Scene Text Recognition in Mobile Applications by Character Descriptor and Structure Configuration

Chucai Yi, *Student Member, IEEE*, and Yingli Tian, *Senior Member, IEEE*

***Abstract*— Text characters and strings in natural scene can provide valuable information for many applications. Extracting text directly from natural scene images or videos is a challenging task because of diverse text patterns and variant background interferences. This paper proposes a method of scene text recognition from detected text regions. In text detection, our previously proposed algorithms are applied to obtain text regions from scene image. First, we design a discriminative character descriptor by combining several state-of-the-art feature detectors and descriptors. Second, we model character structure at each character class by designing stroke configuration maps. Our algorithm design is compatible with the application of scene text extraction in smart mobile devices. An Android-based demo system is developed to show the effectiveness of our proposed method on scene text information extraction from nearby objects. The demo system also provides us some insight into algorithm design and performance improvement of scene text extraction. The evaluation results on benchmark datasets demonstrate that our proposed scheme of text recognition is comparable with the best existing methods.**

***Index Terms*—Scene text detection; Scene text recognition; Mobile application; Character descriptor; Stroke configuration; Text understanding; Text retrieval; Mobile application;**

I. Introduction

Camera-based text information serves as effective tags or clues for many mobile applications associated with media analysis, content retrieval, scene understanding, and assistant navigation. In natural scene images and videos, text characters and strings usually appear in nearby sign boards and hand-held objects and provide significant knowledge of surrounding environment and objects. Text-based tags are much more applicable than barcode or quick response code [12, 18], because the latter techniques contain limited information and require pre-installed marks.

To extract text information by mobile devices from natural scene, automatic and efficient scene text detection and recognition algorithms are essential. However, extracting scene text is a challenging task due to two main factors: 1) cluttered backgrounds with noise and non-text outliers, and 2) diverse text patterns such as character types, fonts, and sizes. The frequency of occurrence of text in natural scene is very low, and a limited number of text characters are embedded into complex non-text background outliers. Background textures, such as grid, window, and brick, even resemble text characters and strings. Although these challenging factors exist in face and car, many state-of-the-art algorithms [5, 24] have demonstrated effectiveness on those applications, because face and car, have relatively stable features. For example, a frontal face normally contains a mouth, a nose, two eyes, and two brows as prior knowledge. However, it is difficult to model the structure of text characters in scene images due to the lack of discriminative pixel-level appearance and structure features from non-text background outliers. Further, text consists of different words where each word may contain different characters in various fonts, styles, and sizes, resulting in large intra-variations of text patterns. To solve these challenging problems, scene text extraction is divided into two processes [32]: text detection and text recognition. Text detection is to localize image regions containing text characters and strings. It aims to remove most non-text background outliers. Text recognition is to transform pixel-based text into readable code. It aims to accurately distinguish different text characters and properly compose text words. This paper will focus on text recognition method. It involves 62 identity categories of text characters, including 10 digits [0-9] and 26 English letters in upper case [A-Z] and lower case [a-z].

We propose effective algorithms of text recognition from detected text regions in scene image. In scene text detection process, we apply the methods presented in our previous work [29]. Pixel-based layout analysis is adopted to extract text regions and segment text characters in images, based on color uniformity and horizontal alignment of text characters. In text recognition process, we design two schemes of scene text recognition. The first one is training a character recognizer to predict the category of a character in an image patch. The second one is training a binary character classifier for each character category to predict the existence of this category in an image patch. The two schemes are compatible with two promising applications related to scene text, which are text understanding and text retrieval. Text understanding is to acquire text information from natural scene to understand surrounding environment and objects. Text retrieval is to verify whether a piece of text information exists in natural scene.

Manuscript received May 16th 2013, revised Oct. 2nd 2013, March 26, 2014. This work was supported in part by NSF grant EFRI-1137172, IIP-1343402, FHWA grant DTFH61-12-H-00002, and ARO grant W911NF-09-1-0565.

