# Scene Text Detection Based on Robust Stroke Width Transform and Deep Belief Network

Hailiang Xu, Like Xue, Feng Su*

State Key Laboratory for Novel Software Technology,
Nanjing University,
Nanjing, 210023, China

**Abstract.** Text detection in natural scene images is an open and challenging problem due to the significant variations of the appearance of the text itself and its interaction with the context. In this paper, we present a novel text detection method combining two main ingredients: the robust extension of Stroke Width Transform (SWT) and the Deep Belief Network (DBN) based discrimination of text objects from other scene components. In the former, smoothness-based edge information is combined with gradient for generating high quality edge images, and various edge cues are exploited in Connected Component (CC) analysis on basis of SWT to eliminate inter-character and intra-character errors. In the latter, DBN is exploited for learning efficient representations discriminating character and non-character CCs, resulting in the improved detection accuracy. The proposed method is evaluated on ICDAR and SVT public datasets and achieves the state-of-the-art results, which reveal the effectiveness of the method.

## 1 Introduction

Text detection in natural scene images is an open problem which has been attracting significant attention as a critical part in many computer vision applications like image retrieval, scene analysis, and robotic navigation in urban environments. Meanwhile, several contests have been held in the past years [1–4], and the winning method in the recent ICDAR 2013 contest [4] was reported capable of detecting 66% words correctly from the testing images.

Numerous methods for text detection in natural scene images have been proposed [5–10] in recent years, which can be roughly divided into two groups: exhaustive search based methods and selective search based methods. The exhaustive search based methods [6, 11, 12] use a multiple-scale sliding window to extract features, such as the variance of intensity, HOG. However, the visual search space is huge, making an exhaustive search computationally expensive. Another group of text detection methods is based on the selective search [5, 7, 10, 13–17]. These methods first extract candidate text regions based on certain text properties, such as stroke width or approximately constant color, then adopt

* Corresponding author (suf@nju.edu.cn).

heuristic filtering rules or trained classifiers to identify text regions. These methods are attractive since they can simultaneously detect texts at any scale and are not limited to horizontal texts. Meanwhile, they usually work fast due to the low number of candidate text regions. However, the recall rates of both categories of present methods are still far from satisfaction for complicated images.

Specifically, as one popular method for text detection in natural images lately, the method of B. Epstein et al. in [5] shows the great success with the innovative and effective Stroke Width Transform (SWT) algorithm, which achieves the high precision and recall rates with very short processing time. However, the SWT algorithm is sensitive to the defections of the edge images, and its ability of discriminating text components from ambiguous non-text components, which usually result from complicated object compositions like varying colors or unfavourable lighting conditions like shadows or low contrast, could be further improved.

In this paper, we present a novel text detection method combining two main ingredients: the robust extension of SWT and the Deep Belief Network (DBN) [18] based discrimination of text objects from other scene components. For the former, we propose a hybrid method combining both the gradient-based and smoothness-based edge information for generating high quality edge images. Meanwhile, we exploit various edge cues in Connected Component (CC) analysis to eliminate inter-character and intra-character errors. For the latter, we propose a DBN-based seed character extraction to discriminate between character and non-character CCs, followed by growing seed characters to the final text lines.

We evaluate the proposed method on the most cited public ICDARs and Street View Text (SVT) datasets. The method achieves state-of-the-art results in the experiments on all datasets, which show the effectiveness of the method for detecting texts in natural images. We describe the proposed text detection method in detail in section 2 and present our experimental results in section 3.

## 2 Proposed Method

In this section, we introduce the details of our method, which consists of five processing stages: (1) edge detection, (2) component extraction, (3) component filtering, (4) seed character extraction and (5) text line formation, as shown in Fig. 1 along with the sample results of each stage.

### 2.1 Edge Detection Combining Gradient and Smoothness Cues

The SWT algorithm tightly relies on the edge detection result, and the Canny method used by the original SWT-based implementation [5] usually failed to acquire robust edges from some complicated image regions like those with low contrast and shadows. Therefore, the first stage of our method is devoted to generating high quality edge images, for which we propose a hybrid edge detection algorithm with two ingredients - the gradient-based and smoothness-based edge detection. Correspondingly, our method will generate two edge images ( as

