Neurocomputing ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Text string detection for loosely constructed characters with arbitrary orientations

Yong Zhang [a,c], Jianhuang Lai [a,c,*], Pong C. Yuen [b]

[a] School of Information Science and Technology, Sun Yat-Sen University, China
[b] Department of Computer Science, Hong Kong Baptist University, Hong Kong, China
[c] Guangdong Key Laboratory of Information Security Technology, Guangzhou, China

ARTICLE INFO

Article history:
Received 14 April 2014
Received in revised form
24 January 2015
Accepted 8 May 2015

Keywords:
Text detection
Text string
Stroke width transform
k-Nearest neighbors
Complex character

ABSTRACT

Text in a scene provides vital information of its contents. With the increasing popularity of vision systems, detecting general text in images becomes a critical yet challenging task. Most existing methods have focused on extracting neatly arranged text string for compactly constructed characters. Motivated by the need to consider the widely varying forms of scene text, we propose a stroke-based text detection method which detects arbitrary orientations text strings with loosely constructed characters in images. Our approach employs result of stroke width transform (SWT) as basic stroke candidates. These candidates are then merged using adaptive structuring elements to generate compactly constructed characters. Individual characters are chained using *k*-nearest neighbors algorithm to identify arbitrary orientations text strings, which are subsequently separated into words if necessary. To better evaluate our system and compare it with other competing algorithms, we generate a new dataset, which includes various characters and text strings in diverse scenes. Experiments on ICDAR datasets and the proposed dataset demonstrate that our approach compares favorably with the state-of-the-art algorithms when handling arbitrary orientations text strings and achieves significantly enhanced performance on loosely constructed characters in scenes.

1. Introduction

With the rapid development of digital technology and the broad use of Internet, the amount of multimedia resources such as image and video is increasing dramatically. Collating these multimedia data based on their contents is a big challenge for people. Manual labeling process is laborious and time consuming. It is desirable to develop the automatic methods for annotation and indexing of these multimedia resources.

A variety of techniques have been put forward to extract the content information from image [1]. Among these techniques, text detection is an effective and efficient method because of the rich information contained in the caption and scene [2]. Although many text detection approaches have been proposed in the last decades [3], detecting general text in scene is still a challenging task because the algorithms often suffer from multilingual text [4], orientation [5], and complexly arranged text string [6].

A text detection process is usually divided into three steps, namely, stroke detection, character extraction, and text string aggregation. Because different languages may have quite different appearances, few methods can handle multilingual text well. Some stroke-based methods claim the multilingual ability in the stroke detection step [7]. However, the stroke differences of different languages are not as critical, whereas the structural differences of them are very critical since they strongly affect the results of character extraction and text string aggregation. From the linguistic point of view, alphabetic literal such as English, Spanish, and German usually has compactly constructed characters, whereas ideograph such as Chinese, Japanese, and Korean is generally constituted by loosely constructed characters. The connected component (CC) based text detection methods [8–10] provides us with ample evidence to support this point. Alphabetic literal characters do not contain many connected components (CCs), this will not affect the character/letter extraction too much. However, ideograph characters contain many CCs and these structures increase the difficulty of character extraction. This fact can be demonstrated by Fig. 1(a), in which we use English and Chinese as an example.

Another major challenge of existing stroke-based text detection methods is aggregating arbitrary orientations text strings. Although this problem for neatly arranged text is usually seen as a solved problem, freestyle-arranged text still present a significant challenge because curvilinear text string and small gap between neighboring text strings present a challenge to aggregation algorithms developed for neatly arranged text string (see Fig. 1(b)).

* Corresponding author.
E-mail addresses: yzhang.user@gmail.com (Y. Zhang), stsljh@mail.sysu.edu.cn (J. Lai), pcyuen@comp.hkbu.edu.hk (P.C. Yuen).

http://dx.doi.org/10.1016/j.neucom.2015.05.028
0925-2312/

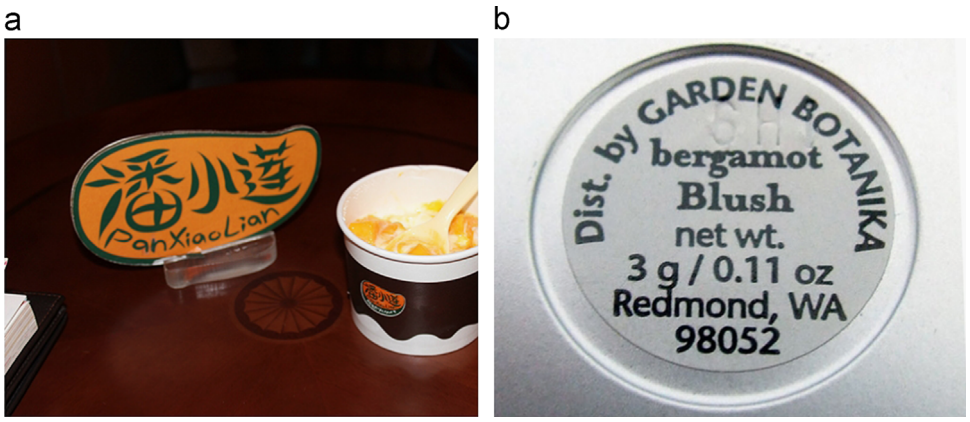


Fig. 1. Two major challenges of existing stroke-based text detection methods. (a) One challenge is extracting loosely constructed character, such as Chinese character. A Chinese character typically contains many CCs, whereas an English letter typically contains one CC (except lowercase "i" and "j"). (b) Another challenge is aggregating arbitrary orientations text strings.

