Vis Comput (2014) 30:401–415
DOI 10.1007/s00371-013-0864-7

ORIGINAL ARTICLE

Transform invariant text extraction

Xin Zhang · Zhouchen Lin · Fuchun Sun · Yi Ma

Published online: 30 August 2013

Abstract Automatically extracting texts from natural images is very useful for many applications such as augmented reality. Most of the existing text detection systems require that the texts to be detected (and recognized) in an image are taken from a nearly frontal viewpoint. However, texts in most images taken naturally by a camera or a mobile phone can have a significant affine or perspective deformation, making the existing text detection and the subsequent OCR engines prone to failures. In this paper, based on stroke width transform and texture invariant low-rank transform, we propose a framework that can detect and rectify texts in arbitrary orientations in the image against complex backgrounds, so that the texts can be correctly recognized by common OCR engines. Extensive experiments show the advantage of our method when compared to the state of art text detection systems.

Keywords Text extraction · Arbitrary orientation · Stroke width transform · Texture invariant low-rank transform

1 Introduction

With the popularization of smart phones in recent years, there has been a tremendous increase in demand of intelligent, interactive applications with their onboard cameras. Applications such as "Google Goggles" and "Word Lens" have captured much of the imagination and expectation of the general public. For such applications, one crucial technological component is the ability to automatically detect, recognize and even translate texts (e.g. street signs) within a captured image. As many web images (e.g. street views) also contain texts, this technology would also enable us to more effectively annotate, index, and search web images.

In the vision community, it is generally believed that text detection and recognition have been an extensively studied problem and mature solutions must already exist. This could not have been far from the truth. As our survey will show later, although there is a vast literature on text detection and recognition, most of it was developed mainly for the classic scenario of digitizing documents or images with nearly frontal texts hence are far from applicable to uncontrolled scenarios mentioned above. For one, most existing text detection and recognition systems assume that the texts in the image are taken from a nearly frontal viewpoint. This is often far from the case for web images or images taken by mobile phones, see Fig. 1(a) for an example. The texts in the image can be severely distorted by an arbitrary planar (rotation, affine, or homography) transform from the image plane. Almost all existing text detection methods perform poorly on such images. Even if such texts are somehow detected (say manually marked out), the subsequent recognition would also fail because the texts are not in their upright position which is often needed by OCR engines for accurate recognition (see Tables 4 for a comparison). Things become even more challenging for conventional systems if there are multiple lines of texts in the same image. Most detection systems tend to fail to group the texts properly if the text lines are not nearly parallel and horizontal (see examples in Fig. 10). Without correct grouping, the detected texts cannot be correctly recognized either.

X. Zhang (✉) · F. Sun
Tsinghua University, Haidian District, Beijing, China
e-mail: xinzhang1111@gmail.com

Z. Lin
Peking University, Haidian District, Beijing, China

Y. Ma
Microsoft Research Asia, Beijing, China

Fig. 1 Extracting texts from images. (**a**) An input image and text detection results. (**b**) Text rectification results. (**c**) Text recognition results

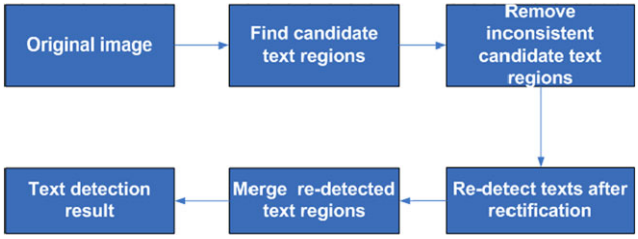


Fig. 2 Overall text detection process

In this paper, we address these challenges in text extraction from natural images, and our goal is to develop a novel system that can effectively and efficiently extract texts in an image that could have undergone significant planar transformations. We will handle similarity, affine, and homography transforms in a unified manner and they together could be used to model almost all practical deformations induced by projecting a text in the scene onto the image plane. The success of our system leverages strongly on two recent breakthroughs on text detection and text rectification, namely the *Stroke Width Transform* (SWT) [1] and the *Transform Invariant Low-rank Textures* (TILT) [2].

Stroke width recently has emerged as a new feature that has shown to be extremely effective for text detection [3–5] and it has quickly been extended to the so-called Stroke Width Transform (SWT) and leads to a state-of-the-art text detection system [1]. However, one limitation of the algorithms in [3–5] is that they can only detect horizontal texts. The algorithm proposed in [1] can detect near-horizontal (within $-15°$ and $15°$) and near-vertical (within $75°$ and $105°$) texts. It cannot detect texts with larger skews or deformations (e.g. Fig. 10).

Obviously, one natural thought to fix the limitations of SWT is to apply it only to text regions whose deformations have largely been corrected or rectified. In a recent work [2], it has been shown that texts belong to the class of so-called Transform Invariant Low-rank Textures (TILT). The TILT algorithm harnesses the fact that the rank of a rectified image is the lowest. By modifying the non-convex object function into a convex one and finding the minimum value of object function, TILT can robustly estimate the project transformed matrix and rectify the distorted image. Notice that the original TILT algorithm need users to annotate the distorted regions, it cannot automatically determine the regions.

So we cannot directly use the SWT and TILT algorithms. The main reason is that the TILT algorithm can only rectify already detected low-rank text regions and the SWT based detection is not effective unless the texts are already somewhat rectified. To resolve this dilemma and combine the strengths of both methods, we propose a simple yet effective scheme that integrates these two methods and can automatically extract almost all texts in an image despite their arbitrary initial deformations. Figure 2 shows the flowchart of the system and Fig. 1 shows a typical result from our system. We have conducted extensive experiments (and comparisons with existing systems) to demonstrate the effectiveness of our system. As we will see from the experimental results, the rectification cannot only improve the detection accuracy considerably (see Table 2), but also improves the overall text recognition rates significantly (see Table 4).

Contributions

1. We propose a simple but effective framework to detect and rectify text of arbitrary direction. By using both the local feature (stroke width transform feature) and global feature (low rank) of text image, this algorithm greatly increase the detection rate as well as recognition rate.
2. We propose an enhanced text rectification algorithm. Although the original rectification algorithm can work well for single distorted character, it does not work well for multiple characters, such as a word or short phrase. Harnessing the fact that those multiple characters share the same transform parameters, we propose an enhanced rectification algorithm.
3. We propose a new evaluation for a distorted text detection algorithm. Since the original text detection evaluation is mainly designed for horizontal aligned or vertical aligned text region, it is no longed suitable for distorted texts cases. Inspired by evaluation of the object detection algorithm, our evaluation is also based on the intersection and union-section of detection text rectangle and ground truth rectangle.

1.1 Relation to prior work

There has been extensive work on text detection from images or videos [6–11]. The algorithms can be roughly divided into two categories: texture-based and region-based. The main idea of texture-based methods is first dividing the image into small patches. For each patch, retrieve the distinct properties that can separate text regions from background ones. Such properties include distribution of wavelet coefficients [12, 13], DCT coefficients [14], edge feature [15], spatial variance [16]. Then use some classifier, such as support vector machine (SVM) [17], neural network [18], and Adaboost [19], to classify text patches and

non-text ones. Although texture-based methods are more robust to noise and have a better self-learning ability, they are all computationally expensive. What is more, they are unable to detect slanted texts effectively. Region-based methods first group the pixels which share the same properties, such as the edges' geometrical arrangement [20, 21] and color uniformity [12, 22]. Then for each group of pixels, some geometrical constraints and other text features are used to remove non-text groups. Region-based methods are often efficient in detecting texts of different scales and they can detect somewhat slanted texts. Although in this work we use the SWT-based method, one could easily replace it with any more effective detection methods in the future.