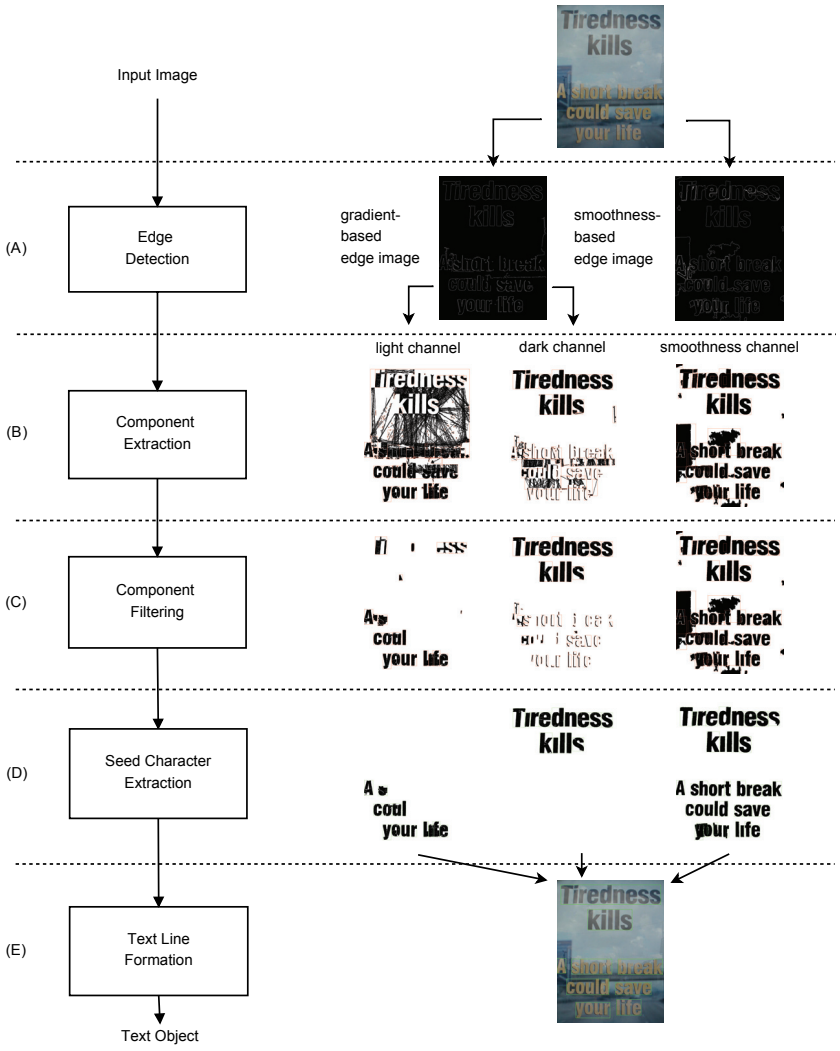


**Fig. 1.** The framework of the proposed approach (left) and illustration of the sample results of each processing stage (right).

shown in Fig. 1(A) ), the *gradient-based* edge image and the *smoothness-based* edge image, in which the value of one no-edge pixel is 0 and that of an edge pixel is 255.

For gradient-based edge detection, to extract more complete edge information from the image, the most direct way is to use a lower threshold in edge detection algorithm like Canny, which may, however, also lead to more erroneous edges. To overcome this, we propose to generate the gradient-based edges with a locally adaptive thresholding mechanism. We first generate a gradient image using Sobel operator, then initialize all pixels of an edge image to 0 and set one pixel in the edge image to 255 if its corresponding gradient value (norm of the gradient) in the gradient image is larger than 50, which is a relatively low initial threshold to allow discovering more edge pixels, and the no-maximal suppression(NMS) is used to eliminate invalid edge pixels. Next, to acquire more accurate edges from the initial detection results, we calculate a locally adaptive threshold $t$ for each edge pixel $p$ as Eq. (1) using a sliding window of the size $16 * 16$ with the step of 4.

$$t = \alpha * \frac{\sum_{i=1}^{n} \sum_{q \in S_i} G(q)}{\sum_{i=1}^{n} |S_i|} \tag{1}$$

where, $\alpha$ is empirically set to 0.9 in this paper, $n$ is the number of the sliding windows passing through the edge pixel $p$, $S_i$ is the set of edge pixels in the $i$th sliding window, $G(q)$ is the gradient value of an edge pixel $q$ in $S_i$, $|S_i|$ is the size of set $S_i$. Furthermore, since the edges in a text region are typically sparse, for a pixel with more than 3 sliding windows passing it, in which the percentage of edge pixels exceeds 30% (i.e. dense), we adjust its threshold $t$ to $max(T_{den}, t)$, $T_{den}$ is an empirical minimum threshold for dense edge regions, which is set to 120 in our method. Given the threshold $t$ for each edge pixel, one edge is extracted starting from an edge pixel $p$ with gradient value higher than $t$, then tracing from $p$ with the deep traverse algorithm, until an edge pixel with gradient value lower than $t/2$ is met.


