Detecting Texts of Arbitrary Orientations in Natural Images

Cong Yao[1,2] Xiang Bai[1,2] Wenyu Liu[1] Yi Ma[2] Zhuowen Tu[2,3]
[1] Huazhong University of Science and Technology
[2] Microsoft Research Asia
[3] Lab of Neuro Imaging and Department of Computer Science, UCLA
yaocong2010@gmail.com, {xbai,liuwy}@hust.edu.cn, mayi@microsoft.com, ztu@loni.ucla.edu

Abstract

With the increasing popularity of practical vision systems and smart phones, text detection in natural scenes becomes a critical yet challenging task. Most existing methods have focused on detecting horizontal or near-horizontal texts. In this paper, we propose a system which detects texts of arbitrary orientations in natural images. Our algorithm is equipped with a two-level classification scheme and two sets of features specially designed for capturing both the intrinsic characteristics of texts. To better evaluate our algorithm and compare it with other competing algorithms, we generate a new dataset, which includes various texts in diverse real-world scenarios; we also propose a protocol for performance evaluation. Experiments on benchmark datasets and the proposed dataset demonstrate that our algorithm compares favorably with the state-of-the-art algorithms when handling horizontal texts and achieves significantly enhanced performance on texts of arbitrary orientations in complex natural scenes.

1. Introduction

The great success of smart phones and large demands in content-based image search/understanding have made text detection a crucial task in human computer interaction. It is desirable to build practical systems that are robust and fast enough to deal with natural scenes of various conditions; as shown in Fig. 1, we want to detect texts of large variations in language, font, color, scale and orientation in complex scenes. Although text detection has been studied extensively in the past [19, 15], the problem remains unsolved. The difficulties mainly come from two aspects: (1) the diversity of the texts and (2) the complexity of the backgrounds. On one hand, text is a high level concept but better defined than the generic objects [8]; on the other hand, repeated patterns (such as windows and barriers) and random clutters (such as grasses and leaves) may be similar to texts, and thus lead to potential false positives.

As our survey of related work shows below, most existing methods [16, 7, 22] have focused on detecting horizontal or near-horizontal texts. Detecting texts of arbitrary orientations in complex natural images has received much less attentions and remains a challenge for most practical systems. In this work, we make an effort to build an effective and practical detection system for texts of arbitrary orientations in complex natural scenes.

Figure 1. Detected texts in natural images.

When directly applied to detect texts of arbitrary orientations, conventional features (such as SWT used in [7]) that are primarily designed for horizontal texts would lead to significant false positives. In this paper, we introduce two additional sets of rotation invariant features for text detection. To further reduce false positives produced by these low-level features, we have also designed a two-level classification scheme that can effectively discriminate texts from non-texts. Hence, combining the strengths of specially designed features and discriminatively trained classifiers, our system is able to effectively detect texts of arbitrary orientations but produce fewer false positives.

To evaluate the effectiveness of our system, we have conducted extensive experiments on both conventional and new image datasets. Compared with the state-of-the-art text detection algorithms, our system performs competitively in the conventional setting of horizontal texts. We have also tested our system on a very challenging large dataset of 500 natural images containing texts of various orientations in complex backgrounds (see Fig. 8 (a)). On this dataset, our system works significantly better than any of the existing systems, with an F-measure about 0.6, more than twice that of the closest competitor.

2. Related Work

There have been a large number of methods dealing with text detection in natural images and videos [18, 16, 30, 27, 22, 26]. Comprehensive surveys can be found in [15, 19].

Existing approaches to text detection can be roughly divided into three categories: texture-based methods, region-based methods, and hybrid methods. Texture-based methods [16, 6, 10] treat texts as a special type of texture and make use of their properties, such as local intensities, filter responses and wavelet coefficients. These methods are computation demanding as all locations and scales are exhaustively scanned. Moreover, these algorithms mostly detect horizontal texts. Region-based methods [14, 7, 22] first extract candidate text regions through edge detection or clustering and then eliminate non-text regions using various heuristic rules. The third category, hybrid methods [23], is a mixture of texture-based and region-based methods.

Most existing algorithms, e.g. [23, 7], have focused on detecting horizontal texts. In this paper, we address the problem of detecting texts of large variations in natural images, which has great practical importance but has not been well studied. In [29, 26], methods that can detect text strings of arbitrary directions are proposed but they have a large set of rules and parameters; how general and applicable they are in dealing with scenes of large variation is unclear.

We observe two-sides aspects about the current text detection algorithms: (1) methods built on heavy learning (nearly black-box) [6] by training classifiers on a large amount of data reach certain but limited level of success (system [6] obtained from the authors produces reasonable results on horizontal English texts but has poor performances in the general cases); (2) systems based on smart features, such as Stroke Width Transform (SWT) [7], are robust to variations in texts but they involve many hand tunings and are still far from producing all satisfactory results, especially for non-horizontal texts.