Our goal is to develop a stroke-based method to detect arbitrary orientations text strings with loosely constructed characters in images. The proposed method consists of three steps. In the first step, the method employs the result of stroke width transform (SWT) as basic stroke candidates. The second step is merging these stroke candidates into characters using adaptive structuring elements. Finally, the individual characters are chained using *k*-nearest neighbors algorithm to identify arbitrary orientations text strings. Though widely used in the community, the ICDAR datasets [11,12] only contain horizontal English texts. In [5,6], two datasets with texts of different directions are released, but they include simple scenes without enough diversity in the languages and arrangements. Here we collect a new dataset with 400 images, which includes various characters and text strings in diverse scenes.

The main contributions and innovations of this paper are as follows:

- We propose a novel character extraction algorithm, in which the previous state-of-the-art work SWT is employed to extract possible text characters in the form of CCs in consistent stroke width, and dynamic structural element is defined for morphological processing to dilate the extracted text characters. The key idea of this operation is intended to solve detection problem of loosely constructed characters, which has not been particularly addressed in most of previous work.
- We investigate a novel method for text string aggregation, which is performed by searching *k*-nearest neighbors of each CC and building search path as a trace of text string in arbitrary directions. Text strings affected by curvature, inclination, and interleaving are robustly detected by the proposed aggregation algorithm.

The rest of the paper is organized as follows. We review related works in Section 2. The detail of the proposed approach is presented in Section 3. Experimental results and comparative studies are presented in Section 4. The paper is concluded with a discussion of future work in Section 5.

2. Related work

According to the features used and the ways they work, text detection approaches can be classified into two categories: texture-based methods and region-based methods.

Texture-based methods treat texts as a special type of texture and use distinct texture properties of text, such as local intensities, filter responses and wavelet coefficients. Some widely used features include local binary patterns (LBP), histograms of gradients (HOGs), Gabor filters and wavelets. These methods are computation demanding as all locations and scales are exhaustively scanned. Some machine learning methods are often used in this approach. Kim et al. [13] use a support vector machine (SVM) to analyze the textural properties of texts. The intensities of the raw pixels are fed directly to the SVM. No external feature extractor is required to reduce the dimensionality of the texture pattern, because SVMs work well even in high dimensional spaces. Yan et al. [14] present a text location method under complex background by combining Gabor filter and SVM. Four kinds of stroke features were extracted using Gabor filters, then the text location problem can be transformed into a texture classification problem, which can use SVM classifier for the purpose. Sin et al. [15] present a method of finding text regions using frequency-based features estimated. The Fourier spectrum is used to estimate the fundamental frequency of the text images. Zhong et al. [16] propose a caption localization method using discrete cosine transform (DCT) coefficients. The directionality and periodicity of local image blocks are used as texture measure to identify text regions. Li et al. [17] propose a text extraction scheme using key text points which can be acquired by the high-frequency sub-bands obtained from the wavelet transform. Shivakumara et al. [18] present a text detection method comprising of wavelet decomposition and color features. The wavelet decomposition is applied to obtain high-frequency sub-bands and then the average of the three sub-bands is computed further to enhance the text pixels. Texture-based methods are efficient in dealing with complex background. However, there is something unsatisfying about this kind of approaches, since the brute force nature of window classification is not particularly appealing and the computational complexity is proportional to the number of scales.

Region-based methods, on the other hand, first extract candidate text regions through edge detection [10], color clustering [6], or maximally stable extremal region (MSER) detection [19–21] and then eliminate non-text regions using various heuristic rules. Edge detection utilizes the geometry and structural properties of the character, since text regions contain a large number of edges. Connected component growing techniques, such as maximum difference (MD) [22] and morphological dilation, are used to group neighboring strokes with similar properties [23]. A typical example of region-based approaches was proposed by Epshtein et al. [7], where the stroke width transform is used to find the value of stroke width for each image pixel and non-characters was removed by a series of rules. Yao et al. [5] adopt SWT and also design various features that are intrinsic to texts and robust to

Please cite this article as: Y. Zhang, et al., Text string detection for loosely constructed characters with arbitrary orientations, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.05.028

variations. Lyu et al. [4] propose a multilingual text detection method which is carried out by edge detection, local thresholding, and hysteresis edge recovery. By the observation that there exists dense and orderly presences of corner points in characters, especially in text and caption, Zhao et al. [24] present a corner based approach to detect text. They use several discriminative features to describe the text regions formed by the corner points. Their method can only deal with the horizontal or vertical text. Shivakumara et al. [25] use sharpness in filter-edge maps, straightness of the edges, and proximity of the edges to identify a true text frame. Region-based methods are efficient when the background mainly contains uniform texts. Because this kind of approach is built on the basic assumption that the text is represented with compactly constructed characters and the text strings are neatly arranged, they may not preserve the full shape of the texts when the characters are loosely constructed or the text strings are freestyle-arranged.

In spite of the variety of methods available, they all suffer from some limitations. None of them has designed adaptive structuring elements for scene text consisting of loosely constructed characters. We argue that adopting a fixed-size structuring element to scene text detection [10,24] is inappropriate, as general loosely constructed characters are likely to be distracted by the structure that one character is composed of several connected components. Another common problem is that current methods do not properly consider global structure of text strings. To our knowledge, none of the proposed methods in the literature tries to define a reasonable aggregation strategy for the freestyle-arranged text strings which are affected by curvature, inclination, and interleaving. To solve these problems, in this paper, we propose a novel text detection method, which employs stroke-width-related structuring elements to solve the problem of loosely constructed character extracting and uses a path searching approach to solve the problem of freestyle-arranged text strings aggregating.