As for text rectification, there have been many techniques, e.g., [23], developed in the past to preprocess and rectify distorted text documents. Almost all these techniques rely on the global regular layout of the texts to rectify the distortion. That is, the rectified texts should lie on a set of horizontal, parallel lines, often in a rectangular region. Hence, many different methods have been developed to estimate the rotation or skew angle based on statistics of the distorted text compared to the standard layout, including methods based on projection profiles [24], Hough transform for gradient/edge directions [25], morphology of the text region [26], and cross-correlation of image blocks [27] etc. In our context, the TILT-based method stands out as it can be engineered to handle letters, words, and phrases in a unified manner. Figure 14 shows some input text regions while Fig. 15 shows the rectified results using TILT. These images are rather challenging for all the conventional rectification techniques.

2 Technical approach

Our text detection algorithm consists of four steps that interleave text detection and rectification. It first finds candidate text areas using Stroke Width Transform (SWT) [1]. Then it removes text areas whose principal directions are inconsistent with others'. Next, it re-detects texts after the regions of interest are rectified with the transforms found by TILT. Finally, it fuses the re-detected text areas and removes non-text areas. The flowchart of our algorithm is shown in Fig. 2.

2.1 Find candidate text regions

Texts are usually printed on different planar surfaces in the scene. If we could undo the transformation from each planar surface to the image plane, then all texts on the surface become rectified and conventional text detection methods should have a better chance of detecting the texts. So first we need to find some candidate text regions on these planar surfaces so that we can compute the rectifying transforms for those planes. Before we discuss our algorithm, we will introduce the Stroke Width Transform (SWT) model.

The SWT algorithm proposed by Epshtein et al. [1] lies in the text stroke width. The nearly constant stroke text width feature separates the text from other elements of a complex scene. The *stroke width map* is the key component the SWT model. In order to get the stroke width map, first we need to calculate the canny edges of the input image and set ∞ to all the pixels in stroke width map. Then a gradient direction d_p of each edge pixel p is calculated. Based on the fact that if p lies on a stroke boundary, d_p will be nearly perpendicular to the stroke's orientation. It follows the direction: $r = p + n \times d_p$ where $n > 0$ to find the next edge pixel q. If the direction of pixel p and q is approximately opposite, then all the pixels along this direction are set to the distance between p and q. If directions of pixel p and q are not opposite or q is not found, this direction will be removed. After this stroke width map is calculated, one uses some logical and flexible geometric rules to group strokes into words and finally detect the text regions in the image. The reasons we choose this method are as follows: firstly, this new image operator (stroke width map) is fast and robust which does not need scanning window techniques and multi-scale calculation. Secondly, the feature is not calculated for single pixel, it is a feature of grouping pixels, which greatly reduce the number of detected pixels. Thirdly, it can detect some text regions with slight distortion. Notice that since our paper proposes a framework for distorted text detection, if other algorithms can robustly detect some slight slant text regions, they can also be used here to replace the SWT method.

Through our experience, the SWT algorithm is very effective at detecting near-rectified texts, roughly tolerating a rotation between $-15°$ and $15°$. In order to find more candidate text regions in a given image, we apply the SWT algorithm to the image rotated by $-30°$, $-10°$, $+10°$, and $+30°$.[1] This has effectively enlarged the working range of SWT, practically covering almost all possible orientations of texts. After rotating the image, the SWT algorithm can detect texts that are roughly aligned with that orientation (or perpendicular to it). Figure 3 shows detected text regions of a representative image under the four orientations. This example also shows that this naive extension of SWT cannot achieve satisfactory detection results. So the purpose of this step is only to find some candidate text regions on different 3D-planes so that we can use them to estimate the associated transformation. The detection results in this step need not be so accurate.

[1] Although $-40°$, $-20°$, $0°$, and $+20°$ are more natural choices of rotation angles, as in natural scenes there are rarely texts that are exactly horizontal, from our experience these angles are not as effective for detecting texts in all directions.

Fig. 3 Find candidate text regions by first rotating the image and then applying SWT [1]. (**a**)–(**d**) are text detection results with the image being rotated at $-30°$, $-10°$, $+10°$, and $+30°$, respectively. (**e**) is the original input image for text detection. When rotated by $+10°$, no text is detected

Fig. 4 Remove inconsistent candidate text regions. (**a**) shows all candidate text regions detected in the previous step, where the *green regions* are the inconsistent ones. (**b**) is the remaining candidate text regions after removing inconsistent text regions

2.2 Remove inconsistent candidate text regions

The candidate text regions obtained above could have some overlap with each other as the same text could be detected multiple times at different orientations (see Fig. 4(a)). To better initialize the rectification algorithm and estimate the transform for a text region, it is important that we remove the regions that are detected at a wrong orientation. We define the principal direction of a candidate text region as the direction of its longer side. We use a Radon transform on the edge map of the text region to get the principal direction. For a group of overlapped candidate text regions, we first compute their joint convex hull and the principal direction of the convex hull. We remove text regions whose principal directions are more than 20° from that of the joint convex hull. These regions are often false positives from the SWT algorithm that correspond to regions that cross multiple lines of texts. With such regions removed, the transform of the text region can be more accurately estimated by TILT [2]. Moreover, the joint principal direction of the remaining regions provides TILT with a good initialization. Figure 4 shows an example of the process of filtering candidate text regions.

2.3 Re-detect texts after rectification

After removing the inconsistent candidate text regions, the next goal is to estimate a more accurate transformation between the text plane to the image plane. To this end, we utilize the TILT [2] algorithm and extend it for our purposes.

Before modifying TILT algorithm, we first introduce the original Transform Invariant Low-rank Textures (TILT) algorithm. TILT is to minimize the robust rank of an image patch D by deforming it with a geometric transform τ:

$$\min_{A,E,\tau} \; rank(A) + \lambda \|E\|_0 \quad \text{s.t.} \quad D \circ \tau = A + E, \tag{1}$$

where D is the matrix of distorted input image, $\circ$ is the homography transform operator, A is the low rank matrix and E is the sparse error term which is used to make the character image as an "approximately" low rank matrix by removing some imperfect parts. λ is the coefficient to balance the low rank term and sparse error term. In this paper, τ is a $\mathbb{R}^2 \to \mathbb{R}^2$ and belongs to a certain Lie group G. To be specific, we only consider G that is either a 2D affine group or homography group. Suppose the homography transform matrix is defined as

$$\begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix}$$

and τ encodes all the information of this matrix.

Then the homography transform can be written as follows:

$$D(x, y) \circ \tau = D\left(\frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + 1}, \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + 1} \right); \tag{2}$$

where (x, y) is the pixel location of the input image D.