In this paper, we adopt SWT and also design various features that are intrinsic to texts and robust to variations; a two-level classification scheme is devised to moderately utilize training to remove sensitive manual parameter tuning. We observe significant improvement over the existing approaches in dealing with real-world scenes.

Though widely used in the community, the ICDAR datasets [20, 24] only contain horizontal English texts. In [29], a dataset with texts of different directions is released, but it includes only 89 images without enough diversity in the texts and backgrounds. Here we collect a new dataset with 500 images of indoor and outdoor scenes. In addition, the evaluation methods used in [13] and the ICDAR competitions [21, 20] are designed for horizontal texts only. Hence, we use a different protocol that is suitable to handle texts of arbitrary orientations (see Sec. 4).

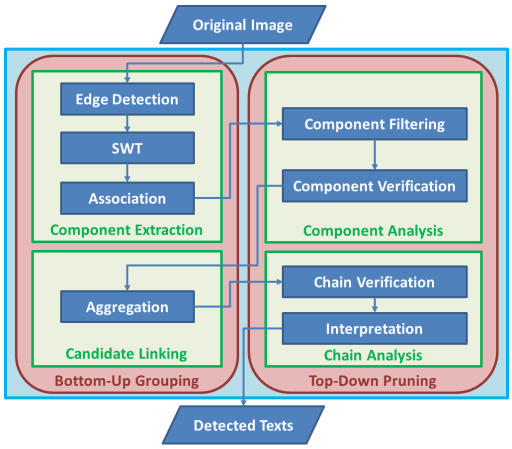

Figure 2. Pipeline of the proposed approach.

3. Methodology

In this section, we present the details of the proposed algorithm. Specifically, the pipeline of the algorithm will be presented in Sec. 3.1 and the details of the features will be described in Sec. 3.2.

3.1. Algorithm Pipeline

3.1.1 overview

The proposed algorithm consists of four stages: (1) component extraction, (2) component analysis, (3) candidate linking, and (4) chain analysis, which can be further categorized into two procedures, bottom-up grouping and top-down pruning, as shown in Fig. 2. In the bottom-up grouping procedure, pixels first form connected components and later these connected components are aggregated to form chains; in the top-down pruning procedure non-text components and chains are successively identified and eliminated.

Component extraction: At this stage, edge detection is performed on the original image and the edge map is input to SWT [7] module to produce an SWT image. Neighboring pixels in the SWT image are grouped together to form connected components using a simple association rule.

Component analysis: Many components extracted at the component extraction stage are not parts of texts. The component analysis stage therefore identifies and filters out those non-text components by a trained classifier.

Candidate linking: The remaining components are taken as character candidates[1]. The first step of the candidate linking stage is to link the character candidates into pairs. Two adjacent candidates are grouped into a pair if they have similar geometric properties and colors. The candidate pairs are then aggregated into chains in a recursive fashion.

Chain analysis: At the chain analysis stage, the chains determined at the former stage are verified by a chain level

[1]In fact, components do not necessarily correspond to characters, because a single character in some languages may consist of several strokes; however, we still call them characters (or character candidates) hereafter for simplicity.

Table 1. Basic component properties and their valid ranges.

Property	Definition	Range
width variation	$WV(c) = \frac{\sigma(c)}{\mu(c)}$	$[0, 1]$
aspect ratio	$AR(c) = \min\{\frac{w(c)}{h(c)}, \frac{h(c)}{w(c)}\}$	$[0.1, 1]$
occupation ratio	$OR(c) = \frac{q}{w(c) * h(c)}$	$[0.1, 1]$

classifier. The chains with low classification scores (probabilities) are discarded. The chains may be in any direction, so a candidate might belong to multiple chains; the interpretation step is aimed to dispel this ambiguity. The chains that pass this stage are the final detected texts.

3.1.2 Component Extraction

To extract connected components from the image, SWT [7] is adopted for its effectiveness and efficiency. In addition, it provides a way to discover connected components from edge map directly. We use Canny edge detector [5] to produce an edge map (Fig. 3 (b)) from the original image (Fig. 3 (a)). SWT is a local image operator which computes per pixel width of the most likely stroke containing the pixel. See [7] for details. The resulting SWT image is shown in Fig. 3 (c).

The next step of this stage is to group the pixels in the SWT image into connected components. The pixels are associated using a simple rule that the ratio of SWT values of neighboring pixels is less than 3.0. The connected components are shown in Fig. 3 (d). Note the red rectangles in the image. Each rectangle contains a connected component.