(a) (b) (c) (d) (e) (f)

**Fig. 2.** Illustration of the gradient-based edge image (b)(e) and the smoothness-based edge image (c)(f) for the left sample image (a) with shadows and the right one (d) with low contrast. Note that the smoothness-based edge image better depicts the text contour than the gradient-based one.

Different from most edge extraction methods solely depending on the gradient information, we additionally introduce the smoothness-based edge image, based on the observation that the gradient-based edge image (as shown in Fig. 2(b)

2(e)) cannot depict text objects well with shadows (Fig. 2(a)) or low contrast (Fig. 2(d)). To compute the smoothness-based edge image, we first select a set of seed pixels, which has less than 5 neighbouring pixels in the $5 * 5$ window whose color distances (computed by CIE94 [1] in Lab color space) with seed pixel exceed 4. Then, we identify *seed regions* by starting growing from the seed pixel to its 8-neighbouring pixels if their color distance does not exceed 4. Next, taking all pixels in the seed regions as seed points and increasing the color distance threshold to 7, we again run the region growing algorithm and repeat it twice to acquire *color consistency regions (CCRs)*. CCRs of too big sizes will be eliminated for they are unlikely to be texts.

To further improve the completeness of CCRs, we apply two operations called smoothing and de-noising on CCRs. For the former, we traverse each pixel $p$ in the CCR, if one of its neighbouring pixels $q$ does not belong to any other CCRs and has a similar color (color distance less than 10) to $p$, then $q$ is added to the CCR. On the other hand, the de-noising operation fills the small cavity (less than 30 pixels) in a CCR. Fig. 3 illustrates the process of CCR extraction.

Based on the extracted CCRs, we compute the smoothness-based edge image by taking the contours of all CCRs as the edges, as shown in Fig. 2(c) 2(f).


**Fig. 3.** Illustration of the color consistency region (CCR) extraction process: (a) original image, (b) seed regions, (c) CCRs. (d) CCRs after smoothing operation, (e) CCRs after de-noising operation.

### 2.2 Component Extraction by Robust SWT

We propose a robust SWT-based component extraction method, which is performed independently on the gradient-based edge image and the smoothness-based edge image. The method consists of two aspects: the modification of SWT for more complete retrieval of matching edges while reducing noises, and the CC analysis exploiting edge information to avoid inter-character and intra-character errors in CC extraction.

**The Modified SWT** The SWT algorithm followed by CC analysis has been proved an effective and efficient way to detect texts in scene images [5, 8]. However, the original SWT and CC algorithm are usually insufficient for complicated occasions like those distorted and interfered texts.

[1] http://en.wikipedia.org/wiki/Color_difference.

We propose two modifications to the SWT algorithm. First, since we observe in many situations that the gradient intersect angle of 30 degree used in the original SWT algorithm is not large enough to generate good character strokes, we increase the angle to 60 degree and in order to reduce the number of false rays between edge pixels in SWT, we introduce a heuristic rule for the admissible gradient value of edge pixels that, if the gradient intersect angle exceeds 30 degree, the ratio of gradient values of the pair of edge pixels must fall in $[\frac{1}{1.5}, 1.5]$.

Second, to reduce the noises of false rays yielded by the SWT (like those in Fig. 4(a)), we propose to locate and remove the isolated rays from the SWT output. One ray is considered isolated if more than 60% of pixels on its path are isolated pixels, which in turn are defined as the pixels that have more than 3 neighboring pixels whose SWT value is $\infty$ (i.e. no passing ray). We remove all isolated rays in an iterative way since the operation may produce new isolated rays. The iteration stops if no more isolated rays are removed or the iteration number reaches 3. Similarly, since continuous isolated pixels are unlikely part of one character stroke, if there are 5 continuous isolated pixels in a ray, they are removed from the ray and this process is iterated until no more isolated pixels have been removed or the iteration number reaches 3. Fig. 4(b) shows the result of eliminating such noises from Fig. 4(a).


(a) (b)

**Fig. 4.** Results of the original SWT (a) and our modified SWT algorithm (b) on the sample image.

**Edge-Augmented Connected Component Analysis** The CC algorithm in [5] may produce two main types of errors - the intra-character and inter-character confusion, as illustrated in Fig. 5(a) 5(b). To eliminate most of such errors, we propose to exploit the edge information to promote the accuracy of CC extraction. First, we sort each pixel into one of four categories - $ES$, $E$, $S$ and $NONE$, as shown in Tab. 1, based on its corresponding value in the edge and SWT images.