Chucai Yi is with the Graduate Center, the City University of New York, New York, NY 10016 USA (e-mail: cyi@gc.cuny.edu).

YingLi Tian is with the City College and the Graduate Center, the City University of New York, New York, NY 10031 USA (phone: 212-650-7046; fax: 212-650-8249; e-mail: ytian@ccny.cuny.edu).

These two applications can be widely used in smart mobile device.

The main contributions of this paper are associated with the proposed two recognition schemes. Firstly, a character descriptor is proposed to extract representative and discriminative features from character patches. It combines several feature detectors (Harris-Corner, Maximal Stable Extremal Regions (MSER), and dense sampling) and Histogram of Oriented Gradients (HOG) descriptors [5]. Secondly, to generate a binary classifier for each character class in text retrieval, we propose a novel stroke configuration from character boundary and skeleton to model character structure.

The proposed method combines scene text detection and scene text recognition algorithms. Fig. 1 presents a flowchart of scene text extraction method. By the character recognizer, text understanding is able to provide surrounding text information for mobile applications, and by the character classifier of each character category, text retrieval is able to help search for expect objects from environment. Similar to other methods, our proposed feature representation is based on the state-of-the-art low-level feature descriptors and coding/pooling schemes. Different from other methods, our method combines the low-level feature descriptors with stroke configuration to model text character structure. Also, we present the respective concepts of text understanding and text retrieval and evaluate our proposed character feature representation based on the two schemes in our experiments. Besides, previous work rarely presents the mobile implementation of scene text extraction, and we transplant our method into an Android-based platform.

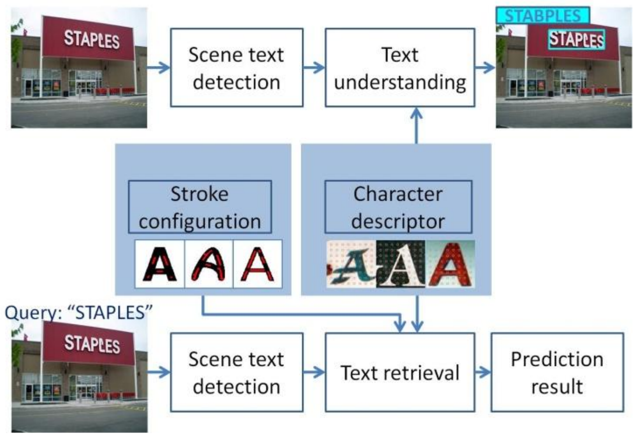


Fig. 1. The flowchart of our designed scene text extraction method: text understanding and text retrieval. It includes scene text detection (Section III) and two scene text recognition schemes (Section IV). In recognition, character descriptor (Section IV.A.) and stroke configuration (Section IV.B.) are designed to model character structure and compute discriminative character features.

II. Related Work

In this section, we present a general review of previous work on scene text recognition respectively. While text detection aims to localize text regions in images by filtering out non-text outliers from cluttered background [3, 7, 10, 16, 21, 28], text recognition is to transform image-based text information in the detected regions into readable text codes.