3.1.3 Component Analysis

The purpose of component analysis is to identify and eliminate the connected components that are unlikely parts of texts. Towards this end, we devise a two-layer filtering mechanism. The first layer is a filter consists of a set of heuristic rules. This filter runs on a collection of statistical and geometric properties of components, which are very fast to compute. For a connected component c with q foreground pixels (black pixels in the SWT image), we first compute its bounding box $bb(c)$ (its width and height are denoted by $w(c)$ and $h(c)$, respectively) and the mean as well as standard deviation of the stroke widths, $\mu(c)$ and $\sigma(c)$. The definitions of these basic properties and the corresponding valid ranges are summarized in Tab. 1.

The components with one or more invalid properties will be taken as non-text regions and discarded. This preliminary filter proves to be both effective and efficient. A large portion of obvious non-text regions are eliminated after this step. Notice the difference between Fig. 3 (d) and Fig. 3 (e).

The second layer is a classifier trained to identify and reject the non-text components that are hard to remove with the preliminary filter. A collection of component level features, which capture the differences of geometric and textural properties between text components and non-text components, are used to train this classifier. The criteria for feature design are: scale invariance, rotation invariance and low computational cost. To meet these criteria, we propose to estimate the center, characteristic scale and major orientation of each component (Fig. 4) before computing the component level features. Based on these characteristics, features that are both effective and computational efficient can be obtained. The details of these component level features are discussed in Sec. 3.2.1.

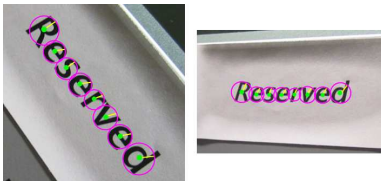

Figure 4. Component characteristics. The green points are the centers of the components. The radii of the pink circles represent their characteristic scales while the yellow lines indicate the major orientations. The two images, which contain the same text line, are taken from different viewpoints and distances.

For a component c, the barycenter $o(c)$, major axis $L(c)$, minor axis $l(c)$, and orientation $\theta(c)$ are estimated using Camshift algorithm [3] by taking the SWT image of component c as distribution map. The center, characteristic scale and major orientation of component c are defined as: $O(c) = o(c)$, $S(c) = L(c) + l(c)$, and $\Theta(c) = \theta(c)$.

These characteristics are invariant to translation, scale and rotation to some degree (Fig. 4). As we will explain in Sec. 3.2.1, this is the key to the scale and rotation invariance of the component level features.

We train a component level classifier using the component level features. Random Forest [4] is chosen as the strong classifier. The component level classifier is the first level of the two-level classification scheme. The probability of component c, $p_1(c)$, is the fraction of votes for the positive class (text) from the trees. The components whose probabilities are lower than a threshold T_1 are eliminated and the remaining components are considered as character candidates (Fig. 3 (f)).

3.1.4 Candidate Linking

The character candidates are aggregated into chains at this stage. This stage also serves as a filtering step because the candidate characters cannot be linked into chains are taken as components casually formed by noises or background clutters, and thus are discarded.

Firstly, character candidates are linked into pairs. In [7], whether two candidates can be linked into a pair is determined based on the heights and widths of their bounding boxes. However, bounding boxes of candidates are not rotation invariant, so we use their characteristic scales instead. If two candidates have similar stroke widths (ratio between the mean stroke widths is less than 2.0), similar sizes (ratio between their characteristic scales does not exceed 2.5), similar colors and are close enough (distance between them

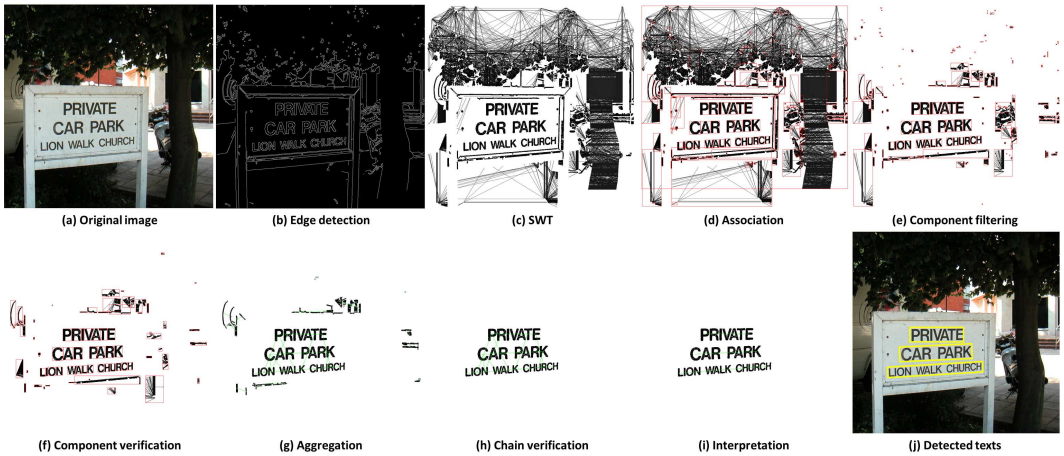

Figure 3. Text detection process. See text for details.