Second, we propose a set of judgement conditions for the connectedness of two neighboring pixels in CC analysis and corresponding processing to avoid intra-character and inter-character errors.


(a) (b) (c) (d)

**Fig. 5.** Results of the original CC algorithm (left) on sample image with intra-character confusions (a) and sample image with inter-character confusions (b), and our edge-augmented CC algorithm (right) on the same sample images (c) and (d). Note that CCs colored in blue are false extraction results by the original algorithm.

**Table 1.** Categorization of pixels based on edge & SWT images.

| Category | Edge Image Value | SWT Image Value |
|---|---|---|
| $ES$ | 255 | Integer |
| $E$ | 255 | $\infty$ |
| $S$ | 0 | Integer |
| $NONE$ | 0 | $\infty$ |

1. Two $ES$ pixels $p$ and $q$ are considered connected if the intersect angle of their gradient directions (noted as $AG(p,q)$ in rest of paper) is less than $\pi/6$, and the ratio between their stroke widths (noted as $RSW(p,q)$) is less than 5, and the color distance between the rays (noted as $RCD(p,q)$) passing the $ES$ pixels does not exceed 10.
2. Two $S$ pixels $p$ and $q$ are considered connected if $RSW(p,q) < 5$.
3. One $ES$ pixel is considered connected with one $S$ pixel if they belong to the same ray.
4. If there is an $E$ pixel $q$ around an $ES$ pixel $p$ (Fig. 6), we adopt 8-neighbouring deep traverse algorithm to search another $ES$ pixel $r$ whose $AG(p,r) < \pi/3$ and $RSW(p,r) < 5$ and $RCD(p,r) < 10$ and the length of the traverse path does not exceed two times of the largest stroke width of pixel $p$ and $r$ and the $AG$ between each pair of two neighbouring $E$ pixels in the path does not exceed $\pi/6$, and we add all $E$ pixels to the component that pixel $p$ belongs to. Otherwise, we abandon all $E$ pixels in the path.

Fig. 5(c) 5(d) show the effectiveness of the edge-augmented CC algorithm, compared to the results by the original CC algorithm.

Note that, in order to detect both light text on dark background and vice-versa, we run the component extraction process twice on the gradient-based edge image, once using the normal gradient direction in SWT, producing the *light channel* of processing flow, and once using the inverse gradient direction in SWT, producing the *dark channel*. We also run the component extraction process on the smoothness-based edge image to produce the *smoothness channel*, as shown in Fig. 1(B).


**Fig. 6.** Illustration of the proposed edge-augmented CC algorithm. Left is the SWT image with the intra-character error as the input of the algorithm. Right is the edge image, in which pixels in pink color correspond to the traversing path of the algorithm connecting the $ES$ pixels $p$ and $r$ on the fragmented contour of the character.

### 2.3 Component Filtering

After component extraction processing, a large number of CCs will be extracted, from which we first exploit some characteristics of the CC to filter out non-character components:

- **Aspect ratio (AR)**.
- **Stroke width variation (SWV)**, defined by $\frac{s(c)}{u(c)}$, in which $u(c)$ and $s(c)$ are the mean and standard deviation, respectively, of the stroke widths in the component $c$.
- **Stroke count ratio (SCR)**, defined by the stroke count, which is the number of different rays within the component in the SWT process, divided by the component height.
- **Stroke width ratio (SWR)**, defined by the stroke width divided by the vertical height of the component.
- **SWT ratio (SWTR)**, defined by $\frac{|E_s|}{|E_{all}|}$, in which, $E_s$ is the set of $ES$ pixels of the component, and $E_{all}$ is the merged set of $ES$ and $E$ pixels. Typically, a character component should have a high SWT ratio.
- **Surround edge ratio (SER)**, defined by $\frac{|n_e|}{|n_{all}|}$, in which, $n_{all}$ is the set of pixels on the out-boundary contour of the component, and $n_e$ is the set of edge pixels in $n_{all}$.

We learn the valid ranges of these characteristics from the training character components in experiment, by choosing proper thresholds in the histograms of these characteristics. Tab. 2 shows the valid ranges learned on the ICDAR 2003 training dataset. Fig. 1 (C) shows the sample results of the component filtering.