3. Proposed approach

The proposed algorithm consists of three stages: stroke detection, character extraction, and text string aggregation, as shown in Fig. 2. In the stroke detection stage, the SWT is used to detect basic stroke candidates. In the character extraction stage, the filtered stroke candidates are merged using a morphological-based dilation with adaptive structuring element. The structuring element of each stroke candidate is calculated according to its median stroke width. In the text string aggregation stage, the individual characters are chained using k-nearest neighbors algorithm to identify arbitrary orientations text strings. In order to improve the aggregation results, the algorithm filtered the unreasonable candidate blocks using some additional criteria. This process is called false positive elimination.

3.1. Stroke detection using SWT

SWT has been a widely exploited preprocessing for stroke detection. The main idea of SWT is to find pairs of corresponding text pixels. The method first computes the edges of the input image using Canny edge detector (see Fig. 3(a)). For each edge pixel, if a ray starting from it with the gradient direction meets another edge pixel with opposite gradient direction, the two pixels are considered to be a pair of corresponding text pixels. A stroke width is defined as the length of a straight line between two edge pixels of stroke in the perpendicular direction. The output of SWT is an image of size equal to the size of input image where each element contains the width of the stroke associated with the pixel, as shown in Fig. 3(b).

These stroke candidates are then grouped into letter candidates using a set of general rules. In this stage, we use the parameters reported from the original SWT paper [7]. To handle both bright-on-dark and dark-on-bright texts, we apply SWT algorithm twice, once by following the gradient directions of edge pixels and once by following the inverse gradient directions. Fig. 4(a) clearly shows how effective these rules can be in finding letter candidates in a raw SWT result (Fig. 3(b)).

3.2. Character extraction

The next state of the method is to group these pixels detected by SWT into character candidates. A morphology dilation operation is performed on the remaining CCs to extract characters. Because loosely constructed character with different stroke width may have different stroke gap. We modify the classical morphological-based dilation algorithm [26] by changing the structuring element from fixed-size to dynamic-size. A structuring element is simply a binary image (or mask) that allows us to define arbitrary neighborhood structures. We use a disk-shaped structuring element, whose radius R is proportional to the median stroke width of current CC:

$$R(p) = \lceil k_1 * SWM(CC) \rceil, p \in CC \tag{1}$$

where p is a single variable for a coordinate pair (x, y), $p = (x, y)$, the function $\lceil \cdot \rceil$ returns a nearest integer which is greater than or equal to the input value, k_1 is a scale factor, and $SWM(CC)$ represents the median stroke width of current CC.

In order to avoid merging the strokes with different width, only the CCs which have similar median stroke width are grouped together after dilation. We can think of a binary image $I(p)$ as the set of all pixel locations with the median stroke width value between m_l and m_h in the foreground:

$$I_m = \{p \mid m_l \le I(p) < m_h\} \tag{2}$$

$$m_l = k_2 * SWM(CC) \tag{3}$$

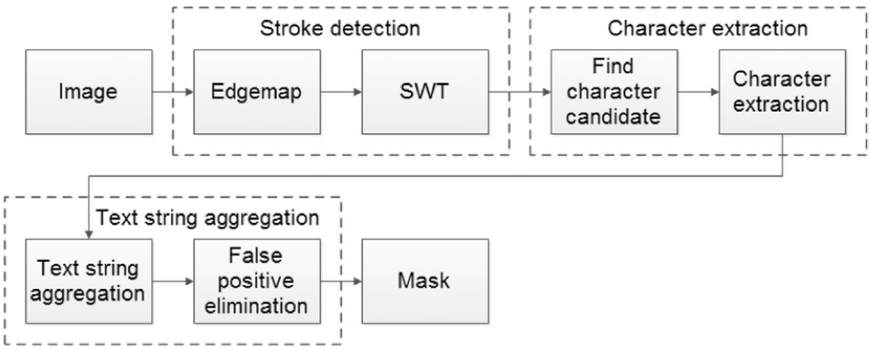


Fig. 2. Flowchart of the proposed method which consists of three stages: stroke detection, character extraction, and text string aggregation.

Please cite this article as: Y. Zhang, et al., Text string detection for loosely constructed characters with arbitrary orientations, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.05.028

Fig. 3. Stroke detection using SWT. (a) Canny edge of the original image (see in Fig. 1(a)). (b) SWT output.

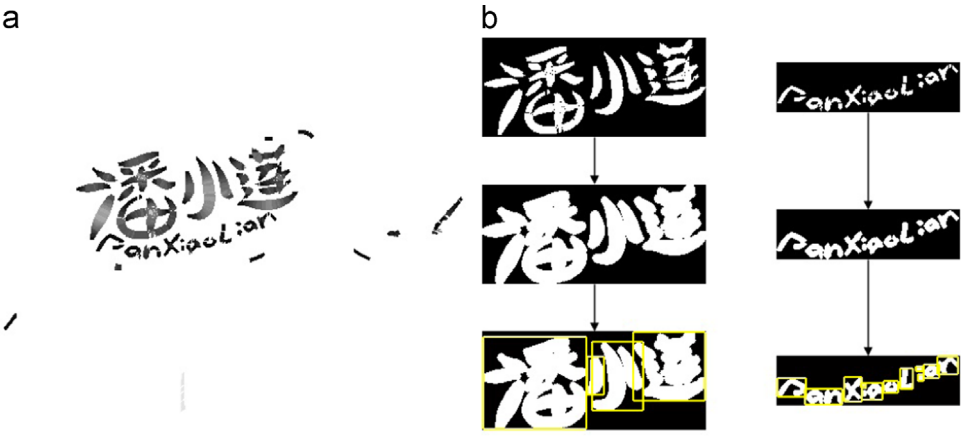


Fig. 4. Implementation of the SWT. (a) Character candidates are extracted from the original SWT output (see Fig. 3(b)) using a set of fairly flexible rules proposed by [7]. (b) The CCs which have similar median stroke width are grouped together after dilation. In each column, the first row is the original SWT output, the second row is the dilated CCs, the third row is the grouped CCs surrounded by yellow boxes. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

$$m_h = k_3 * SWM(CC) \tag{4}$$

k_2 and k_3 are two scale factor, and the value of k_1, k_2, and k_3 is learned on a fully annotated ICDAR training set by optimizing performance.