is less than two times the sum of their characteristic scales), they are labeled as a pair. Unlike [7], which only considers horizontal linkings, the proposed algorithm allows linkings of arbitrary directions. This endows the system with the ability of detecting texts of arbitrary orientations, not limited to horizontal texts (see Fig. 1). Note that a character candidate may belong to several pairs.

Next, a greedy hierarchical agglomerative clustering [12] method is applied to aggregate the pairs into candidate chains. Initially, each pair constitutes a chain. Then the similarity between each couple of chains that share at least one common candidate and have similar orientations is computed; chains with the highest similarity are merged together to form a new chain. The orientation consistency $s_o(C_1, C_2)$ and population consistency $s_p(C_1, C_2)$ between two chains C_1 and C_2, which share at least one common candidate, are defined as:

$$s_o(C_1, C_2) = \begin{cases} 1 - \dfrac{\gamma(C_1, C_2)}{\pi/2} & \text{if } \gamma(C_1, C_2) \leq \dfrac{\pi}{8} \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

and

$$s_p(C_1, C_2) = \begin{cases} 1 - \dfrac{|n_{C_1} - n_{C_2}|}{|n_{C_1} + n_{C_2}|} & \text{if } \gamma(C_1, C_2) \leq \dfrac{\pi}{8} \\ 0 & \text{otherwise} \end{cases}, \quad (2)$$

where $\gamma(C_1, C_2)$ is the included angle of C_1 and C_2 while n_{C_1} and n_{C_2} are the candidate numbers of C_1 and C_2. The similarity between two chains C_1 and C_2 is:

$$s(C_1, C_2) = \omega \cdot s_o(C_1, C_2) + (1 - \omega) \cdot s_p(C_1, C_2), \quad (3)$$

where $\omega \in [0, 1]$ is a control parameter. ω is set to 0.5 to give equal weights to $s_o(C_1, C_2)$ and $s_p(C_1, C_2)$. According to this similarity definition, the chains with proximal sizes and orientations are merged with priority. This merging process proceeds until no chains can be merged.

At last, the character candidates not belonging to any chain are discarded. The candidate chains after aggregation are shown in Fig. 3 (g). Each green line represents a candidate chain.

3.1.5 Chain Analysis

The candidate chains formed at the previous stage might include false positives that are random combinations of scattered background clutters (such as leaves and grasses) and repeated patterns (such as bricks and windows). To eliminate these false positives, a chain level classifier is trained using the chain level features (Sec. 3.2.2). Random Forest [4] is again used. The chain level classifier is the second level of the two-level classification scheme. The probability of chain C, $p_2(C)$, is the fraction of votes for the positive class (text) from the trees. The chains with probabilities lower than a threshold T_2 are eliminated.

To make better decisions, the total probability of each chain is also calculated. For a chain C with n candidates $c_i, i = 1, 2, \cdots, n$, the total probability is defined as: $p(C) = (\dfrac{\sum_{i=1}^{n} p_1(c_i)}{n} + p_2(C))/2$. The chains whose total probabilities are lower than a threshold T are discarded.

As texts of arbitrary orientations are considered, the remaining chains may be in any direction. Therefore, a candidate might belong to multiple chains. For example, in Fig. 3 (h) the character 'P' in the first line is linked in three chains (note the green lines). In reality, however, a character is unlikely to belong to multiple text lines. If several chains compete for the same candidate, only the chain with the highest total probability will survive (note the difference between Fig. 3 (h) and Fig. 3 (i)).

The survived chains are outputted by the system as detected texts (Fig. 3 (j)). For each detected text, its orientation is calculated through linear least squares [12] using the centers of the characters; its minimum area rectangle [9] is estimated using the orientation and the bounding boxes of

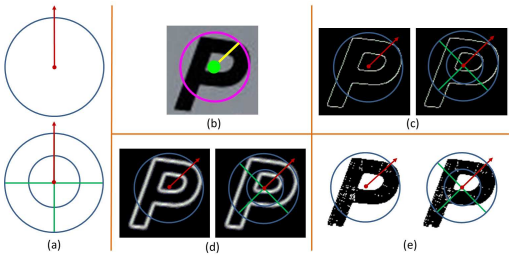

Figure 5. Templates and calculation of scalable rotative descriptors. (a) Two templates used for computing the descriptors. The radius space and angle space are partitioned evenly in a coarse-to-fine manner. The red arrows indicate the reference orientations of the templates. (b) Component and its characteristics. (c)(d)(e) Calculation of contour shape, edge shape and occupation ratio. See text for details.

the characters. Word partition, which divides text lines into separate words, is also implemented in the proposed algorithm; but it is not shown in Fig. 3 since the general task of text detection does not require this step.