Scene text recognition is still an open topic to be addressed. In the Robust Reading Competition of International Conference on Document Analysis and Recognition (ICDAR) 2011 [19], the best word recognition rate for scene images is only about 41.2%. In general, scene text characters are composed of cross-cutting stroke components in uniform colors and multiple orientations, but they are usually influenced by some font distortions and background outliers. We observe that text characters from different categories are distinguished by boundary shape and skeleton structure, which plays an important role in designing character recognition algorithm. Current optical character recognition (OCR) systems [2, 23] can achieve almost perfect recognition rate on printed text in scanned documents, but cannot accurately recognize text information directly from camera-captured scene images and videos, and are usually sensitive to font scale changes and background interference which widely exists in scene text. Although some OCR systems have started to support scene character recognition, the recognition performance is still much lower than the recognition for scanned documents. Many algorithms were proposed to improve scene-image-based text character recognition. Weinman *et al.* [27] combined the Gabor-based appearance model, a language model related to simultaneity frequency and letter case, similarity model, and lexicon model to perform scene character recognition. Neumann *et al.*[17] proposed a real time scene text localization and recognition method based on extremal regions Smith *et al.* [22] built a similarity model of scene text characters based on SIFT, and maximized posterior probability of similarity constraints by integer programming. Mishra *et al*. [15] adopted conditional random field to combine bottom-up character recognition and top-down word-level recognition. Lu *et al*. [13] modeled the inner character structure by defining a dictionary of basic shape codes to perform character and word retrieval without OCR on scanned documents. Coates *et al.* [4] extracted local features of character patches from an unsupervised learning method associates with a variant of K-means clustering, and pooled them by cascading sub-patch features. In [31], a complete performance evaluation of scene text character recognition was carried out to design a discriminative feature representation of scene text character structure. In [20], a part-based tree structure model was designed to detect text characters under Latent-SVM [8], and recognize text words from text regions under conditional random field. In [33], Scale Invariant Feature Transform (SIFT) feature matching was adopted to recognize text characters in different languages, and a voting and geometric verification algorithm was presented to filter out false positive matches. In [25], generic object recognition method was imported to extract scene text information. A dictionary of words to be spot is built to improve the accuracy of detection and recognition. Character structure was modeled by HOG features and cross correlation analysis of character similarity for text recognition and detection. In [26], Random Ferns algorithm was used to perform character detection and constructed a system for query-based word detection in scene images.

III. Layout-Based Scene Text Detection

In natural scene, most text information is set for instruction or identifier. Text strings in print font are located at signage boards or object packages. They are normally composed of characters in uniform color and aligned arrangement, while non-text background outliers are in the form of disorganized layouts. Based on our previous work [29, 30], the color uniformity and horizontal alignment are employed to localize text regions in scene images. In our current work, scene text detection process is improved to be compatible with mobile applications.

A. Layout Analysis of Color Decomposition

According to our observations, the text on sign boards or print labels on nearby objects in general appear in uniform color. Thus we can group the pixels with similar color into the same layers, and separate text from background outliers in different colors.

To decompose a scene image into several color-based layers, we have designed a boundary clustering algorithm based on bigram color uniformity in our previous work [30]. Text information is generally attached to a plane carrier as attachment surface with uniform colors respectively. We define the uniformity of their color difference as bigram color uniformity. Color difference is related to the character boundary, which serves as a border between text strokes and the attachment surfaces. We then model color difference by a vector of color pair, which is obtained by cascading the RGB colors of text and attachment surfaces. Each boundary can be described by a color-pair, and we cluster the boundaries with similar color-pairs into the sample layer. The boundaries of text characters are separated from those of background outliers, as shown in Fig. 2.

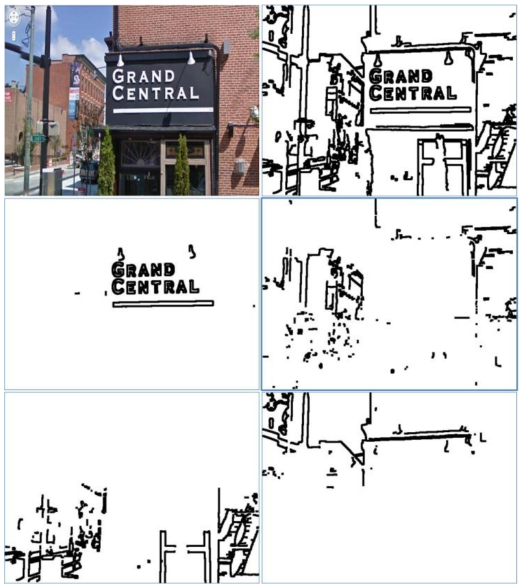


Fig. 2. Color decomposition of scene image by boundary clustering algorithm. The top row presents original scene image and the edge image obtained from canny edge detector. The other rows present color layers obtained from bigram color uniformity. It shows that the text information in signage board is extracted from complex background in a color layer.