A dilation of an image I_m by the structure element SE can be given by a set operation. Taking the union of copies of the structuring element, $SE(p)$, centered at every pixel location p in the foreground, the dilation of an image I_m by the structure element SE is defined as

$$I_m \oplus SE = \bigcup_{p \in I_m} SE(p) \tag{5}$$

where

$$SE(p) = \{q \mid q \in I \wedge \|q - p\| \le R(p)\} \tag{6}$$

The results of the dilation were demonstrated in Fig. 4(b)

It should be noted that most of characters can be reasonably extracted in this stage, some of them may still be incorrectly divided, such as one character may be divided into a plurality of CCs or one CC may contain more than one character. This has little influence on the text string aggregation, or can be fixed in the subsequent process (see Section 3.3.3).

3.3. Text string aggregation

Because text often appear in linear or slight curvilinear form, text strings are important cues for the existence of text. Although text string aggregation for caption or neatly arranged text is usually seen as a solved problem, freestyle-arranged text string aggregation remains an open problem. In this section, we investigate a path searching approach using k-nearest neighbors algorithm to aggregate text string affected by curvature, inclination, and interleave. The interleaved text strings are separated using a skeleton-based segmentation. A heuristic that computes a histogram of distances between consecutive characters and text lines is used to further separate the characters which are merged incorrectly before. Finally, some additional criteria are used to filter unreasonable text string candidates in the false positive elimination.

3.3.1. Path searching using k-nearest neighbors algorithm

Characters belonging to the same text string are linked into a path, in which each two neighboring characters are assumed to have similar median stroke width and character height.

To activate this, linking the neighboring characters is the first step. Because in most cases a character can be represented as a CC after the character extraction (see Section 3.2). Two neighboring CCs can be linked if the ratio of their stroke width medians is small and their height ratio is not too big (e.g. the height ratio is lower than 2 taking upper and lower case letters into account). Therefore, for each CC, we first calculate its centroid, then find its nearest neighbor and second nearest neighbor according to the centroid distance using the k-nearest neighbors (k-NN) algorithm.

Please cite this article as: Y. Zhang, et al., Text string detection for loosely constructed characters with arbitrary orientations, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.05.028

Additionally, two CCs should not be linked if they are very distant. The procedure is outlined in Algorithm 1.

Parameters and functions of Algorithm 1 with the definition are as follows: n is the total number of CCs. *centroid* is a $n*2$ matrix used to store the centroid of CCs. m_i is the total number of pixels of i-th CCs. x_j and y_j are the horizontal and vertical coordinates of j-th pixel of a CCs, respectively. *neighbor* is a $n*2$ matrix used to store the neighboring relationship of CCs. $zeros(n,2)$ is a function used to generate a $n*2$ matrix with zero elements. $knnsearch(centroid(i), centroid, k)$ is a k-NN function used to search the k-th nearest neighbor of $centroid(i)$ in the set *centroid*, and returns its label. $nearest_1$ and $nearest_2$ are used to store the label of the nearest neighbors. The function H is a used to calculate the height of input CCs defined by its label. The function R_MSW is a used to calculate the ratio of stroke width medians of two input CCs defined by their labels. Similarly, R_H and D are two functions used to calculate the height ratio and centroid distance of two input CCs, respectively. t_1, t_2, and t_3 are three thresholds learned by optimizing performance on the ICDAR training set, as described in Section 3.2.

Algorithm 1. Linking neighboring characters using k-NN algorithm.

```
begin
  for i=1 to n
    centroid(i) = (∑_{j=1}^{m_i}(x_j, y_j))/m_i;
  end
  neighbor = zeros(n, 2);
  for i=1 to n
    nearest_1 = knnsearch(centroid(i), centroid, 1);
    nearest_2 = knnsearch(centroid(i), centroid, 2);
    if (R_MSW(i, nearest_1) <= t_1 and R_H(i, nearest_1) <= t_2
      and D(i, nearest_1) <= t_3*H(i))
    neighbor(i, 1) = nearest_1;
  end
  if (R_MSW(i, nearest_2) <= t_1 and R_H(i, nearest_2) <= t_2
    and D(i, nearest_2) <= t_3)
  neighbor(i, 2) = nearest_2;
  end
  end
  return neighbor;
end
```

The output of Algorithm 1 is the neighboring relationship matrix *neighbor* in which the first and second columns of i-th row are assigned two values equal to the labels of the first and second nearest neighbors of i-th CC, respectively. Fig. 5 shows an example of linking neighboring characters in a freestyle-arranged text scene. In Fig. 5(c), the yellow rectangles represent the minimum bounding box of CCs, the green asterisks represent the centroid of CCs, the red edges represent the neighboring relationship between two CCs.

At the second step of the algorithm, the neighboring CCs determined above are clustered together into chains. Two CC pairs can be merged together if they share one end and have similar direction ($d_2 = d_1 \pm \pi/3$, where d_1 and d_2 are the two directions, respectively). A chain is declared to be a text string if it contains sufficient characters (at least 2 in our experiments).