The whole algorithm is performed twice to handle both bright text on dark background and dark text on bright background, once along the gradient direction and once along the inverse direction. The results of two passes are fused to make final decisions. For clarity, only the results of one pass are presented in Fig. 3.

3.2. Feature Design

We design two collections of features, component level features and chain level features, for classifying text and non-text, based on the observation that it is the median degree of regularities of text rather than particular color or shape that distinguish it from non-text, which usually has either low degree (random clutters) or high degree (repeated patterns) of regularities. At character level, the regularities of text come from nearly constant width and texturelessness of strokes, and piecewise smoothness of stroke boundaries; at line level, the regularities of text are similar colors, sizes, orientations and structures of characters, and nearly constant spacing between consecutive characters.

3.2.1 Component Level Features

Inspired by Shape Context [1] and Feature Context [28], we devise two templates (Fig. 5 (a)) to capture the regularities of each component in coarse and fine granularity, respectively. The radius and orientation of the templates are not stationary, but adaptive to the component. When computing descriptors for a component, each template is placed at the center and rotated to align with the major orientation of the component; the radius is set to the characteristic scale of the component. Different cues from the sectors are encoded and concatenated into histograms. In this paper, the following cues are considered for each sector:

– **Contour shape [11].** Contour shape is a histogram of oriented gradients. The gradients are computed on the component contour (Fig. 5 (c)).

– **Edge shape [11].** Edge shape is also a histogram of oriented gradients; but the gradients are computed at all the pixels in the sector (Fig. 5 (d)).

– **Occupation ratio.** Occupation ratio is defined as the ratio between the number of the foreground pixels of the component within the sector and the sector area (Fig. 5 (e)).

To achieve rotation invariance, the gradient orientations are rotated by an angle $\Theta(c)$, before computing contour shape and edge shape. Then, the gradient orientations are normalized to the range $[0, \pi]$. 6 orientation bins are used for computing histograms of contour shape and edge shape, to cope with different fonts and local deformations. For each cue, the signals computed in all the sectors of all the templates are concatenated to form a descriptor. We call these descriptors scalable rotative descriptors, because they are computed based on templates that are scalable and rotative. Scalable rotative descriptors are similar to PHOG [2], as they both adopt spatial pyramid representation [17].

Different from the templates used for computing PHOG, our templates are circular and their scale and orientation are adaptive to the component being described. This is the key to the scale and rotation invariance of these descriptors. We found through experiments (not shown in this paper) that using finer templates can slightly improve the performance, but will largely increase the computational burden.

Another three types of features are also considered:

– **Axial ratio.** Axial ratio is computed by dividing the major axis of the component c with its minor axis: $XR(c) = L(c)/l(c)$.

– **Width variation.** This feature is the same as defined in Tab. 1.

– **Density.** The density of component c is defined as the ratio between its pixel number q and characteristic area (here the characteristic area is $\pi \cdot S^2(c)$, not the area of the bounding box): $D(c) = q/(\pi \cdot S^2(c))$.

3.2.2 Chain Level Features

Eleven types of chain level features are designed to discriminate text lines from false positives (mostly repeated patterns and random clutters) that cannot be distinguished by the component level features.

For a candidate chain C with n ($n \geq 2$) candidates $c_i, i = 1, 2, \ldots, n$, the features are defined as below:

– **Candidate count.** This feature is adopted based on the observation that false positives usually have very few (random clutters) or too many (repeated patterns) candidates.

– **Average probability.** The probabilities given by the component level classifier are reliable. This feature is the average of all the probabilities ($p_1(c_i), i = 1, 2, \ldots, n$) of the candidates belonging to C.

– **Average turning angle.** Most texts present in linear form, so for a text line the mean of the turning angles at the interior characters ($\tau(c_i), i = 2, 3, \ldots, n-1$) is very small;

however, for random clutters this property will not hold. $\tau(c_i)$ is the included angle between the line $\overline{O(c_{i-1})O(c_i)}$ and $\overline{O(c_i)O(c_{i+1})}$.

– **Size variation.** In most cases characters in a text line have approximately equal sizes; but it's not that case for random clutters. The size of each component is measured by its characteristic scale $S(c_i)$.