### 2.4 Seed Character Extraction Based on DBN

Deep Belief Network (DBN) [18] is a neural network consists of many layers of Restricted Boltzmann Machines (RBMs). One RBM has a single layer of hidden

**Table 2.** CC characteristics for component filtering and seed character extraction and their valid ranges.

| Characteristics | Valid Ranges for Component Filtering | Valid Ranges for Seed Character Extraction |
|---|---|---|
| AR | [0.1,4] | $[\frac{1}{3},2]$ |
| SWV | [0, 1] | [0, 0.6] |
| SWR | [0.05, 0.75] | - |
| SCR | [0.8, $\infty$) | - |
| SWTR | [0.55, 1] | [0.75, 1] |
| SER | [0.65, 1] | [0.75, 1] |

units that are not connected to each other and have undirected, symmetrical connections to a layer of visible units. The RBM is trained by a learning algorithm called Contrastive Divergence (CD) [18], which uses the Gibbs sampling and the reconstruction error to train the weights of RBM. The role of a RBM is to model the distribution of its input. The DBN is constructed by stacking many RBMs on top of each other and linking the hidden layer of one RBM to the visible layer of the next RBM. One way to use the DBN for classification is to simply use the hidden layer activation of the top level RBM as features for any standard classifier or to add a topmost label layer, and train the whole model as a feedforward-backpropagate neural network.

In this work, we exploit DBN to detect seed characters from CCs. Since the appearance of characters in complicated images could be very different from that of normal characters, which may be caused by merged characters or fragmentary edges, it's necessary to screen out candidate CCs corresponding to characters from similar non-character ones. However, the discrimination plane between character and non-character components is highly non-linear and complicated, simple classifiers with manually crafted features may be insufficient, resulting in a relatively low accuracy of detection. On the other hand, DBN provides the capability of generatively discovering and learning effective features from data as activations on intermediate hidden layers, and thus is exploited in this work.

Note that CCs corresponding to characters may have very variant appearances, some of them may preserve the relatively complete characteristic of the character, while the others may have degraded and ambiguous appearances that are hard to be discriminated, as shown in Fig. 7. Therefore, given the filtered components from the previous processing stage, we first pick out candidate seed characters by some pre-filtering rules, and next classify them by DBN to acquire the final seed characters, on basis of which we then detect characters with less representative appearances by growing from the seed characters.

To efficiently pick out potential seed characters from CCs, we first learn the valid ranges and proper thresholds of the four characteristics of CCs (SWV, AR, SWTR and SER) from training samples of seed characters, then they are exploited to create a candidate seed character set $SC$ by abandoning other CCs not satisfying the thresholds. Tab. 2 shows the set of valid ranges of characteris-

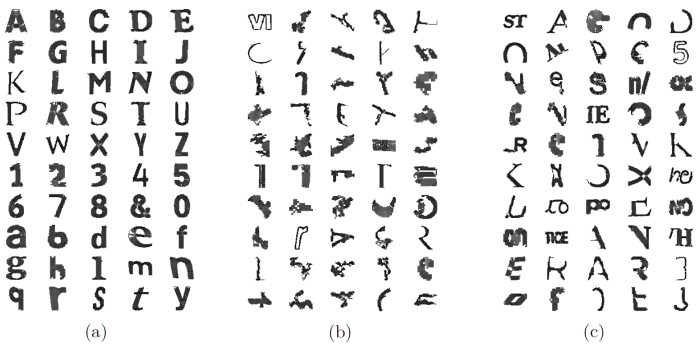

**Fig. 7.** Illustration of the training samples for seed character classification by DBN: (a) seed characters (positive samples), (b) non-character components (negative samples), (c) non-seed characters (also used as negative samples) for their vague appearance characteristics.

tics for seed characters, which is learned from the ICDAR 2003 training dataset consisting of 8821 non-seed characters and 7449 seed characters.

Next, DBN is exploited to classify final seed characters from $SC$. Each candidate seed character in $SC$ is first normalized to $48 * 48$ pixels, preserving the aspect of character, and is then used for the input layer of DBN. We add a top label layer of logistic regression to the DBN model for classification purpose and train the model with more than 100 configurations of parameter combinations. Tab. 3 shows the selected configuration yielding the best performance.