3.3.2. Segmenting interleaved text strings

In order to output single text string of a complex character chain which includes interleaved text strings, we need to segment, or split it into multiple simple chains, based on the intersection points. In Fig. 5(d), point A shows the location where the first text chain of Fig. 5(c) connects to the second text chain. By segmenting the complex character chain from B to C, we are able to get back the first character chain. BC is called a chain segment, which is defined as a continuous path from an end point to either intersection point and or another end point. Moreover, the path should not include any other intersection point in the middle. A simple way to find the chain segments is to delete all of the neighboring relationship of intersection points that have neighborhood only in one direction, such as AB and DE in Fig. 5(d).

3.3.3. Separating merged characters

The text string segmented above can be broken into separate characters using a heuristic that computes a histogram of distances between consecutive CCs of the original SWT output and estimates the distance threshold that can used to separate characters. While the general text detection problem does not require this step, we do it in order to further separate some incorrect text merging results caused by the morphology dilation, as mentioned in Section 3.2. In addition, sometimes we do it just for comparing the performance of algorithm with the ones in ICDAR database [27,28,12].

To overcome the arbitrary orientation text string, the distances between consecutive CCs in the histogram computation should be measured in the direction of text string, rather than the horizontal direction. We calculate the axis of minimal moment of inertia for each text string. Suppose X and Y are the abscissa and ordinate set of a text string, respectively. $x \in X$, $y \in Y$. C is the covariance matrix of X and Y:

$$C = \begin{bmatrix} E[x-\mu_x]^2 & E[x-\mu_x][y-\mu_y] \\ E[x-\mu_x][y-\mu_y] & E[y-\mu_y]^2 \end{bmatrix} \tag{7}$$

where E is the mathematical expectation and $\mu_x = E(x)$, $\mu_y = E(y)$. The axis of minimal moment of inertia for a text string is the straight line through the center of gravity of the text string and parallel to the eigenvector related maximum eigenvalue of the matrix C (denoted as V_{max}). The angle between the direction of text string and horizontal direction is equal to the angle between V_{max} and horizontal axis. Fig. 6 shows an example of separating merged characters using heuristic.

3.3.4. False positive elimination

Finally, in order to improve the final detection results, the algorithm needs to further filter the unreasonable candidate text strings which are not likely to be generated by texts from a global perspective. This process is called false positive elimination.

First of all, the repeating structures such as bricks and windows can be filtered by applying template matching among the CCs of characters in a text string. A text string is rejected if most of the CCs are repetitive. The template matching method comprises three steps: image block extraction, image block alignment, and difference calculation between template image I_T and search image I_S, as shown in Fig. 7. For each pair of adjacent CCs of characters in a text string, we extract two image blocks according to their minimum bounding boxes. The first image block is called template image and the second image block is called search image. We then scale the search image to the same size of the template image using bilinear interpolation. This operation is called image block alignment. The aligned template image is represented as $I_T(x,y)$, where (x, y) represent the coordinates of each pixel. Similarly, the aligned search image is represented as $I_S(x,y)$. The difference between I_T and I_S can be defined as

$$Diff = \sum_{x=1}^{W_I} \sum_{y=1}^{H_I} (I_S(x,y) - I_T(x,y))^2 \tag{8}$$

Please cite this article as: Y. Zhang, et al., Text string detection for loosely constructed characters with arbitrary orientations, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.05.028

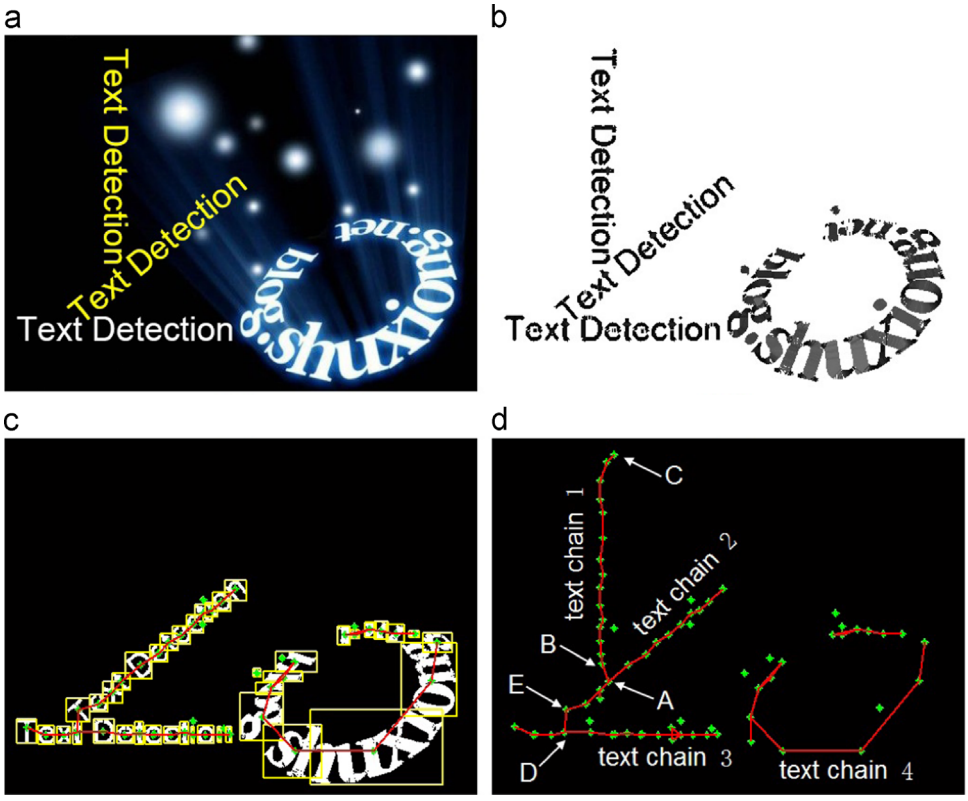


Fig. 5. Text string aggregation using *k*-NN algorithm. (a) Input image. (b) SWT output. (c) Linking neighboring characters using *k*-NN algorithm. (d) Segmenting character chain. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 6. Separating merged characters using heuristic. (a) Original SWT output. (b) Histogram of SWT output along the direction of text string, in which the *x*-axis and *y*-axis represent the value of gray and number of pixels, respectively.

