matrices, along with two additional features: (1) the number of components in the text-line, normalized by dividing the maximum number of components in a textline, *e.g.* 10; and (2) the mean confident value of the components generated by the previous component-level classifier. The final TCD-T feature vector thus has 216 dimensions. Note that according to our feature design, TCD-T allows a single component to be treated as a word or text-line (only happens when using the text separation method in text-line aggregation). In this case, two covariance vectors are both $\mathbf{0}$ and the TCD-T vector only has non-zero entries in the last two dimensions.

Given the constructed TCD-T vectors, we train a discriminative text-line classifier using the random forests classifier [2] again. The text-line classifier generates a confidence value for each tex-line candidate, and the final text-line detection result is produced by simply thresholding this confidence map. In Fig. 3, we demonstrate using examples that TCD-T can effectively remove a large amount of text-like outliers in cluttered backgrounds.

4. Experiments and Results

4.1. Datasets

The proposed SFT-TCD method was evaluated on two public datasets, the ICDAR 2005 [13, 12] and ICDAR 2011 [18]. Both datasets have been widely used as the standard benchmarks for text detection in natural images. The ICDAR 2005 dataset includes 509 color images with image sizes varying from 307×93 to 1280×960. It contains 258 images in the training set and 251 images for testing. The ICDAR 2011 dataset contains 229 training images and 255 testing ones. Both datasets are evaluated in the word level, and have 1114 and 1189 words annotated in their test sets,

Table 1. Experimental results on the ICDAR 2005 dataset. (P for precision, R for recall and F for F-measure)

Method	Year	P	R	F
SFT-TCD	–	**0.81**	**0.74**	**0.72**
Yao *et al.* [26]	2012	0.69	0.66	0.67
Chen *et al.* [3]	2012	0.73	0.60	0.66
Epshtein *et al.* [6]	2010	0.73	0.60	0.66
Yi and Tian [28]	2013	0.71	0.62	0.63
Neumann and Matas [15]	2011	0.65	0.64	0.63
Zhang and Kasturi [29]	2010	0.73	0.62	–
Yi and Tian [27]	2011	0.71	0.62	0.62
Becker *et al.* [12]	2005	0.62	0.67	0.62
Minetto *et al.* [19]	2010	0.63	0.61	0.61
Chen and Yuille [4]	2004	0.60	0.60	0.58

respectively.

The performance of our method is quantitatively measured by *precision (P)*, *recall (R)* and *F-measure (F)*. They are computed using the same definitions in [13, 12] on image level. The final performance values are computed as the average values of all images in each dataset.

4.2. Experimental Setup

We use the training set of the ICDAR 2005 to collect positive and negative samples for training our component and text-line classifiers. The proposed SFT operator was first applied on the training images to obtain positive and negative component samples, classified by matching to the ground truth text regions. Then, components were paired and aggregated into words to form training data for the text-line classifier.

We apply the trained classifiers to both test sets of the ICDAR 2005 and ICDAR 2011 datasets. In other words, we intentionally do not re-train the classifiers on the ICDAR 2011 training data set to test the generalization power of the classifiers. In our experiments, we kept all components (with confidence values larger than zero) after running them through the component classifier, and discarded non-paired components and pairs with confidence values less than 0.1. The confidence value of a pair is the mean of the confidence values of its two components. Then the remaining pairs were used to construct the text-lines candidates. Finally, after applying the text-line classifier, we threshold the confidence map at 0.25 to generate the final detection results. We assume each test image includes at least one text-line, and select the pair or component of top confident value as the detected text if no text-line is detected in an image.

4.3. Results

The performance of the proposed approach on the two datasets is shown in Table 1 and 2. In ICDAR 2005 dataset, our method achieved the *precision*, *recall* and *F-measure* of 0.81, 0.74 and 0.72, respectively. All three values are higher than the closest reported results by large margins (0.08, 0.07 and 0.05), which demonstrates a significant improvement over existing methods, such as the SWT-based methods [26, 6] and MSER-based methods [3, 15]. Our method achieved a similar improvement on the ICDAR 2011 dataset, indicating that the two classifiers generalizes well. After carefully examining the results on each test images, we conclude that our significant performance improvement is mainly due to two factors: (1) the excellent performance of the proposed low-level SFT filter, which reliably detects letter candidates in most examples, leading to high *recall*, and (2) the effectiveness of the two-level TCDs, which lead to high *precision*.

Table 2. Experimental results on the ICDAR 2011 dataset. (P for precision, R for recall and F for F-measure)

Method	Year	P	R	F
SFT-TCD	–	**0.82**	**0.75**	**0.73**
Neumann and Matas [16]	2012	0.73	0.65	0.69
Yi and Tian [28]	2013	0.76	0.68	0.67
González *et al.* [7]	2012	0.73	0.56	0.63
Yi and Tian [27]	2011	0.67	0.58	0.62
Neumann and Matas [15]	2011	0.69	0.53	0.60

Figure 5. Failure cases.