B. Layout Analysis of Horizontal Alignment

In each color layer, we analyze geometrical properties of the boundaries to detect the existence of text characters. According to our observation, text information generally appears in text strings composed of several character members in similar sizes rather than single character, and text strings are normally in approximately horizontal alignment. Thus we design an adjacent character grouping algorithm in our previous work [29] to search for image regions containing text strings.

To model the boundary size and location of a text string, a bounding box is assigned to each boundary in a color layer. For each bounding box, we search for its siblings in similar size and vertical locations (horizontal alignment). If several sibling bounding boxes are obtained on its left and right, then we merge all these involved bounding boxes into a region. This region contains a fragment of text string. Then we repeat above method to calculate all text string fragments in this color layer, and merge the string fragments with intersections. Fig. 3 depicts the process of adjacent character grouping.

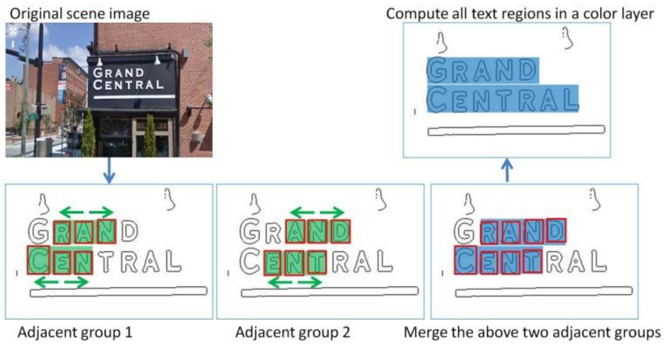

Original scene image
Compute all text regions in a color layer
Adjacent group 1
Adjacent group 2
Merge the above two adjacent groups

Fig. 3. The adjacent character grouping process. The red box denotes bounding box of a boundary in a color layer. The green regions in the bottom-left two figures represent two adjacent groups of consecutive neighboring bounding boxes in similar size and horizontal alignment. The blue regions in the bottom-right figure represent the text string fragments, obtained by merging the overlapping adjacent groups.

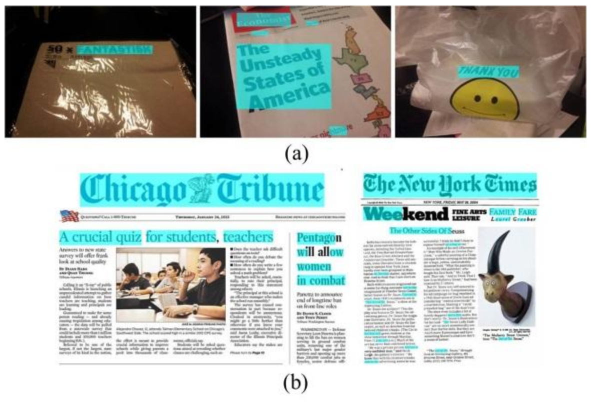

(a)
(b)
Chicago Tribune
The New York Times
Weekend
A crucial quiz for students, teachers
Pentagon will allow women in combat
The Unsteady States of America
THANK YOU

Fig. 4. (a) Our proposed text layout analysis can still be adaptive to oriented text string within reasonable range. (b) The method can handle some challenging cases of font variations, such as newspaper titles.

In order to extract text strings in slightly non-horizontal orientations (see Fig. 4(a)), we search for possible characters of a text string within a reasonable range of horizontal orientation. When estimating horizontal alignment, we do not require all the characters exactly align in horizontal orientation, but allow some differences between neighboring characters that are assigned into the same string. In our system we set this range as $\pm\pi/6$ degrees relative to the horizontal line. This range could be set to be larger but it would bring in more false positive strings from background. In addition, our scene text detection algorithm can handle challenging font variations, as long as the text has enough resolutions, such as newspaper titles, as shown in Fig. 4(b).