As pointed out in [28], the problem in the mathematic model (1) is a general NP-hard problem because the rank function and ℓ_0 norm in the original TILT are extremely difficult to optimize. However, in recent breakthroughs in sparse representation and low-rank matrix recovery, the above problem can be replaced by its convex surrogates under fairly broad conditions. They can be replaced by their convex surrogates. The rank of A can be replaced by the matrix nuclear norm $\|A\|_*$ and the $\|E\|_0$ can be replaced by ℓ_1 norm $\|E\|_1$. Thus we get the following object function:

$$\min_{A,E,\tau} \; \|A\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad D \circ \tau = A + E, \tag{3}$$

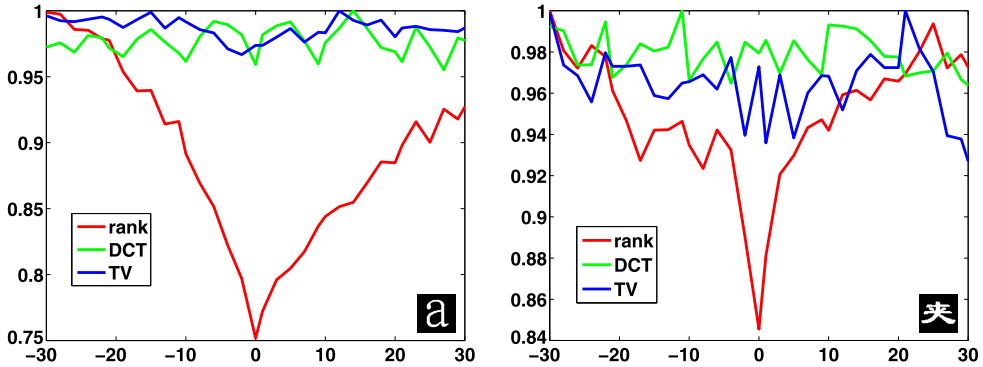


Fig. 5 Values of TILT, l_1 norm of Discrete Cosine Transform (DCT) coefficients and Total Variation (TV) objective functions (y-axis) of the input character (shown as the embedded icon) against the rotation angle (x-axis). TILT (the *red line*) achieves its minimal values when the character is unrotated, while DCT (the *green line*) and TV (the *blue line*) are much less robust and sensitive to the correct position

where $\|A\|_*$ is the nuclear norm of A, i.e., the sum of the singular values of A, and $\|E\|_1$ is the l_1 norm of E, i.e., the sum of the absolute values of the entries in E. Empirically, one can verify that the minimal value of the above objective function can be a good indicator of the upright position of English and Chinese characters and digits. Figure 5 shows the values of the objective function of TILT when the characters or digits are rotated. They all achieve their minimal value when the characters or digits are in their upright positions. This may be due to the rich structural regularity, such as local or global symmetry or parallelism, which can be accounted for by the low rankness of A measured by $\|A\|_*$. Any small deviation from the perfect structural regularity can be accounted for by the sparse error E measured by $\|E\|_1$.

Note that TILT is mainly effective for rectifying a single character, while our detected text regions usually consists of multiple characters, which can be a short phrase on a street sign or an entry on a restaurant menu. Although the images of individual characters may be low-rank, the image of multiple characters may no longer be low-rank anymore *w.r.t. its minimal dimension*. So the original TILT model (3) would not work on an image of multiple characters as robustly as on an individual character. To remedy this issue, one could divide the region into multiple smaller ones and measure the overall regularity as the sum of all:

$$\begin{aligned} &\min_{A_i,E,\tau} \sum_{i=1}^{n} \|A_i\|_* + \|A\|_* + \lambda\|E\|_1, \\ &\text{s.t.} \quad D \circ \tau = A + E, \quad A = [A_1, \ldots, A_n], \end{aligned} \tag{4}$$

where A_i stands for the ith block of A and τ is searched within transform group. The above formulation is motivated by the observation that each character (hence subimage) is of low rank. Although A_i should ideally correspond to a character, in practice it is not necessary to segment the region accurately. Only a rough estimate of the number of characters, which can be easily derived from the aspect ratio of the text region, is needed and A_i can be obtained by equally partitioning the region. The readers can also find it in Supplementary Materials. The readers may refer to [2, 29] for how to derive the detailed optimization algorithms.

When applying TILT to rectify a text region, we initialize the transform τ with the rectangle found in Sect. 2.2. Moreover, the image patch D is the binary text mask map provided by SWT, which does not contain too much background. In the mask map, the text area has value 1 and the background has value 0. Then the TILT algorithm will find the best (affine) transform that rectifies the text region to its upright position. Some examples of rectification are shown in Figs. 14 and 15.

Next, we group the obtained affine transforms from all the candidate text regions. We first group by rotations such that the maximum difference of rotation degrees is within 5 degrees. For each group, we further group the transforms by their skew values such that the maximum difference is within 0.1.

Finally, we apply the mean of the transforms in each group to the whole image and apply SWT to re-detect texts in the transformed image. Since our transform matrix is parameterized by some parameters, such as scale, rotation angle, transition in x and y axis, the mean of the transforms is actually the mean of the parameters. Using those mean parameters, we generate the corresponding transform matrix. Some text re-detection results after image rectification are shown in Fig. 6. Notice that all the rectified images are transformed by the original input image because we have already calculated the corresponding transform parameter in the original input image, see (a) in Fig. 4 for example. We can see that the detection accuracies are significantly improved. This is partially because SWT relies heavily on the constant stroke width. However, in the original (distorted) image, stroke width can change significantly due to affine

Fig. 6 Text re-detection results on images rectified by different affine transforms. *Red rectangles* indicate the text areas detected. The detection results are significantly better than those in Fig. 3

Fig. 7 The process of merging re-detected text regions. (**a**) is all the re-detected text regions, shown as *red windows*. (**b**) is the final detection results after merging

Fig. 8 Convex hulls of different groups of re-detected text areas. We first find groups of rectangles whose pairwise overlapping ratio is greater than 0.7. After removing inconsistent rectangles by principal directions, we calculate the convex hull of each rectangle group and segment the hulls out of the original image

or perspective transform. Hence, rectification is crucial for enhancing SWT-based text detection.

2.4 Merge re-detected text regions

The re-detected text regions from the above step are a set of rectangles which indicate refined candidate text regions (see Fig. 7(a)). Again, many rectangles can overlap on the same text region. So we have to identify a best joint region from these overlapping rectangles.

To this end, we first group the rectangles by checking their pairwise overlapping ratio, defined as the common area of two rectangles divided by the area of the smaller region. If the overlapping ratio is above 0.7, they are grouped together. Then for each group of rectangles, we calculate the principal direction of the union of their binary text mask maps by projection. Namely, the direction that accumulates most of the text pixels and has the most salient peaks and valleys if there are multiple lines of texts. Next, we remove the rectangles whose principal direction is inconsistent with that of the union. Finally, we calculate the convex hull of each group and extract the hulls out of the original image (see Fig. 8 for some of the examples).

After we get these text segments, we apply TILT (4) once more on the binary text mask map of each of the text regions, with the initial τ being the rectangular bounding box of the text region. This time, TILT (4) can produce a very accurate (affine) transform for rectifying each text region. After rectification, we resize them so that their heights are between 10 to 300 pixels, within the working range of SWT. In this way, when we re-apply SWT on each text segments, both small and large texts can be detected together. Figure 7(b) shows an example of the final results of text detection, where the affine transform for each planar text region is accurately undone.