– **Distance variation.** Another property of text is that characters in a text line are distributed uniformly, i.e. the distances between consecutive characters have small deviation. The distance between two consecutive components is the distance of their centers $O(c_{i-1})$ and $O(c_i)$.

– **Average direction bias.** For most text lines, the major orientations of the characters are nearly perpendicular to the major orientation of the text line.

– **Average axial ratio.** Some repeated patterns (e.g. barriers) that are not texts consist of long and thin components, this feature can help differentiate them from true texts.

– **Average density.** On the contrary, other repeated patterns (e.g. bricks) consist of short and fat components, this feature can be used to eliminate this kind of false positives.

– **Average width variation.** False positives formed by foliage usually have varying widths while texts have constant widths. This feature is defined as the mean of all the width variation values of the candidates.

– **Average color self-similarity.** Characters in a text line usually have similar but not identical color distributions with each other; yet in false positive chains, color self-similarities [25] of the candidates are either too high (repeated patterns) or too low (random clutters). The color similarity $cs(x, y)$ is defined as the cosine similarity of the color histograms of the two candidates x and y.

– **Average structure self-similarity.** Likewise, characters in a text line have similar structure with each other while false positives usually have almost the same structure (repeated patterns) or diverse structures (random clutters). The structure similarity $ss(x, y)$ is defined as the cosine similarity of the edge shape descriptors of the two components x and y.

4. Dataset and Evaluation Protocol

In this section, we introduce a dataset for evaluating text detection algorithms, which contains images of real-world complexity; a new evaluation method is also proposed.

Although widely used in the community, the ICDAR dataset [21, 20] has two major drawbacks. First, most of the text lines (or single characters) in the ICDAR dataset are horizontal. In real scenarios, however, text may appear in any orientation. The second drawback is that all the text lines or characters in this dataset are in English. These two shortcomings are also pointed out in [23, 29]. In this work, we generate a new multilingual image dataset with horizontal as well as skewed and slant texts. We name this dataset

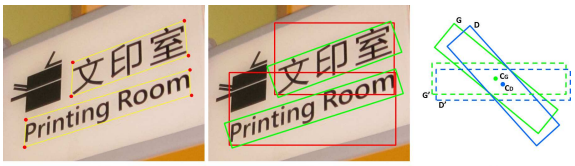

Figure 6. Ground truth generation and overlap ratio calculation. (a) Human annotations. The annotators are required to bound each text line using a four-vertex polygon (red dots and yellow lines). (b) Ground truth rectangles (green). The ground truth rectangle is generated automatically by fitting a minimum area rectangle using the polygon. (c) Calculation of overlap ratio between detection rectangle and ground truth rectangle.

MSRA Text Detection 500 Database (MSRA-TD500)[2], because it contains 500 natural images in total. These images are taken from indoor (office and mall) and outdoor (street) scenes using a packet camera. The indoor images are mainly signs, doorplates and caution plates while the outdoor images are mostly guide boards and billboards in complex background. The resolutions of the images vary from 1296×864 to 1920×1280. Some typical images from this dataset are shown in Fig. 8 (a).

This dataset is very challenging because of both the diversity of the texts and the complexity of the backgrounds in the images. The texts may be in different languages (Chinese, English or mixture of both), fonts, sizes, colors and orientations. The backgrounds may contain vegetation (e.g. trees and grasses) and repeated patterns (e.g. windows and bricks), which are not so distinguishable from text.

The dataset is divided into two parts: training set and test set. The training set contains 300 images randomly selected from the original dataset and the rest 200 images constitute the test set. All the images in this dataset are fully annotated. The basic unit in this dataset is text line rather than word, which is used in the ICDAR dataset, because it is hard to partition Chinese text lines into individual words based on their spacings; even for English text lines, it is non-trivial to perform word partition without high level information. The procedure of ground truth generation is shown in Fig. 6 (a) and (b).

Minimum area rectangles [9] are used in our protocol because they (green rectangles in Fig. 6 (b)) are much tighter than axis-aligned rectangles (red rectangles in Fig. 6 (b)). However, a problem imposed by using minimum area rectangles is that it is difficult to judge whether a text line is correctly detected. As shown in Fig. 6 (c), it is not trivial to directly compute the overlap ratio between the estimated rectangle D and the ground truth rectangle G. Instead, we compute the overlap ratio using axis-aligned rectangles G' and D', which are obtained by rotating G and D round their centers C_G and C_D, respectively. The overlap ratio between G and D is defined as: $m(G, D) = \dfrac{A(G' \cap D')}{A(G' \cup D')}$ where $A(G' \cap D')$ and $A(G' \cup D')$ denote the areas of the intersection and union of G' and D'. Similar to the evalu-