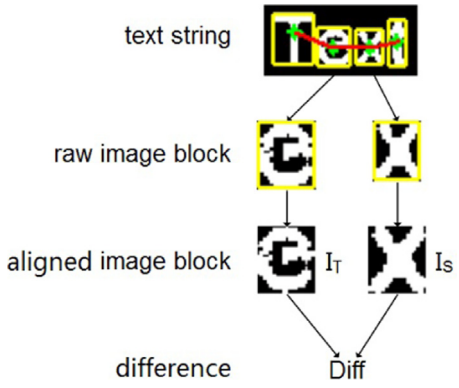


Fig. 7. Template matching among the CCs of characters in a text string.

where W_I and H_I are the width and the height of the template image, respectively. Low *Diff* score gives the estimate for high matching between the template image and search image.

Besides, a text string is rejected if most of the CCs within it have a very high saturation based on the observation that most characters have low saturation (very few characters consist of a single horizontal or vertical straight line). The saturation of CC indicates the proportion of the CC pixels in its minimum bounding box. The saturation of cc_i can be calculated by

$$Sat(cc_i) = \frac{Area(cc_i)}{Box\_W(cc_i) * Box\_H(cc_i)} \tag{9}$$

where $Sat(cc_i)$ indicates the saturation of the *i*-th CC. $Area(\cdot)$ indicates the area of input region, which is equal to its total number of pixel (in binary image). $Box\_W(\cdot)$ and $Box\_H(\cdot)$ are the width and the height of minimum bounding box of input CC, respectively.

4. Experiments

To better evaluate our system and compare it with other competing algorithms, we applied it to several images in ICDAR text locating datasets [12] as well as our own dataset. The parameters of our algorithm were learned by optimizing performance on the ICDAR training set. The parameters of the existing methods are set according to the recommended values in the respective papers. There are six parameters in the proposed algorithm. The parameter values in the corresponding formula are shown in Table 1. The values of these parameters are consistent in all of our experiments. The parameters of the existing

Please cite this article as: Y. Zhang, et al., Text string detection for loosely constructed characters with arbitrary orientations, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.05.028

methods are set according to the recommended values in the respective papers.

4.1. Experiment on ICDAR text locating dataset

Unlike many other unavailable custom datasets, the well-known ICDAR dataset, as a publicly available dataset, remains the most widely used benchmark for text detection. We first apply our algorithm to the ICDAR text locating datasets, which was also used as a benchmark for [5–7,20,23].

Fig. 8 illustrates some results of our algorithm. Detected text regions are surrounded by yellow bounding boxes. Estimated text strings are chained along their centroids using red lines. The set of ground truth is provided in the dataset, and marked in the image using yellow boxes. With the proposed method, most text regions are detected, meanwhile, few false positives left. Note that the proposed method rejects single character CC, since it is designed to detect text string (at least 2 characters).

4.2. Experiments on our own datasets

Although widely used in the community, the ICDAR dataset has two major drawbacks [5,6,29]. First, most of the text strings in the ICDAR dataset are horizontal. Second, all the characters are in English. In real scenes, however, text may appear in any orientation and in any language.

In this work, we generate a new multilingual image dataset with arbitrary oriented text strings. The new dataset contains 400 indoor and outdoor images in total. The images were selected from Internet, video clips, and photographs. The variation range of resolutions of these images is from 320∗240 to 1920∗1280. Some typical images from this dataset are shown in Fig. 9. This dataset is very challenging because of both the structure of the characters and the complexity of the orientations of the text strings in the images. The texts may be in different languages with loosely constructed characters (Chinese, Japanese, Korean or mixture of them) and orientations. All the images in this dataset are fully annotated. The basic unit in this dataset is text string rather than individual word, which is used in the ICDAR dataset, because it is hard to partition ideograph text strings (e.g. Chinese, Japanese, and Korean) into individual words. The dataset was internally published to evaluate the performance of the proposed text detection algorithm due to copyright issue.

Some text detection results on several images from the dataset are shown in the first row of Fig. 9. The procedure of ground truth

Table 1
Parameters settings of the proposed method.

Parameter	k_1	k_2	k_3	t_1	t_2	t_3
Value	1.5	0.5	2	1.5	2	2
Related formula	(1)	(3)	(4)	Algorithm 1		

Table 2
Results on ICDAR 2013 dataset.

Algorithm	*Recall*	*Precision*	f-*Measure*
Epshtein (CVPR 2010) [7]	0.64	0.74	0.69
Li (TIP 2014) [20]	**0.80**	0.63	0.70
Yin (TPAMI2014) [19]	0.66	**0.88**	0.76
Zamberletti (IWRR 2014) [21]	0.70	0.86	**0.77**
Proposed method	0.66	0.77	0.71

The bold values represent the best results.

Fig. 8. Detected text strings in images from the ICDAR test set. The first row is the output of our method. The second row is the ground truth provided by ICDAR datasets. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 9. Some text detection results on several images from our own dataset.

Please cite this article as: Y. Zhang, et al., Text string detection for loosely constructed characters with arbitrary orientations, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.05.028

Table 3
Performance measure of experiment on our multilingual dataset.