To be compatible with blind-assistant demo system, some technical details of our scene text detection algorithm are adjusted. At first the input image is down-sampled to improve the efficiency. Then in color decomposition, only the edge pixels from a boundary that satisfies specific geometrical constraints are adopted to build color layers. Also some parameters related to horizontal similarity and alignment are adjusted according to our evaluations in real environments.

IV. STRUCTURE-BASED SCENE TEXT RECOGNITION

From the detected text regions, character recognition is performed to extract text information. In current work, scene text characters include 10 digits [0-9] and 26 English letters in upper case [A-Z] and lower case [a-z], 62 character classes in total. Three public datasets are employed for training character recognizer and evaluating its performance, and these datasets contain image patches of complete and regular text characters cropped from scene images. We will provide detailed descriptions in Section VI.

As mentioned in Section I, we design two character recognition schemes. In text understanding, character recognition is a multi-class classification problem. We train a character recognizer to classify the 62 classes of characters. In text retrieval, character recognition is a binary classification problem. For each of the 62 character classes, we train a binary classifier to distinguish a character class from the other classes or non-text outliers. For example, we train a binary classifier for character class 'A', then this classifier will predict a patch containing 'A' as positive, and predict a patch containing other character classes or non-text outliers as negative. The specified character classes are defined as queried characters.

In both schemes, a robust character descriptor is required to extract structure features from character patches. In text retrieval, to better model character structure, we define stroke configuration for each character class based on specific partitions of character boundary and skeleton.

A. *Character Descriptor*

We propose a novel character descriptor to model character structure for effective character recognition. Fig. 5 depicts the flowchart of our proposed character descriptor.

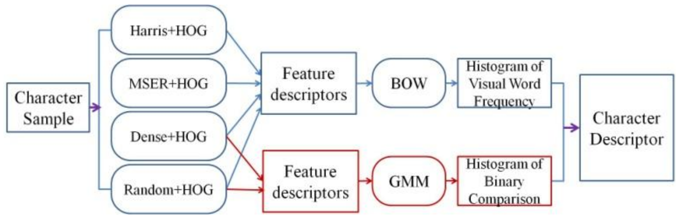


Fig. 5. Flowchart of our proposed character descriptor, which combines four keypoint detectors, and HOG features are extracted at keypoints. Then BOW and GMM are employed to respectively obtain visual word histogram and binary comparison histogram.

It employs four types of keypoint detectors, Harris detector (HD) to extract keypoints from corners and junctions, MSER detector (MD) to extract keypoints from stroke components, Dense detector (DD) to uniformly extract keypoints, and Random detector (RD) to extract the preset number of keypoints in a random pattern. As shown in Fig. 6, four feature detectors are able to cover almost all representative keypoints related to the character structure.

At each of the extracted keypoints, the HOG feature is calculated as an observed feature vector x in feature space. HOG is selected as local features descriptor because of its compatibility with all above keypoint detectors. Some other feature descriptors like SIFT and SURF are not employed in our method, because our experimental results show that their performance on character recognition is lower than HOG. It might be because SIFT (or SURF) descriptor relies on the keypoints obtained from SIFT (or SURF) detector of its own.