3 Experiments

In this section, we present extensive experiments to verify the effectiveness of our system in extracting arbitrarily oriented texts in natural images, in comparison with many of the state-of-the-art systems and methods. In addition, we verify empirically how the results of our system could significantly impact on the performance of OCR engines.

3.1 Text detection

To compare the accuracy of our system with others in the literature, we use three databases. The first is the public database used for ICDAR 2003 and ICDAR 2005 competitions [30]. This database mainly consists of nearly frontal texts. Although this dataset is not ideal for testing the strength of our system, we can show that our system is on par with or better than other state of the art methods in this conventional setting. The second is an Oriented Scene Text Dataset [31] which contains lots of distorted text regions.

Table 1 Performances of different text detection methods on the public ICDAR database. Since most texts in this database are nearly frontal, our algorithm is only marginally better than Epshtein et al.'s SWT [1] method. The results of other methods are quoted from [1]

Algorithm	Precision	Recall	f-measure
Our system	**0.73**	**0.64**	**0.68**
Epshtein [1]	**0.73**	0.60	0.66
Yi [31]	0.71	0.62	0.66
Alex Chen [19]	0.60	0.60	0.58

The third is a set of typical natural images collected by ourselves that are rich of texts in all geometric orientations. We will see that our system can significantly improve text detection than other methods for such less controlled image sets.

3.1.1 On the standard ICDAR database

In this section, we compare the detection results of various methods on the public database for ICDAR 2003 and ICDAR 2005 competitions [30]. The ICDAR database contains 258 images in the training set and 251 images in the test set. The test set is used for evaluation. The images are full-color and vary in size from 307×93 to 1280×960 pixels. A commonly adopted way to evaluate algorithms is to use the f-measure, which is a combination of precision and recall. They are defined as follows [1]:

$$precision \doteq \sum_{d \in D} \frac{m(d, G)}{|D|}, \qquad recall \doteq \sum_{g \in G} \frac{m(D, g)}{|G|}, \tag{5}$$

$$f \doteq \frac{1}{\frac{\alpha}{precision} + \frac{1-\alpha}{recall}}, \tag{6}$$

where D and G are the sets of detection rectangles and ground truth rectangles, respectively. $m(d, G)$ is the best matching score of rectangle d with those in G, defined as the area of intersection divided by the area of the minimum bounding box containing both rectangles. $m(D, g)$ is defined similarly. $|D|$ and $|G|$ are the numbers of rectangles in D and G, respectively. A typical choice for α is 0.5.

In the above standard settings, the precisions, recalls, and f-measures of various methods are shown in Table 1. There is little surprise to see that our method is as good as Epshtein et al.'s SWT [1] as our method builds upon it. As the dataset contains primarily nearly frontal texts, additional rectification in our method does not seem to improve much of the performance, only improving the recall slightly.

3.1.2 Our dataset of images from the web and phones

To build a database that contains rich texts with natural orientations, we download some images from the Internet and also capture some images with a mobile phone by ourselves. We collect a total of 141 images, containing road signs, shop name boards, name cards, bill boards, and book covers. The image sizes vary from 237×194 to 1323×998 pixels. So images in our database is very diverse (see Fig. 10 for some representative examples). We label the ground truth of our dataset by clicking the four corners of a quadrilateral that encompasses an entire text line exactly.[2]

In the preceding section, the evaluation focuses on measuring how accurately the detected regions overlap with the ground truth regions. This measure may become too demanding for images with texts of arbitrary orientation. In practice, we often are more interested if all labeled regions have been successfully detected and we care less about their precise overlap percentage with the detected regions. Therefore, we here use an evaluation protocol that is similar to that in PASCAL object detection task [32]. Given a detected text area d and a ground truth text area g, we first calculate the ratio of their intersection area to their union area:

$$m(d, g) \doteq \frac{A(d \cap g)}{A(d \cup g)}. \tag{7}$$

If this overlapping ratio is above 50 %, the text area is considered as detected correctly. So we define the true positive score as

$$TP(d, g) \doteq \begin{cases} 1, & \text{if } m(d, g) > 0.5, \\ m(d, g), & \text{if } m(d, g) \le 0.5. \end{cases} \tag{8}$$

Then the new precision and recall are defined as

$$precision \doteq \sum_{d \in D, g \in G} \frac{TP(d, g)}{|D|},$$

$$recall \doteq \sum_{d \in D, g \in G} \frac{TP(d, g)}{|G|},$$

and the f-measure is still computed as (6) and α is still chosen as 0.5.

As Epshtein et al.'s SWT [1] method and Alex Chen's method have been proven to be better than other methods and Alex Chen's method and Yi's method claimed to be able to detect slant text regions, we compare our method with Epshtein et al.'s SWT, that of Alex Chen [19] and Yi's method.

Table 2 shows the results on our dataset. We can see that our method is significantly better than SWT and Alex Chen's on the dataset that contains arbitrarily oriented text regions. To help the reader to see the difference visually,

[2] Notice that this labeling is somewhat different from the ground truth labeling used in the ICDAR dataset in which each word is labeled separately. The dataset will be released for public evaluation if our paper is accepted for publication.

Table 2 Performances of different text detection methods on our own database. Since most texts in our database are slant, our algorithm significantly outperforms Epshtein et al.'s SWT [1] method, Yi [31] and Alex Chen's [19] in precision, recall, and f-measure

Algorithm	Precision	Recall	f-measure
Our system	**0.7956**	**0.8299**	**0.8124**
Epshtein [1]	0.5501	0.6164	0.5814
Yi [31]	0.3624	0.5686	0.4427
Alex Chen [19]	0.3658	0.3254	0.3444

Table 3 Performances of different text detection methods on OSTD database. Since most texts in OSTD are slant, our algorithm significantly outperforms Epshtein et al.'s SWT [1] method, Yi [31], Alex Chen's [19] in precision, recall, and f-measure

Algorithm	Precision	Recall	f-measure
Our system	**0.9519**	**0.7917**	**0.8644**
Epshtein [1]	0.4383	0.5392	0.4835
Yi [31]	0.4915	0.5412	0.5152
Alex Chen [19]	0.3910	0.3243	0.3545

some representatively detection results of our method and SWT are shown in Figs. 9 and 10. SWT is clearly limited to detect near horizontal or vertical texts, while our method can effectively detect arbitrarily oriented texts. Moreover, our method separates the text lines better than SWT does (see (d1), (d2) and (e1), (e2) in Fig. 10).

3.1.3 Oriented Scene Text Dataset

In paper [31], they propose a public dataset Oriented Scene Text Dataset (OSTD). It contains texts of arbitrary orientations. There are totally 89 images and the images varies from street view to indoor objects. The quantitative results are shown in Table 3. Since the image in OSTD is do not contains many background, that is there always contains only text objects, so the precision of our algorithm is rather high. However, since there are multiple lines of text, our algorithm sometimes failed to detect all of them, so the recall rate is not as high as the precision rate. Still our algorithm outperforms the SWT, and Yi's and Alex Chen's method.