Algorithm	*Actual objects*	*Truly detected objects*	*Falsely detected objects*	*Detected objects with missing data*	*Recall*	*Precision*	f-*Measure*	*Misdetection rate*
Epshtein (CVPR 2010) [7]	1782	1123	459	202	0.63	0.71	0.67	0.18
Li (TIP 2014) [20]	1782	1105	451	166	0.62	0.71	0.66	0.15
Yin (TPAMI 2014) [19]	1782	1176	413	141	0.66	**0.74**	0.70	0.12
Proposed method	1782	1230	455	135	**0.69**	0.73	**0.71**	**0.11**

The bold values represent the best results.

Fig. 10. Negative detection results. The proposed method fails when text strings have ambiguous direction or when the characters are too small or excessive blur.

generation is shown in the second row of Fig. 9. This indicates that our algorithm can handle several types of challenging scenes, e.g. variations in orientation of text string, size, as well as structure of character.

4.3. Performance measures

We evaluate the performance at "word level" on ICDAR dataset and "line level" on our own dataset. The ICDAR 2013 Robust Reading Competition (Challenge 2: Reading Text in Scene Images) database [12] is a widely used database for benchmarking scene text detection algorithms. The database contains 229 training images and 233 testing images. Table 2 shows the performance of our method, the original SWT method, and three very recent methods with top score from ICDAR 2013 Competition.

Our method achieves a precision of 66%, a recall of 77% and an *f*-measure of 71% on the ICDAR 2013 dataset which outperforms the original SWT method [7], with an improvement of 2% in *f*-measure.

In order to give the fair comparison between our algorithm and relevant literatures on our new dataset, the performance of our algorithm was measured by four indicators: *recall*, *precision*, f-*measure*, and *misdetection rate*, which can be defined as follows:

$$Recall = \frac{Truly\ detected\ objects}{Actual\ objects}$$

$$Precision = \frac{Truly\ detected\ objects}{Truly\ detected\ objects + Falsely\ detected\ objects}$$

$$f\text{-}Measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$Misdetection\ rate = \frac{Detected\ objects\ with\ missing\ data}{Truly\ detected\ objects}$$

In the above formulas, the *Actual objects* means the real number of text block manually counted. The *Truly detected objects* means the detected object that contains at least one true character. The *Falsely detected objects* means the detected object that does not contain text. The *Detected objects with missing data* means the detected object that misses more than 20% of actual text region [30].

The total *actual objects* of our multilingual dataset is 1782. The performance of the proposed method on our dataset is listed in Table 3 with the existing methods. We obtained the software or binary of the reference methods [19,20] on the authors web page (https://github.com/yaoliUoA/characterness/, http://prir.ustb.edu.cn/TexStar/scene-text-detection/). For the original SWT method, we implemented the algorithm according to the corresponding paper [7], and used the parameters reported from the paper. Compared with the best algorithm we have listed that the proposed method improved the *recall* by 3%, improved the f-*measure* by 1%, and improved the *misdetection rate* by 1%.

Fig. 10 shows two typical cases where text strings were not detected correctly. These are due to ambiguous layout of text string and characters that are too small or excessive blur.

5. Conclusion and future work

We have presented a stroke-based text detection method. A stroke-width-related structuring elements was used to merge the loosely constructed stroke candidates extracted by SWT. The *k*-nearest neighbors algorithm was used to chain individual characters into arbitrary orientations text strings. The method compares favorably with the state-of-the-art algorithms when handling arbitrary orientations text string and achieves significantly enhanced performance on texts string with loosely constructed characters, such as Chinese, Japanese, and Korean.

There are several possible extensions for this work. The orderly chain level features of text string are actually a semantic-related description. It can distinguish meaningful sequence of texts among

Please cite this article as: Y. Zhang, et al., Text string detection for loosely constructed characters with arbitrary orientations, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.05.028

different characters, thus it can be adopted to assist understanding texts in scenes. We plan to make use of this property and develop an unified framework for text string recognition and high-level semantic understanding in the future.

Acknowledgments

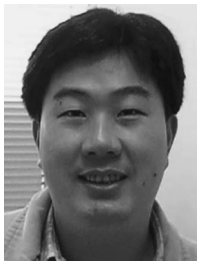
The authors would like to thank the anonymous reviewers for their insightful comments and suggestions which helped enhance this paper significantly. This work was supported by National Science & Technology Pillar Program (No. 2012BAK16B06), NSFC (61173084) and the open fund from Key Lab of Digital Signal and Image Processing of Guangdong Province (No. 54600321), and the GuangDong Program (Grant no. 2012A080104005).