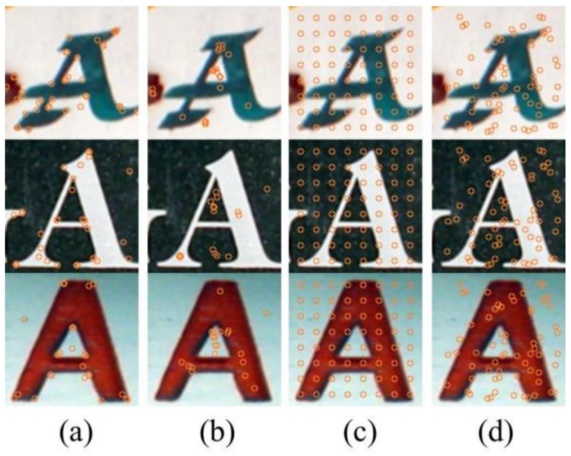


Fig. 6. Keypoints extracted respectively by the four detectors from three character patches. (a) Keypoints detected by HD. (b) Keypoints detected by MD. (c) Keypoints detected by DD. (d) Keypoints detected by RD.

Each character patch is normalized into size 128×128, containing a complete character. In the process of feature quantization, the Bag-of-Words (BOW) Model and Gaussian Mixture Model (GMM) are employed to aggregate the extracted features. BOW is applied to keypoints from all the four detectors. GMM is applied to those only from DD and RD, because GMM-based feature representation requires fixed number and locations of the keypoint all character patch samples, while the numbers and locations of keypoints from

HD and MD depend on character structure in the character patches. In both models, character patch is mapped into characteristic histogram as feature representation. By the cascade of BOW-based and GMM-based feature representations, we derive the character descriptor with significant discriminative power for recognition.

Bag-of-Words Model (BOW)

The BOW model represents a character patch from the training set as a frequency histogram of visual words. The BOW representation is computationally efficient and resistant to intra-class variations. At first, k-means clustering is performed on HOG features extracted from training patches to build a vocabulary of visual words. Then feature coding and pooling are performed to map all HOG features from a character patch into a histogram of visual words. We adopt soft-assignment coding and average pooling schemes in the experiments. More other coding/pooling schemes will be tested in our future work.

For each of the four feature detectors HD, MD, DD, and RD, we build a vocabulary of 256 visual words. This number of visual words is experimentally chosen to balance the performance of character recognition and the computation cost. At a character patch, the four detectors are applied to extract their respective keypoints, and then their corresponding HOG features are mapped into the respective vocabularies, obtaining four frequency histograms of visual words. Each histogram has 256 dimensions. Then we cascade the four histograms into BOW-based feature representation in $256 \times 4 = 1024$ dimensions.

Gaussian Mixture Model (GMM)

In DD and RD, keypoints are extracted from each character patch according to predefined parameters rather than character structure. In our experiments, DD generates a uniform 8×8 keypoint array and RD generates 64 keypoints randomly, but all character patches share the same random pattern. Therefore, the keypoints extracted by RD and DD are always located at the same positions in all character patches, as shown in Fig. 6. To describe the local feature distributions, we build a GMM over all character patches in training set. In our experiments, each GMM contains 8 Gaussian distributions. This parameter is selected from the best results of scene character recognition.

In the process of building GMM, k-means clustering $(k = 8)$ is first applied to calculate k centers of the HOG descriptors, where the s-th $(1 \leq s \leq k)$ center is used as initial means μ_s of the s-th Gaussian in GMM. Then the initial weights w_s and co-variances σ_s are calculated from the means. Next, an EM algorithm is used to obtain maximum likelihood estimate of the three parameters, weights, means, and co-variances of all the Gaussian mixture distributions. A likelihood vector from all Gaussians is represented by Eq. (1).

$$P_x = \langle w_s p_s(x|\mu_s, \sigma_s) \rangle \Big|_{s=1}^{k} = \langle w_1 p_1(x|\mu_1, \sigma_1), w_2 p_2(x|\mu_2, \sigma_2), \dots, w_s p_s(x|\mu_s, \sigma_s) \rangle \qquad (1)$$

$$p_s(x|\mu_s, \sigma_s) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

where x denotes a HOG-based feature vector at a keypoint, P_x denotes the likelihood vector of feature vector x, and $p_s(x|\mu_s, \sigma_s)$ denotes the probability value of x at the s-th Gaussian. Since there is $k = 8$ Gaussians in a GMM, the likelihood vector P_x is 8 dimensions.