3.2 Text recognition

In practice, the goal to detect texts in images is to recognize them. In this section, we compare the OCR recognition rates on the detected texts from our system and others. Notice that one good property of texts detected from our system is that they are already rectified and hence should be much more suitable for recognition by conventional OCR engines. Here we use two OCR engine, one is widely used ABBYY Fine Reader [33] and the other is open source OCR engine Tesseract-ocr [34] which was developed by HP Labs.

Since we now are only interested in how rectification helps recognition, we use only original color detected text regions by our method or by others for evaluation. Ground truth is labeled for these regions. The precision and recall to measure the recognition rates are defined as

$$precision \doteq 1 - \frac{\sum_i dist(d_i, g_i)}{\sum_i length(d_i)},$$

$$recall \doteq 1 - \frac{\sum_i dist(d_i, g_i)}{\sum_i length(g_i)},$$

where $dist(d, g)$ is the edit distance [35] between the strings of the detected and ground truth texts. The f-measure is still defined as (6), and the parameter α is still set at 0.5.

In order to show the importance of text rectification for OCR, we calculate the precision, recall and f-measure on five datasets of two OCR engines. The first dataset is the text areas detected by SWT [1] on our database described in Sect. 3.1.2. We call it "Epshtein dataset". The second dataset is the text areas detected by Alex Chen's method [19] on our database. We call it "Alex Chen dataset". The third is the text areas detected by Yi's method [31] method. Both the fourth and fifth datasets are text regions detected on our database by our algorithm. The only difference is that the "unrectified dataset" are the text regions in the original image which are not rectified; and the "rectified dataset" are the same regions but rectified. There are 6689 characters in "Epshtein dataset" (see Fig. 11), 7283 characters in "Yi dataset" (see Fig. 13), 7335 characters in "Alex Chen dataset", and "unrectified dataset" and "rectified dataset" both have 7566 characters.

From Tables 4 and 5 we can see that the recognition rates on text regions extracted by SWT, Yi's method and Alex Chen's method are not satisfactory. Although SWT, Yi's method and Alex Chen's method basically detect nearly horizontal texts, there are still small rotation, skew and perspective in the detected texts left unrectified (see Fig. 12). So the performance of the OCR engine is still greatly affected. On the unrectified dataset detected by our method (see Fig. 14), as expected the OCR engine performs very poorly. After rectification, the texts are much closer to their frontal upright positions (see Fig. 15) and the recognition rates are improved dramatically.

Although our system works well to detect text regions with arbitrary distortions, there are still some limitations of our algorithm. The first limitation is we can only handle arbitrary distortions in planar, however, we cannot handle variety distortions in 3D dimension. This is because different shapes in 3D dimension can be parameterized in different ways. There is no uniform formula for those shapes, such as the surface of the cylinder cup, on a warped paper or surface of the ball. If all the 3D shapes belong to a same

(a1) (b1) (c1)

(a2) (b2) (c2)

(a3) (b3) (c3)

(d1) (e1) (f1)

(d2) (e2) (f2)

(d3) (e3) (f3)

Fig. 9 Part of the text detection results on our database. The (*1) is by SWT [1]. The (*2) is by the Yi method. The (*3) is by our method. The detected text areas are marked by *red rectangles*

Fig. 10 Part of the text detection results on OSTD database. The (*1) is by SWT [1]. The (*2) is by Yi method. The (*3) is by our method. The detected text areas are marked by *red rectangles*

Fig. 11 Epshtein dataset. This dataset consists of text areas detected by SWT [1]. SWT mainly detects texts that are nearly horizontal or with slight distortion

group, such as all cylinder shape, our algorithm can still work. If there are variation shape in 3D, we need to determine which model should we use. This can also be the future works.

The second limitation is that we only propose a framework for slant text detection. In our framework, we use the SWT for our text detection part. However, other techniques which can robustly detect slight distort text can also be used to replace the SWT algorithm. Since we only propose a framework for slant text detection, we did not compare the detection rate with other state-of-the-art text detection algorithm.

The other limitations includes how to robustly group the text regions, how to handle the shadow over text areas, glossy text region and how to handle blurred or noisy text region.

4 Conclusions and future works

In this paper, we propose a system that can automatically and effectively detect text regions in natural images that may have very diverse orientations and backgrounds. Our method

Table 4 ABBYY OCR recognition rates on different datasets. "Epshtein dataset", "Yi dataset" and "Alex Chen dataset" consist of text regions detected by the SWT method [1], Yi's method [31], Alex Chen's method [19] on our database. The recognition rates on these two datasets are not so miserable because most of the tests are nearly horizontal (see Fig. 12). "unrectified dataset" and "rectified dataset" are the text regions detected by our method. "unrectified dataset" are not rectified (see Fig. 14) and the recognition rates are very low. "rectified dataset" is rectified (see Fig. 15) and the recognition rates are dramatically better. All datasets are color images

Datasets	Precision	Recall	f-measure
Epshtein dataset [1]	0.5708	0.4793	0.5211
Yi dataset [31]	0.4306	0.3945	0.4118
Alex Chen dataset [19]	0.5819	0.4897	0.5318
Unrectified dataset	0.2175	0.2055	0.2113
Rectified dataset	0.7661	0.7162	0.7403

Table 5 Tesseract OCR recognition rates on different datasets. "Epshtein dataset", "Yi dataset" and "Alex Chen dataset" consist of text regions detected by the SWT method [1], Yi's method [31], Alex Chen's method [19] on our database. All datasets are color images

Datasets	Precision	Recall	f-measure
Epshtein dataset [1]	0.5610	0.3990	0.4663
Yi dataset [31]	0.3952	0.3878	0.3915
Alex Chen dataset [19]	0.6039	0.4316	0.5034
Unrectified dataset	0.1509	0.1793	0.1639
Rectified dataset	0.6228	0.5861	0.6039

Fig. 12 Alex Chen dataset. This dataset consists of text areas detected by Alex Chen's [19] method

leverages on two state-of-the-art text detection and text rectification techniques, namely SWT and TILT, and show how they could be combined to remedy each other's limitations and lead to a much more powerful text detection system. Our system can handle texts lying on multiple planar surfaces in a scene and can correctly segment multiple lines of texts on the same plane. Our system dramatically outperform existing text detection systems on datasets that contain practical, uncontrolled images from the Internet or taken by mobile phones. In addition, the detected texts from our system are automatically rectified and, as result, the performance of

Fig. 13 Yi dataset. This dataset consists of text areas detected by method [1]. This method mainly detects horizontal texts and can also detect slight distorted text regions

Fig. 14 Unrectified dataset. This dataset consists of texts detected by our method in the original images. Most of them are not in upright position

Fig. 15 Rectified dataset. This dataset consists of the corresponding rectified text outputs from our method. Almost all are correctly rectified to their upright position

OCR is improved significantly when using outputs from our text detection system. As both SWT and TILT are insensitive to the language, we expect that our method should also work reasonably well for extracting texts of other languages. This will be our future work.

Algorithm 1 (General Augmented Lagrange Multiplier Method)

1: Initialize variables.
2: **while** not converged **do**
3: Solve $x_{k+1} = \arg\min_x L(x, y_k, \mu_k)$.
4: $y_{k+1} = y_k + \mu_k h(x_{k+1})$;
5: Update μ_k to μ_{k+1}.
6: **end while**

Output: x_k.