References

[1] A.W. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the early years, IEEE Trans. Pattern Anal. Mach. Intell. 22 (12) (2000) 1349–1380.
[2] K. Jung, K.I. Kim, A.K. Jain, Text information extraction in images and video: a survey, Pattern Recognit. 37 (5) (2004) 977–997.
[3] H. Zhang, K. Zhao, Y.-Z. Song, J. Guo, Text extraction from natural scene image: a survey, Neurocomputing 122 (2013) 310–323.
[4] M.R. Lyu, J. Song, M. Cai, A comprehensive method for multilingual video text detection, localization, and extraction, IEEE Trans. Circuits Syst. Video Technol. 15 (2) (2005) 243–255.
[5] C. Yao, X. Bai, W. Liu, Y. Ma, Z. Tu, Detecting texts of arbitrary orientations in natural images, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2012, pp. 1083–1090.
[6] C. Yi, Y. Tian, Text string detection from natural scenes by structure-based partition and grouping, IEEE Trans. Image Process. 20 (9) (2011) 2594–2605.
[7] B. Epshtein, E. Ofek, Y. Wexler, Detecting text in natural scenes with stroke width transform, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2010, pp. 2963–2970.
[8] Y. Zhong, K. Karu, A.K. Jain, Locating text in complex color images, Pattern Recognit. 28 (10) (1995) 1523–1535.
[9] A.K. Jain, B. Yu, Automatic text location in images and video frames, Pattern Recognit. 31 (12) (1998) 2055–2076.
[10] P. Shivakumara, T.Q. Phan, C.L. Tan, A Laplacian approach to multi-oriented text detection in video, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 412–419.
[11] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young, ICDAR 2003 robust reading competitions, in: Proceedings of 7th International Conference on Document Analysis and Recognition, ICDAR, 2003, pp. 682–687.
[12] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L.G.I Bigorda, S.R. Mestre, J. Mas, D.F. Mota, J.A. Almazan, L.P. de las Heras, ICDAR 2013 robust reading competition, in: Proceedings of 12th International Conference of Document Analysis and Recognition, ICDAR, 2013, pp. 1115–1124.
[13] K.I. Kim, K. Jung, J.H. Kim, Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 25 (12) (2003) 1631–1639.
[14] J. Yan, J. Li, X. Gao, Chinese text location under complex background using Gabor filter and SVM, Neurocomputing 74 (17) (2011) 2998–3008.
[15] B.-K. Sin, S.-K. Kim, B.-J. Cho, Locating characters in scene images using frequency features, in: Proceedings of 16th International Conference on Pattern Recognition, ICPR, 2002, pp. 489–492.
[16] Y. Zhong, H. Zhang, A.K. Jain, Automatic caption localization in compressed video, IEEE Trans. Pattern Anal. Mach. Intell. 22 (4) (2000) 385–392.
[17] Z. Li, G. Liu, X. Qian, D. Guo, H. Jiang, Effective and efficient video text extraction using key text points, IET Image Process. 5 (8) (2009) 671–683.
[18] P. Shivakumara, T.Q. Phan, C.L. Tan, New wavelet and color features for text detection in video, in: Proceedings of 20th International Conference on Pattern Recognition, ICPR, 2010, pp. 3996–3999.
[19] X.-C. Yin, X. Yin, K. Huang, H.-W. Hao, Robust text detection in natural scene images, IEEE Trans. Pattern Anal. Mach. Intell. 36 (5) (2014) 970–983.
[20] Y. Li, W. Jia, C. Shen, A. van den Hengel, Characterness: an indicator of text in the wild, IEEE Trans. Image Process. 23 (4) (2014) 1666–1677.
[21] A. Zamberletti, L. Noce, I. Gallo, Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions, in: Proceedings of 1st International Workshop on Robust Reading, IWRR, 2014, pp. 1–15.
[22] E.K. Wong, M. Chen, A new robust algorithm for video text extraction, Pattern Recognit. 36 (6) (2003) 1397–1406.
[23] H.I. Koo, D.H. Kim, Scene text detection via connected component clustering and nontext filtering, IEEE Trans. Image Process. 22 (6) (2013) 2296–2305.
[24] X. Zhao, K.-H. Lin, Y. Fu, Y. Hu, Y. Liu, T.S. Huang, Text from corners: a novel approach to detect text and caption in videos, IEEE Trans. Image Process. 20 (3) (2011) 790–799.
[25] P. Shivakumara, T.Q. Phan, C.L. Tan, New Fourier-statistical features in rgb space for video text detection, IEEE Trans. Circuits Syst. Video Technol. 20 (11) (2010) 1520–1532.
[26] E.R. Dougherty, An Introduction to Morphological Image Processing, SPIE Optical Engineering Press: Bellingham, WA, 1992.
[27] S.M. Lucas, ICDAR 2005 text locating competition results, in: Proceedings of the 8th International Conference on Document Analysis and Recognition, ICDAR, 2005.
[28] A. Shahab, F. Shafait, A. Dengel, ICDAR 2011 robust reading competition challenge 2: reading text in scene images, in: Proceedings of the 11th International Conference on Document Analysis and Recognition, ICDAR, 2011, pp. 1491–1496.
[29] Y.-F. Pan, X. Hou, C.-L. Liu, A hybrid approach to detect and localize texts in natural scene images, IEEE Trans. Image Process. 20 (3) (2011) 800–813.
[30] D. Chen, J.-M. Odobeza, J.-P. Thiran, A localization/verification scheme for finding text in images and video frames based on contrast independent features and machine learning methods, Signal Process.: Image Commun. 19 (3) (2004) 205–217.

Yong Zhang received the M.Sc. degree in computer science and technology from Hefei University of Technology, Hefei, China. He is currently completing his Ph. D. degree in computer science from the School of Information Science and Technology, Sun Yat-Sen University, Guangzhou, China. His current research interests include image processing and computer graphics.

Jianhuang Lai received the M.Sc. degree in applied mathematics in 1989 and the Ph.D. degree in mathematics in 1999, both from Sun Yat-Sen University, Guangzhou, China. He is currently a Professor with the Department of Automation, and the Dean of the School of Information Science and Technology. He has published over 80 scientific papers in international journals and conferences on image processing and computer vision, e.g., IEEE TNN, IEEE TIP, IEEE TPAMI, ICCV and CVPR.



Pong C. Yuen received the B.S. degree in electronic engineering with first class honors from the City Polytechnic of Hong Kong, Kowloon, Hong Kong, in 1989, and the Ph.D. degree in electrical and electronic engineering from the University of Hong Kong, Pokfulam, Hong Kong, in 1993. He joined the Department of Computer Science, Hong Kong Baptist University, Kowloon, as an Assistant Professor in 1993, where he is currently a Professor.