[2] $http://users.loni.ucla.edu/{\sim}ztu/Download\_front.htm$

Figure 7. Detected texts in images from the ICDAR test set.

ation method for the PASCAL object detection task [8], in our protocol detections are considered true or false positives based on the overlap ratio between the estimated minimum area rectangles and the ground truth rectangles. If the included angle of the estimated rectangle and the ground truth rectangle is less than $\pi/8$ and their overlap ratio exceeds 0.5, the estimated rectangle is considered a correct detection. Multiple detections of the same text line are taken as false positives. The definitions of precision and recall are: $precision = |TP|/|E|$, $recall = |TP|/|T|$ where TP is the set of true positive detections while E and T are the sets of estimated rectangles and ground truth rectangles. The F-measure, which is a single measure of algorithm performance, is a combination of the two above measures: $f = 2 \cdot precision \cdot recall/(precision + recall)$.

5. Experiments

We implemented the proposed algorithm and trained two text detectors, one on the mixture of the ICDAR training set and the training set of the proposed dataset, and the other only on the ICDAR training set. These two text detectors are denoted by TD-Mixture and TD-ICDAR, respectively. 200 trees are used for training the component level classifier and 100 trees for the chain level classifier. The threshold values are: $T_1 = 0.1$, $T_2 = 0.3$ and $T = 0.4$. We found empirically that the text detectors under this parameter setting work well for all the datasets used in this paper.

In order to compare the proposed algorithm with existing methods, we evaluated the algorithm on the standard benchmark ICDAR dataset [21, 20]. The ICDAR dataset contains 509 fully annotated text images. 258 images from the dataset are used for training and 251 for testing.

Some text detection examples of the proposed algorithm are presented in Fig. 7. The algorithm can handle several types of challenging scenarios, e.g. variations in text font, color and size, as well as repeated patterns and background clutters. The quantitative comparisons of different methods evaluated on the ICDAR test set are shown in Tab. 2. Our algorithm compares favorably with the state-of-the-art algorithms when dealing with horizontal texts.

Besides the ICDAR dataset, we also tested the proposed algorithm and the systems of Chen et al. [6] and Epshtein et al. [7] on the proposed dataset. Examples of our algorithm on this dataset are shown in Fig. 8 (a). Our algorithm is able to detect texts of large variation in natural scenes, with the presence of vegetation and buildings. The images in the last row of Fig. 8 (a) are some typical cases where our algorithms failed to detect the texts or gave false positives. The misses (pink rectangles) are mainly due to strong highlights, blur and low resolution; the false positives (red rectangles) are usually caused by windows, trees, or signs that are very alike text.

Table 2. Performances of different text detection methods evaluated on the ICDAR test set.

Algorithm	Precision	Recall	F-measure
TD-Mixture	0.69	0.66	**0.67**
TD-ICDAR	0.68	0.66	0.66
Epshtein et al. [7]	**0.73**	0.60	0.66
Yi et al. [29]	0.71	0.62	0.62
Becker et al. [20]	0.62	**0.67**	0.62
Chen et al. [6]	0.60	0.60	0.58
Zhu et al. [20]	0.33	0.40	0.33
Kim et al. [20]	0.22	0.28	0.22
Ezaki et al. [20]	0.18	0.36	0.22

Table 3. Performances of different text detection methods evaluated on the proposed dataset.

Algorithm	Precision	Recall	F-measure
TD-Mixture	**0.63**	**0.63**	**0.60**
TD-ICDAR	0.53	0.52	0.50
Epshtein et al. [7]	0.25	0.25	0.25
Chen et al. [6]	0.05	0.05	0.05

Table 4. Performances of different text detection methods evaluated on the Oriented Scene Text Database (OSTD) [29].

Algorithm	Precision	Recall	F-measure
TD-Mixture	**0.77**	**0.73**	**0.74**
TD-ICDAR	0.71	0.69	0.68
Yi et al. [29]	0.56	0.64	0.55
Epshtein et al. [7]	0.37	0.32	0.32
Chen et al. [6]	0.07	0.06	0.06

The performances are measured using the proposed evaluation protocol and shown in Tab. 3. Our algorithm achieves significantly enhanced performance when detecting texts of arbitrary orientations. The performances of other competing algorithms are not presented because of unavailability of their executables. The average processing time of our algorithm on this dataset is 7.2s and that of Epshtein et al. is 6s (both tested on a 2.53GHz CPU without optimization). Our algorithm is a bit slower, but with the advantage of being able to detect multi-oriented texts.