**Table 3.** Parameter settings of DBN in experiment.

| | |
|---|---|
| Number of hidden layers | 3 |
| Units per layer | 1400 600 100 |
| Unsupervised learning rate | 0.01 |
| Number of unsupervised epochs | 25 |
| Supervised learning rate | 2 |
| Number of supervised epochs | 50 |

Given the set of final seed characters $SC_f$ found by DBN, we grow from $SC_f$ to search for other degraded characters. Specifically, we search along the horizontal direction of each seed character for the non-seed characters, which have similar heights (difference ratio less than 2.0), stroke widths (difference ratio less than 2.0), gradient values (difference ratio less than 3.0), colors (color distance less than 8) and spatially close enough (distance less than 3 times of the width of the wider one) to the seed character. If found, the non-seed character is merged into $SC_f$. The process is repeated iteratively until no more such non-seed characters can be added to $SC_f$, then elements in $SC_f$ are taken as the character components, as shown in Fig. 1 (D).

### 2.5 Text Line Formation

The final stage of our method is to merge character components into text lines. Notice that we perform the previous processing stages independently in each of three channels: the dark channel, the light channel and the smoothness channel. Now, to combine characters detected in all three channels, we first merge the character components detected in the dark and the light channels into the so called *gradient-based* character component set $C_g$. Then, for each character component $c_i$ in the smoothness channel, we add it to $C_g$ if $c_i$ does not intersects with any character components in $C_g$.

Finally, to combine character components into text lines, when two character components in $C_g$ satisfy the same rules as used in the seed growing in Section 2.4, they are aggregated into a text line. Furthermore, two text lines can be merged together if they share one CC at one of their ends. Finally, we consider a character component not belonging to any text line as a special line containing only one character component, if it satisfies $SWV < 0.25$ and $SWTR > 0.9$ and $SER > 0.9$. Otherwise, the isolated character components are discarded.

Furthermore, to divide a text line into words, the word break algorithm is employed which estimates the typical word distance by computing the median value of character distances, and accordingly splits the text line at the inter-character gaps larger than the typical word distance. Fig. 1(E) shows the example of text detection results.

## 3 Experiments

We evaluated our method on several public datasets, including ICDAR 2003 Text Locating Competition dataset [1], Robust Reading Competition datasets of ICDAR 2011 [3] and ICDAR 2013 (Challenge 2: Reading Text in Scene Images) dataset [4], and Street View Text (SVT) [19] dataset.

The ICDAR 2003 dataset contains 509 fully annotated text images including 258 images for training and 251 images for testing. The ICDAR 2011 dataset consists of 484 text images with 229 training images and 255 testing images. The ICDAR 2013 dataset contains 562 images including 229 images for training and 233 images for testing. All the images are full-color and vary in size from $307 * 93$ to $3888 * 2592$ pixels. For processing convenience, they are resized so that the maximum of height and width is equal to 1600 pixels, considering the trade-off of runtime and detection accuracy. The method proposed in this paper is implemented in C language with the DBN implementation based on the DeepLearnToolbox library [20]. The experiments are performed on a PC platform with one Intel Core i3 CPU at 3.30 GHz.

We first evaluate our algorithms for edge detection, modified SWT and CC extraction on the ICDAR 2003 dataset, and compare them to those used in the original SWT work [5]. Tab. 4 shows the comparison results, along with the statistics of the outputs of the component filtering and the seed component selection processings. We can observe that the proposed method (with the hybrid

edge detection and the modified SWT and CC algorithms) achieves the best results with $f$ measure 0.70, although it is relatively slower than the original method. Comparatively, the original method [5] achieves the $f$ measure 0.66.

**Table 4.** Comparison of variant edge detection and component extraction methods. The 'Original SWT' and 'Original CC' algorithms are described in [5]. Filtered CCs are the CCs after the component filtering described in section 2.3. Seed CCs are the seed characters extracted by the proposed method in section 2.4. Indices $p$, $r$, $f$ are the metrics for text word detection used in ICDAR 2003 dataset.

| Edge Alg. | SWT & CC Alg. | # CCs | # Filtered CCs | # Seed CCs | $p$ | $r$ | $f$ | Time |
|---|---|---|---|---|---|---|---|---|
| Canny | Original | 68519 | 24966 | 4256 | 0.56 | 0.52 | 0.53 | 4s |
| Canny | Modified | 67925 | 26384 | 6109 | 0.68 | 0.63 | 0.64 | 4s |
| Gradient-based | Original | 190220 | 65436 | 6586 | 0.60 | 0.58 | 0.57 | 7s |
| Gradient-based | Modified | 169076 | 69133 | 11146 | 0.71 | 0.70 | 0.69 | 7s |
| Hybrid | Original | 212961 | 84146 | 12424 | 0.69 | 0.69 | 0.67 | 10s |
| Hybrid | Modified | 191817 | 88411 | 18614 | 0.70 | 0.74 | 0.70 | 10s |