Algorithm 2 (General Alternating Direction Method)

1: Initialize variables.
2: **while** not converged **do**
3: Solve $a_{k+1} = \arg\min_a L(a, b_k, y_k, \mu_k)$.
4: Solve $b_{k+1} = \arg\min_b L(a_{k+1}, b, y_k, \mu_k)$.
5: $y_{k+1} = y_k + \mu_k h(x_{k+1})$, where $x_{k+1} = (a_{k+1}^T, b_{k+1}^T)^T$;
6: Update μ_k to μ_{k+1}.
7: **end while**

Output: x_k.

Acknowledgements X. Zhang and F. Sun are supported by the National Natural Science Foundation of China (NNSFC) under Grant No. 2013CB329403. Z. Lin is supported by the NNSFC under Grant Nos. 61272341, 61231002, and 61121002. Y. Ma is partially supported by the funding of ONR N00014-09-1-0230, NSF CCF 09-64215, NSF IIS 11-16012.

Appendix: Detailed derivation of the ADM for multi-component TILT

A.1 ALM and ADM methods

The alternating direction method (ADM, also called inexact augmented Lagrange multiplier (IALM) method in [29]) is a variant of the augmented Lagrange multiplier (ALM) method. The usual ALM method for solving the following model problem:

$$\min f(x), \quad \text{s.t.} \quad h(x) = 0, \tag{9}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$, is to minimize over its augmented Lagrangian function:

$$L(x, y, \mu) = f(x) + \langle y, h(x)\rangle + \frac{\mu}{2}\|h(x)\|_F^2, \tag{10}$$

where μ is a positive scalar and $y \in \mathbb{R}^m$ is the Lagrange multiplier. The ALM method is outlined as Algorithm 1 (see [36] for more details).

Solving the subproblem

$$\min_x L(x, y_k, \mu_k)$$

is actually not always easy. When $f(x)$ is separable, e.g., $f(x) = f_1(a) + f_2(b)$, where $x = (a^T, b^T)^T$, we may solve this subproblem by minimizing over a and b alternately until convergence. However, in the case that $f(x)$ is convex and $h(x)$ is a linear function of x, the ADM requires updating a and b only once and it can still be proven that the iteration converges to the optimal solution [37]. The ADM for two variables is outlined as Algorithm 2. The generalization to the multi-variable and multi-constraint case is straightforward.

A.2 Subproblems of multi-component TILT

When applying ADM to the linearized multi-component TILT, whose augmented Lagrangian function is

$$\begin{aligned} &L\big(\{A_i\}, A, E, \Delta\tau, \{Y_i\}, Y, \mu\big) \\ &\quad = \sum_{i=1}^n \|A_i\|_* + \|A\|_* + \lambda\|E\|_1 \\ &\qquad + \big\langle Y, D\circ\tau + J\Delta\tau - A - E\big\rangle + \sum_{i=1}^n \big\langle Y_i, A_i - \mathcal{P}_i(A)\big\rangle \\ &\qquad + \frac{\mu}{2}\Bigg(\|D\circ\tau + J\Delta\tau - A - E\|_F^2 \\ &\qquad + \sum_{i=1}^n \big\|A_i - \mathcal{P}_i(A)\big\|_F^2\Bigg), \end{aligned}$$

where $\mathcal{P}_i(A)$ extracts the ith block of A, we have to solve multiple subproblems:

$$\begin{aligned} A_i^{(k+1)} = \arg\min_{A_i} L\big(&A_1^{(k+1)}, \ldots, A_{i-1}^{(k+1)}, A_i, \\ &A_{i+1}^{(k)}, \ldots, A_n^{(k)}, A^{(k)}, E^{(k)}, \Delta\tau^{(k)}, \\ &\big\{Y_j^{(k)}\big\}, Y^{(k)}, \mu^{(k)}\big), \quad i = 1, \ldots, n, \end{aligned} \tag{11}$$

$$\begin{aligned} A^{(k+1)} = \arg\min_{A} L\big(&\big\{A_j^{(k+1)}\big\}, A, E^{(k)}, \Delta\tau^{(k)}, \\ &\big\{Y_j^{(k)}\big\}, Y^{(k)}, \mu^{(k)}\big), \end{aligned} \tag{12}$$

$$\begin{aligned} E^{(k+1)} = \arg\min_{E} L\big(&\big\{A_j^{(k+1)}\big\}, A^{(k+1)}, E, \Delta\tau^{(k)}, \\ &\big\{Y_j^{(k)}\big\}, Y^{(k)}, \mu^{(k)}\big), \end{aligned} \tag{13}$$

$$\begin{aligned} \Delta\tau^{(k+1)} = \arg\min_{E} L\big(&\big\{A_j^{(k+1)}\big\}, A^{(k+1)}, E^{(k+1)}, \\ &\Delta\tau, \big\{Y_j^{(k)}\big\}, Y^{(k)}, \mu^{(k)}\big). \end{aligned} \tag{14}$$

When solving (11), by leaving out the terms in L that are independent of A_i, we see

$$\begin{aligned} A_i^{(k+1)} &= \arg\min_{A_i} \|A_i\|_* + \langle Y_i^{(k)}, A_i - \mathcal{P}_i(A^{(k)})\rangle \\ &\quad + \frac{\mu^{(k)}}{2}\|A_i - \mathcal{P}_i(A^{(k)})\|_F^2 \\ &= \arg\min_{A_i} \|A_i\|_* + \frac{\mu^{(k)}}{2}\|A_i - \hat{A}_i^{(k+1)}\|_F^2, \end{aligned}$$

where

$$\hat{A}_i^{(k+1)} = \mathcal{P}_i(A^{(k)}) - (\mu^{(k)})^{-1} Y_i^{(k)}.$$

Then by Theorem 2.1 of [38], $A_i^{(k+1)}$ can be obtained by thresholding the singular values of $\hat{A}_i^{(k+1)}$. Namely, if the singular value decomposition (SVD) of $\hat{A}_i^{(k+1)}$ is $U_{ik}\Sigma_{ik}V_{ik}^T$, then

$$A_i^{(k+1)} = U_{ik}\mathcal{S}_{(\mu^{(k)})^{-1}}(\Sigma_{ik})V_{ik}^T,$$

where

$$\mathcal{S}_\varepsilon(x) = \max(|x| - \varepsilon, 0)\,\mathrm{sgn}(x)$$

is the shrinkage operator.