In [29], a dataset called Oriented Scene Text Database (OSTD), which contains texts of various orientations, is released. This dataset contains 89 images of logos, indoor scenes and street views. We perform text detection on all the images in this dataset. The quantitative results are presented in Tab. 4. Our method outperforms [29] on the Oriented Scene Text Database (OSTD), with an improvement of 0.19 in F-measure.

(a) (b)

Figure 8. (a) Detected texts in images from the proposed dataset. Yellow rectangles: true positives, pink rectangles: false negatives, red rectangles: false positives. Best viewed in color. (b) Detected texts in various languages in images collected from the internet. Note that the texts are detected in full images. We only show cropped sub images because of space limitation.

From Tab. 3 and Tab. 4, we observe that even TD-ICDAR (only trained on horizontal texts) achieves much better performance than other methods on non-horizontal texts. It demonstrates the effectiveness of the proposed features. Fig. 8 (b) shows some detected texts in various languages, including both oriental and western languages, such as Japanese, Korean, Arabic, Greek, and Russian. Though our text detector is only trained on Chinese and English texts, it can effortlessly generalize to texts in different languages. It indicates that the proposed algorithm is quite general and it can serve as a multilingual text detector if sufficient training examples are available.

6. Conclusions and Future Work

We have presented a text detection system that detects texts of arbitrary directions in natural images. Our system compares favorably with the state-of-the-art algorithms when handling horizontal texts and achieves significantly enhanced performance on texts of arbitrary orientations in complex natural scenes.

The component level features are actually character descriptors that can distinguish among different characters, thus they can be adopted to recognize characters. We plan to make use of this property and develop an unified framework for text detection and character recognition in the future.

Acknowledgment

This work was supported by National Natural Science Foundation of China (grant No. 61173120 and 60903096), Office of Naval Research Award N000140910099 and NSF CAREER award IIS-0844566.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI*, 2002.

[2] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proc. CIVR*, 2007.

[3] G. R. Bradski. Real time face and object tracking as a component of a perceptual user interface. In *Proc. IEEE Workshop on Applications of Computer Vision*, 1998.

[4] L. Breiman. Random forests. *Machine Learning*, 2001.

[5] J. F. Canny. A computational approach to edge detection. *IEEE Trans. PAMI*, 1986.

[6] X. Chen and A. Yuille. Detecting and reading text in natural scenes. In *Proc. CVPR*, 2004.

[7] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Proc. CVPR*, 2010.

[8] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[9] H. Freeman and R. Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Comm. ACM*, 1975.

[10] J. Gllavata, R. Ewerth, and B. Freisleben. Text detection in images based on unsupervised classification of high-frequency wavelet coefficients. In *Proc. ICPR*, 2004.

[11] C. Gu, J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *Proc. CVPR*, 2009.

[12] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: Data mining, inference, and prediction, second edition. *New York: Springer*, 2009.

[13] X. S. Hua, W. Liu, and H. J. Zhang. An automatic performance evaluation protocol for video text detection algorithms. *IEEE Trans. CSVT*, 2004.

[14] A. Jain and B. Yu. Automatic text location in images and video frames. *PR*, 1998.

[15] K. Jung, K. Kim, and A. Jain. Text information extraction in images and video: a survey. *PR*, 2004.

[16] K. I. Kim, K. Jung, and J. H. Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Trans. PAMI*, 2003.

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.

[18] H. P. Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. *IEEE Trans. IP*, 2000.

[19] J. Liang, D. Doermann, and H. Li. Camera-based analysis of text and documents: a survey. *IJDAR*, 2005.

[20] S. M. Lucas. Icdar 2005 text locating competition results. In *Proc. ICDAR*, 2005.

[21] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *Proc. ICDAR*, 2003.

[22] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Proc. of ACCV*, 2010.

[23] Y. Pan, X. Hou, and C. Liu. A hybrid approach to detect and localize texts in natural scene images. *IEEE Trans. IP*, 2011.

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

[25] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Proc. CVPR*, 2007.

[26] P. Shivakumara, T. Q. Phan, and C. L. Tan. A laplacian approach to multi-oriented text detection in video. *IEEE Trans. PAMI*, 2011.

[27] K. Wang and S. Belongie. Word spotting in the wild. In *Proc. ECCV*, 2010.

[28] X. Wang, X. Bai, W. Liu, and L. J. Latecki. Feature context for image classification and object detection. In *Proc. CVPR*, 2010.

[29] C. Yi and Y. Tian. Text string detection from natural scenes by structure-based partition and grouping. *IEEE Trans. IP*, 2011.

[30] M. Zhao, S. T. Li, and J. Kwok. Text detection in images using sparse representation with discriminative dictionaries. *IVC*, 2010.