Fig. 4 shows some successful results. They suggest that our system is robust against large variations in text font, color, size, and geometric distortion. In addition to detected text lines, our system also generates text pixel segmentation results shown at the bottom of each example, where white pixels include all pixels in the remaining valid text components. The segmentation can be potentially used in other applications such as text content or font recognition.

Fig. 5 shows two failure examples. Our system fails when the text strokes are too subtle and do not have strong edges (left), or the text region is partially occluded by other structures (right). In both cases the low level SFT filter failed to detect the right stroke pixels. More results can be found in the supplementary materials.

5. Conclusion

We have presented a novel system for detecting and localizing text-lines in natural images. Our key technical con-

P = 1.0000, R = 0.9536, F = 0.9762 | P = 0.9676, R = 0.9777, F = 0.9726 | P = 0.9772, R = 1.0000, F = 0.9885 | P = 0.9829, R = 0.9870, F = 0.9849 | P = 0.9567, R = 0.9381, F = 0.9473

Figure 4. Successful text detection results. Ground truth and our results are shown in green and yellow bounding boxes, respectively. Text pixel segmentation result and Precision, Recall and F values are given below each example.

tributions include a novel Stroke Feature Transform, a low-level filter that is extended from the original SWT approach by jointly considering stroke width and color information for robust filtering. We also propose two Text Covariance Descriptors for reliably filtering out text-like outliers at the component level and text-line level. Experimental results show that our approach has achieved the state-of-the-art results on publicly available datasets.

References

[1] A. Bosch, A. Zisserman, and X. Munoz. Image classication using random forests and ferns, 2007. ICCV. 3

[2] L. Breiman. Random forests. *Mach. Learning*, 45(1):5–32, 2001. 5, 6

[3] H. Chen, S. Tsai, G. Schronth, D. Chen, R. Grzeszczuk, and B. Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions, 2012. ICIP. 1, 5, 7

[4] X. Chen and A. Yuille. Detecting and reading text in natural scenes, 2004. CVPR. 1, 2, 5, 7

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection, 2005. CVPR. 1, 6

[6] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform, 2010. CVPR. 1, 2, 3, 5, 7

[7] A. González, L. Bergasa, J. Yebes, and S. Bronte. Text location in complex images, 2012. ICPR. 7

[8] R. Gonzalez and R. Woods. *Digital Image Processing 2nd Edition*. Prentice Hall, New Jersey, 2002. 4

[9] S. Hanif and L. Prevost. Text detection and localization in complex scene images using constrained adaboost algorithm, 2009. ICDAR. 1, 2

[10] 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. PAMI*, pages 1631–1639, 2003. 1, 5

[11] C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efcient subwindow search, 2008. CVPR. 1

[12] S. Lucas. Icdar 2005 text locating competition results, 2005. ICDAR. 6, 7

[13] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions, 2003. ICDAR. 6, 7

[14] L. Neumann and K. Matas. A method for text localization and recognition in real-world images, 2010. ACCV. 2

[15] L. Neumann and K. Matas. Text localization in real-world images using efciently pruned exhaustive search, 2011. ICDAR. 1, 2, 7

[16] L. Neumann and K. Matas. Real-time scene text localization and recognition, 2012. CVPR. 1, 2, 7

[17] Y.-F. Pan, X. Hou, and C.-L. Liu. Hybrid approach to detect and localize texts in natural scene images. *IEEE Trans. IP*, pages 800–813, 2011. 3

[18] A. Shahab, F. Shafait, and A. Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images, 2011. ICDAR. 6

[19] A. Shahab, F. Shafait, and A. Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images, 2011. ICDAR. 7

[20] P. Shivakumara, T. Phan, and C. Tan. A laplacian approach to multi-oriented text detection in video. *IEEE Trans. PAMI*, pages 412–419, 2011. 2

[21] J. Sochman and J. Matas. Waldboost learning for time constrained sequential detection, 2005. CVPR. 3

[22] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classication, 2006. ECCV. 4

[23] O. Tuzel, F. Porikli, and P. Meer. Human detection via classication on riemannian manifolds, 2007. CVPR. 4

[24] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, pages 137–157, 2004. 1

[25] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition, 2011. ICCV. 1, 2

[26] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images, 2012. CVPR. 1, 3, 5, 7

[27] C. Yi and Y. Tian. Text string detection from natural scenes by structure-based partition and grouping. *IEEE Trans. IP*, pages 2594–2605, 2011. 1, 2, 5, 7

[28] C. Yi and Y. Tian. Text extraction from scene images by character appearance and structure modeling. *CVIU*, pages 182–194, 2013. 7

[29] J. Zhang and R. Kasturi. Character energy and link energy-based text extraction in scene images, 2010. ACCV. 7