Similarly, when solving (12) with some elaboration we have

$$\begin{aligned} A^{(k+1)} &= \arg\min_{A} \|A\|_* \\ &\quad + \langle Y^{(k)}, D\circ\tau + J\Delta\tau^{(k)} - A - E^{(k)}\rangle \\ &\quad + \sum_{i=1}^n \langle Y_i^{(k)}, A_i^{(k+1)} - \mathcal{P}_i(A)\rangle \\ &\quad + \frac{\mu^{(k)}}{2}\left(\|D\circ\tau + J\Delta\tau^{(k)} - A - E^{(k)}\|_F^2\right. \\ &\quad \left. + \sum_{i=1}^n \|A_i^{(k+1)} - \mathcal{P}_i(A)\|_F^2\right) \\ &= \arg\min_{A} \|A\|_* + \mu^{(k)}\|A - \hat{A}^{(k+1)}\|_F^2, \end{aligned}$$

where

$$\begin{aligned} \hat{A}^{(k+1)} &= \frac{1}{2}\big(D\circ\tau + J\Delta\tau^{(k)} - E^{(k)} + (\mu^{(k)})^{-1}Y^{(k)} \\ &\quad + \tilde{A}^{(k+1)}\big), \end{aligned}$$

in which $\tilde{A}^{(k+1)}$ is a matrix whose ith block is $A_i^{(k+1)} + (\mu^{(k)})^{-1}Y_i^{(k)}$. Then again we can apply Theorem 2.1 of [38] to solve for $A^{(k+1)}$ as we did above for $A_i^{(k+1)}$.

Algorithm 3 (Solving Multi-Component TILT by ADM)

INPUT: Initial rectangular window $D \in \mathbb{R}^{m\times n}$ and initial transform τ in a group $\mathbb{G}$ (affine or projective).

WHILE not converged **DO**

Step 1: Normalize the image and compute the Jacobian w.r.t. the transform:

$$D\circ\tau \leftarrow \frac{D\circ\tau}{\|D\circ\tau\|_F}, \quad J \leftarrow \frac{\partial}{\partial\zeta}\left(\frac{D\circ\tau}{\|D\circ\tau\|_F}\right)\Big|_{\zeta=\tau}.$$

Step 2: Solve the linearized multi-component TILT with the initial conditions:

$A^{(0)} = 0,\ E^{(0)} = 0,\ \Delta\tau^{(0)} = 0,\ Y_i^{(0)} = 0,\ Y^{(0)} = 0,$ $\mu_0 > 0,\ \rho > 1,\ k = 0.$

WHILE not converged **DO**

$(U_{ik}, \Sigma_{ik}, V_{ik}) \leftarrow \mathrm{svd}(\mathcal{P}_i(A^{(k)}) - (\mu^{(k)})^{-1}Y_i^{(k)}),$

$A_i^{(k+1)} \leftarrow U_{ik}\mathcal{S}_{(\mu^{(k)})^{-1}}(\Sigma_{ik})V_{ik}^T, \quad i = 1,\ldots,n,$

$(U_k, \Sigma_k, V_k) \leftarrow \mathrm{svd}(\frac{1}{2}(D\circ\tau + J\cdot\Delta\tau^{(k)} - E^{(k)} + (\mu^{(k)})^{-1}Y^{(k)} + \tilde{A}^{(k+1)})),$

$A^{(k+1)} \leftarrow U_k\mathcal{S}_{(2\mu^{(k)})^{-1}}(\Sigma_k)V_k^T,$

$E^{(k+1)} \leftarrow \mathcal{S}_{\lambda(\mu^{(k)})^{-1}}(D\circ\tau + J\cdot\Delta\tau^{(k)} - A^{(k+1)} + (\mu^{(k)})^{-1}Y^{(k)}),$

$\Delta\tau^{(k+1)} \leftarrow J^\dagger(-D\circ\tau + A^{(k+1)} + E^{(k+1)} - (\mu^{(k)})^{-1}Y^{(k)}),$

$Y_i^{(k+1)} \leftarrow Y_i^{(k)} + \mu^{(k)}(A_i^{(k+1)} - \mathcal{P}_i(A^{(k+1)})),$

$Y^{(k+1)} \leftarrow Y^{(k)} + \mu^{(k)}(D\circ\tau + J\Delta\tau^{(k+1)} - A^{(k+1)} - E^{(k+1)}),$

$\mu^{(k+1)} \leftarrow \min(\rho\mu^{(k)}, \mu_{\max}),$

$k \leftarrow k+1.$

END WHILE

Step 3: Update transform:

$\tau \leftarrow \tau + \Delta\tau^{(k+1)}.$

ENDWHILE

OUTPUT: A, E, τ

When solving (13), we have

$$\begin{aligned} E^{(k+1)} &= \arg\min_{E} \lambda\|E\|_1 \\ &\quad + \langle Y^{(k)}, D\circ\tau + J\Delta\tau^{(k)} - A^{(k+1)} - E\rangle \\ &\quad + \frac{\mu^{(k)}}{2}\|D\circ\tau + J\Delta\tau^{(k)} - A^{(k+1)} - E\|_F^2 \\ &= \lambda\|E\|_1 + \frac{\mu^{(k)}}{2}\|E - \hat{E}^{(k+1)}\|_F^2, \end{aligned}$$

where

$$\hat{E}^{(k+1)} = D \circ \tau + J\Delta\tau^{(k)} - A^{(k+1)} + \left(\mu^{(k)}\right)^{-1} Y^{(k)}.$$

Then its well known [39] that the solution to the above problem is

$$E^{(k+1)} = \mathcal{S}_{\lambda(\mu^{(k)})^{-1}}\left(\hat{E}^{(k+1)}\right),$$

where $\mathcal{S}$ is the shrinkage operator defined above.

When solving (14), we have

$$\Delta\tau^{(k+1)} = \arg\min_{\Delta\tau} \left\| J\Delta\tau - \widehat{\Delta\tau}^{(k+1)} \right\|_F^2, \tag{15}$$

where

$$\widehat{\Delta\tau}^{(k+1)} = -D \circ \tau + A^{(k+1)} + E^{(k+1)} - \left(\mu^{(k)}\right)^{-1} Y^{(k)}.$$

So

$$\Delta\tau^{(k+1)} = J^{\dagger}\widehat{\Delta\tau}^{(k+1)},$$

where $J^{\dagger}$ is the pseudo-inverse of J and $\widehat{\Delta\tau}^{(k+1)}$ is rearranged into a vector.

A.3 The complete algorithm

Now we can present the ADM for multi-component TILT as Algorithm 3.

Note that in Algorithm 3, the transformed image $D \circ \tau$ has to be normalized to prevent the region of interest from shrinking into a black pixel [2]. For more details of implementation issues, such as maintaining the area and aspect ratio, please refer to [2]. We will also post our code online if the paper is accepted.