Next, we evaluate and compare our text detection method with some other representative methods reported on the same ICDAR 2003, 2011 and 2013 datasets. We adopt the official evaluation protocol of ICDAR corresponding to the datasets. Notice that the evaluation protocol of ICDAR 2003 Text Locating Competition is different from that of Robust Reading Competition of ICDAR 2011 and ICDAR 2013. The $f$ measure calculated by ICDAR 2003 evaluation protocol is lower than the harmonic mean value $\frac{2*p*r}{p+r}$ by definition, where $p$ is the precision rate and $r$ is the recall rate, while the $f$ measures calculated by the ICDAR 2011 and ICDAR 2013 evaluation protocols are equal to the harmonic mean value.

Tab. 5 compares the performances of our method and some other methods evaluated on the ICDAR 2003 competition dataset. Among all methods, ours and that of Hinnerk Becker are evaluated with the official evaluation protocol of ICDAR 2003. The results show that our method performs overall much better than Hinnerk's. Since the other methods, however, used the different evaluation protocol from the official one, we focus on the precision and recall rates instead of the $f$ measure obtained by these methods in comparison. Our method achieves the significantly higher recall rate than other methods, owing to the effectiveness of the proposed hybrid edge detection and seed character extraction algorithms.

In Tab. 6 and Tab. 7, we compare the performances of our method with some other methods evaluated on the ICDAR 2011 and ICDAR 2013 competition datasets, respectively. Note that the evaluation protocol adopted is same for all methods being compared, in which the $f$ measure is equal to the harmonic mean value. Our method achieves again the significantly highest recall rate (74% and 75% respectively), with its $f$ measure slightly lower than the top one while better than all other methods.

**Table 5.** Comparison on ICDAR 2003 dataset

| Methods | $p$ | $r$ | $f$ |
|---|---|---|---|
| A. Mosleh [8] | **0.76** | 0.66 | **0.71** |
| **Our method** | 0.70 | **0.74** | 0.70 |
| X.B. Wang [9] | 0.71 | 0.70 | 0.70 |
| B. Epshtein [5] | 0.73 | 0.60 | 0.66 |
| H. Becker [2] | 0.62 | 0.67 | 0.62 |

**Table 6.** Comparison on ICDAR 2011 dataset

| Methods | $p$ | $r$ | $f$ |
|---|---|---|---|
| X.C. Yin [21] | **0.86** | 0.68 | **0.76** |
| **Our method** | 0.74 | **0.74** | 0.74 |
| L. Neumann [10] | 0.79 | 0.66 | 0.72 |
| Kim's [3] | 0.83 | 0.63 | 0.71 |
| X.B. Wang [9] | 0.73 | 0.67 | 0.70 |
| L. Neumann [7] | 0.73 | 0.65 | 0.69 |

**Table 7.** Comparison on ICDAR 2013 dataset

| Methods | $p$ | $r$ | $f$ |
|---|---|---|---|
| X.C. Yin [21] | **0.88** | 0.66 | **0.76** |
| **Our method** | 0.75 | **0.75** | 0.75 |
| Text Spotter[4] | 0.88 | 0.65 | 0.75 |
| CASIA NLPR[4] | 0.79 | 0.68 | 0.73 |
| Text Detector CASIA[4] | 0.85 | 0.63 | 0.72 |

Fig. 8 shows the texts detected by the proposed method from some sample images of the ICDAR datasets, which were interfered by complicated illumination, shadow, low contrast or color variation. The proposed text detection method exhibits sufficient robustness to these interferences.

As another popular public dataset for text detection evaluation, SVT dataset contains 647 words and 3796 letters in 249 images harvested from Google Street View. Our method achieves a recall rate of 60% and a precision rate of 34% using the ICDAR 2003 evaluation protocol. Fig. 9 shows the texts detected by the proposed method from some SVT sample images.

## 4 Conclusion

In this paper, we present a novel text detection method for natural scene images, which combines two main ingredients: the robust extension of the Stroke Width Transform algorithm and the Deep Belief Network based discrimination of text objects from other scene components. On the basis of SWT, we combine the gradient-based and smoothness-based edge information for generating high quality edge images, and further exploit various edge cues in connected component analysis to eliminate inter-character and intra-character errors. Given connected components extracted from the image, we exploit the DBN to discriminate character CCs from non-character ones and effectively aggregate multiple processing channels, yielding the overall improved detection performance in the experiments on ICDAR and SVT public datasets.