References

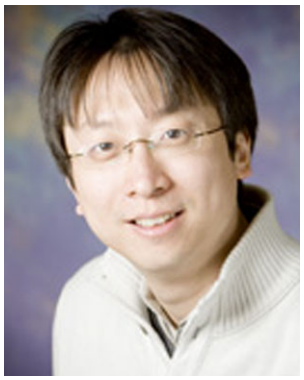
1. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: CVPR, pp. 2963–2970 (2010)
2. Zhang, Z., Ganesh, A., Liang, X., Ma, Y.: TILT: transform invariant low-rank textures. In: ACCV (2010)
3. Dinh, V.C., Chun, S.S., Cha, S., Ryu, H., Sull, S.: An efficient method for text detection in video based on stroke width similarity. In: ACCV, vol. 1, pp. 200–209 (2007)
4. Subramanian, K., Natarajan, P., Decerbo, M., Castañón, D.: Character-stroke detection for text-localization and extraction. In: ICDAR, pp. 33–37 (2007)
5. Jung, C., Liu, Q., Kim, J.: A stroke filter and its application to text localization. Phys. Rev. Lett. **30**(2), 114–122 (2009)
6. Jung, K., Kim, K.I., Jain, A.K.: Text information extraction in images and video: a survey. Pattern Recogn. **37**(5), 977–997 (2004)
7. Liang, J., Doermann, D.S., Li, H.: Camera-based analysis of text and documents: a survey. Int. J. Doc. Anal. Recognit. **7**(2–3), 84–104 (2005)
8. Daniilidis, K., Maragos, P., Paragios, N. (eds.): Proceedings of Computer vision—ECCV 2010, 11th European conference on computer vision, Part I, Heraklion, Crete, Greece, September 5–11, 2010. Lecture Notes in Computer Science, vol. 6311. Springer, Berlin (2010)
9. Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D.J., Ng, A.Y.: Text detection and character recognition in scene images with unsupervised feature learning. In: ICDAR, pp. 440–445 (2011)
10. Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: ACCV, vol. 3, pp. 770–783 (2010)
11. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: ICCV, pp. 1457–1464 (2011)
12. Shivakumara, P., Phan, T.Q., Tan, C.L.: New wavelet and color features for text detection in video. In: ICPR, pp. 3996–3999 (2010)
13. Ye, Q., Huang, Q., Gao, W., Zhao, D.: Fast and robust text detection in images and video frames. Image Vis. Comput. **23**(6), 565–576 (2005)
14. Zhong, Y., Zhang, H., Jain, A.K.: Automatic caption localization in compressed video. IEEE TPAMI **22**(4), 385–392 (2000)
15. Shivakumara, P., Huang, W., Tan, C.L.: Efficient video text detection using edge features. In: ICPR, pp. 1–4 (2008)
16. Wu, V., Manmatha, R., Riseman, E.M.: Textfinder: an automatic system to detect and recognize text in images. IEEE TPAMI **21**(11), 1224–1229 (1999)
17. Kim, K.I., Jung, K., Kim, J.H.: Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. IEEE TPAMI **25**(12), 1631–1639 (2003)
18. Li, H., Doermann, D.S., Kia, O.E.: Automatic text detection and tracking in digital video. IEEE TIP **9**(1), 147–156 (2000)
19. Chen, X., Yuille, A.L.: Detecting and reading text in natural scenes. In: CVPR, vol. 2, pp. 366–373 (2004)
20. Zhang, J., Kasturi, R.: Text detection using edge gradient and graph spectrum. In: ICPR, pp. 3979–3982 (2010)
21. Shivakumara, P., Phan, T.Q., Tan, C.L.: Video text detection based on filters and edge features. In: ICME, pp. 514–517 (2009)
22. Yi, J., Peng, Y., Xiao, J.: Color-based clustering for text detection and extraction in image. In: ACM Multimedia, pp. 847–850 (2007)
23. Liang, J., DeMenthon, D., Doermann, D.: Geometric rectification of camera-captured document images. IEEE TPAMI **30**(4), 591–605 (2008)
24. Pastor, M., Toselli, A., Vidal, E.: Projection profile based algorithm for slant removal. In: Image Analysis and Understanding. Lecture Notes in Computer Science, vol. 3212, pp. 183–190 (2004)
25. Singha, C., Bhatiab, N., Kaur, A.: Hough transform based fast skew detection and accurate skew correction methods. Pattern Recogn. **14**(12), 3528–3546 (2008)
26. Yuan, B., Tan, C.L.: Convex hull based skew estimation. Pattern Recogn. **40**(2), 456–475 (2007)
27. Yan, H.: Skew correction of document images using interline cross-correlation. CVGIP, Graph. Models Image Process., **55**(6), 538–543 (1993)
28. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: Rasl: robust alignment by sparse and low-rank decomposition for linearly correlated images. In: CVPR, pp. 763–770 (2010)
29. Lin, Z., Chen, M., Wu, L., Ma, Y.: The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. 2009, uIUC Technical Report UILU-ENG-09-2215. arXiv:1009.5055
30. Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: ICDAR 2003 robust reading competitions. In: ICDAR, pp. 682–687 (2003)

31. Yi, C., Tian, Y.: Text string detection from natural scenes by structure-based partition and grouping. IEEE Trans. Image Process. **20**(9), 2594–2605 (2011)
32. Everingham, M., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (voc) challenge 2006 (voc 2006) results
33. ABBYY Corp.: Abbyy finereader
34. HP Labs.: "Tesseract-ocr," 1985–1995. Now available at: http://code.google.com/p/tesseract-ocr/
35. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
36. Bertsekas, D.: Nonlinear Programming. Athena Scientific, Belmont (1999)
37. He, B., Yang, H.: Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities. Oper. Res. Lett. **23**, 151–161 (1998)
38. Cai, J., Candès, E., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM J. Optim. **20**(4), 1956–1982 (2010)
39. Yin, W., Hale, E., Zhang, Y.: Fixed-point continuation for ℓ_1-minimization: methodology and convergence. SIAM J. Optim. **19**(3), 1107–1130 (2008)

Xin Zhang was born in ShaanXi, China. She received the bachelor degree in 2005. She is now a Ph.D. student in department of computer Science and technology in Tsinghua University, China. Currently she is working on computer vision, image processing, and pattern recognition.

Zhouchen Lin received the Ph.D. degree in applied mathematics from Peking University, China, in 2000. He was a lead researcher in the Visual Computing Group, Microsoft Research, Asia. Now he has moved to Peking University. His research interests include computer vision, computer graphics, pattern recognition, statistical learning, document processing, and human computer interaction.

Fuchun Sun was born in Jiangsu Province, China, in 1964. From 1998 to 2000, he was a Postdoctoral Fellow with the Department of Automation, Tsinghua University, where he is currently a Professor with the Department of Computer Science and Technology. His research interests include intelligent control, networked control system and management, neural networks, fuzzy systems, nonlinear systems, and robotics. He has authored or coauthored two books and over 100 papers which have appeared in various journals and conference proceedings. Dr. Sun was the recipient of the Excellent Doctoral Dissertation Prize of China in 2000 and the Choon-Gang Academic Award by Korea in 2003, and was recognized as a Distinguished Young Scholar in 2006 by the National Science Foundation of China. He has been a member of the Technical Committee on Intelligent Control of the IEEE Control Systems Society since 2006.



Yi Ma received bachelor degrees in automation and applied mathematics from Tsinghua University, Beijing, in 1995. He received the MS degree in electrical engineering and computer science (EECS) in 1997, the MA degree in mathematics in 2000, and the Ph.D. degree in EECS in 2000, all from the University of California, Berkeley. Since 2000, he has been with the faculty of the Electrical and Computer Engineering Department, University of Illinois, Urbana-Champaign, where he now holds the rank of associate professor. His main research interests include systems theory and computer vision. He is a senior member of the IEEE, the IEEE Computer Society, and a member of the ACM. He was the recipient of the David Marr Best Paper Prize at the International Conference on Computer Vision in 1999 and Honorable Mention for the Longuet–Higgins Best Paper Award at the European Conference on Computer Vision in 2004. He received the CAREER Award from the US National Science Foundation in 2004 and the Young Investigator Program Award from the US Office of Naval Research in 2005.