**Acknowledgement** Research supported by the National Science Foundation of China under Grant Nos. 61003113, 61272218 and 61321491.

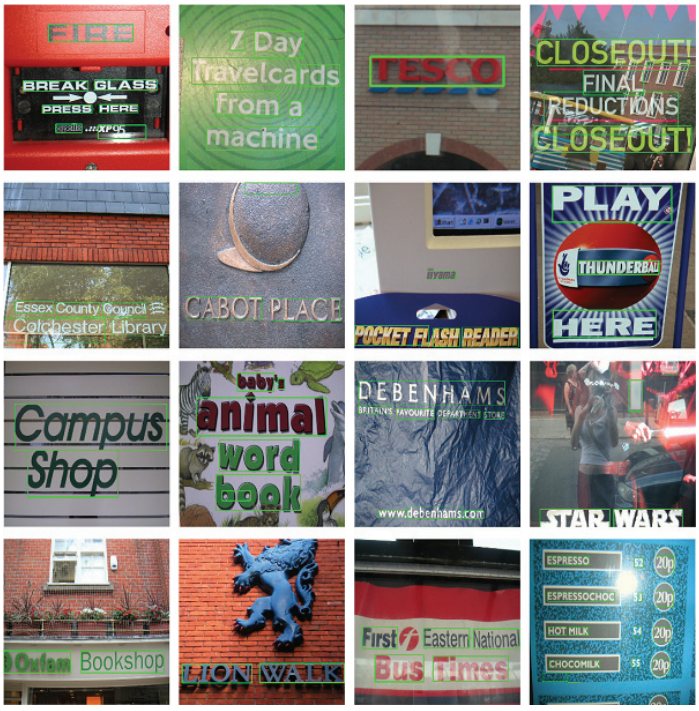


**Fig. 8.** Detected texts in some sample images from ICDAR 2003, 2011 and 2013 datasets. Only the cropped image regions containing the detected texts are shown.


**Fig. 9.** Detected texts in some sample images from SVT dataset. Only the cropped image regions containing the detected texts are shown.

## References


1. Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: ICDAR 2003 robust reading competitions. In: ICDAR. (2003) 682–687
2. Lucas, S.M.: ICDAR 2005 text locating competition results. In: ICDAR. (2005) 80–84
3. Shahab, A., Shafait, F., Dengel, A.: ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In: ICDAR. (2011) 1491–1496
4. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Bigorda, L.G., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., Heras, L.P.: ICDAR 2013 robust reading competition. In: ICDAR. (2013) 1484–1493
5. Epsthtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: CVPR. (2010) 2963–2970
6. Chen, X., Yuille, A.L.: Detecting and reading text in natural scenes. In: CVPR. (2004) 366–373
7. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: CVPR. (2012) 3538–3545
8. Mosleh, A., Bouguila, N.: Image text detection using a bandlet-based edge detector and stroke width transform. In: BMVC. (2012) 1–12
9. Wang, X.B., Song, Y.H., Zhang, Y.L.: Natural scene text detection with multi-channel connected component segmentation. In: ICDAR. (2013) 1375–1379
10. Neumann, L., Matas, J.: Scene text localization and recognition with oriented stroke detection. In: ICCV. (2013) 97–104
11. Mishra, A., Alahari, K., Jawahar, C.V.: Top-down and bottom-up cues for scene text recognition. In: CVPR. (2012) 2687–2694
12. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: ICCV. (2011) 1457–1464
13. Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: ACCV. (2010) 770–783
14. Yi, C., Tian, Y.: Text detection in natural scene images by stroke gabor words. In: ICDAR. (2011) 177–181
15. Koo, H.I., Kim, D.H.: Scene text detection via connected component clustering and nontext filtering. IEEE TIP **22** (2013) 2296–2305
16. Minetto, R., Thome, N., Cord, M., Stolfi, J., Precioso, F., Guyomard, J., Leite, N.: Text detection and recognition in urban scenes. In: ICCVW. (2011) 227–234
17. Zhang, J., Kasturi, R.: A novel text detection system based on character and link energies. IEEE Trans. Image Processing **23** (2014) 4187–4198
18. Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation **18** (2006) 1527–1554
19. Wang, K., Belongie, S.: Word spotting in the wild. In: ECCV. (2010) 591–604
20. Palm, R.B.: Prediction as a candidate for learning deep hierarchical models of data. Master's thesis, Technical University of Denmark (2012)
21. Yin, X., Yin, X., Huang, K., , Hao, H.: Robust text detection in natural scene images. IEEE Trans. PAMI **36** (2014) 970–983