At each keypoint extracted by DD or RD detectors, we can generate a corresponding likelihood vector. For likelihood vectors $\langle P_x, P_y \rangle$, where $P_x = \langle w_s p_s(x|\mu_s, \sigma_s) \rangle \Big|_{s=1}^{8}$ and $P_y = \langle w_s p_s(y|\mu_s, \sigma_s) \rangle \Big|_{s=1}^{8}$, we define a GMM-based feature representation by histogram of binary comparisons, as Eq. (2).

$$F_{x,y} = \sum_{s=1}^{8} \left[2^{s-1} \times (P^{(s)})\right] \qquad (2)$$
$$F = \langle F_{x,y} \rangle \Big|_{x=1, y=1}^{64}$$

where F denotes the histogram of binary comparisons, $F_{x,y}$ denotes one element of the histogram, $P^{(s)} = 1$ if $w_s p_s(x|\mu_s, \sigma_s) \geq w_s p_s(y|\mu_s, \sigma_s)$, and $P^{(s)} = 0$ if $w_s p_s(x|\mu_s, \sigma_s) < w_s p_s(y|\mu_s, \sigma_s)$. Fig. 7 illustrates the process of calculating one element of histogram of binary comparisons. From 64 keypoints of a character patch, the histogram of binary comparison has $\binom{64}{2} = 2016$ dimensions. In this process, to ensure consistent feature representation and feature dimension, the numbers and locations of keypoints from each patch should be identical, so GMM is only applied to the keypoints from DD and RD. The two histograms obtained respectively from DD and RD are cascaded into GMM-based feature representation.

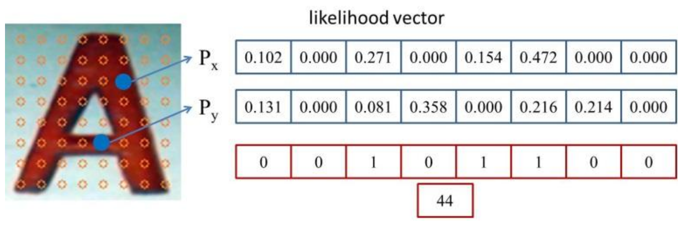


Fig. 7. A pair of likelihood vectors and the two corresponding keypoints from DD in a character patch. Red boxes indicate the results of binary comparisons in the form of binary value. Then it is transformed into a decimal value.

For a character patch, we generate its visual word histogram as BOW-based feature representation, and binary comparison histogram as GMM-based feature representation. Then the two feature representations are cascaded into the character descriptor of the patch. It is used as a feature vector of character patch to train text character recognizer in SVM model. The combination of BOW-based and GMM-based feature representations improve the performance on scene character recognition, which is prepared for text understanding. The evaluation results will be presented in Section VI.

B. *Character Stroke Configuration*

In text retrieval application, the query character class is considered as an object with fixed structure, and we generate its binary classifier according to structure modeling. Character structure consists of multiple oriented strokes, which serve as basic elements of a text character. From the pixel-level perspective, a stroke of printed text is defined as a region bounded by two parallel boundary segments. Their orientation is regarded as stroke orientation and the distance between them is regarded as stroke width. Stroke width consistency achieves outstanding performance on scene text detection [7]. In order to locate stroke accurately, stroke is redefined in our algorithm as skeleton points within character sections with consistent width and orientation. A character can be represented as a set of connected strokes with specific configuration which includes the number, locations, lengths and orientations of the strokes. Here, the structure map of strokes is defined as stroke configuration. In a character class, although the character instances appear in different fonts, styles, and sizes, the stroke configurations is always consistent. For example, character 'B' is always a vertical stroke with two arc strokes in any pattern. Therefore for each of the 62 character classes, we can estimate a stroke configuration from training patches to describe its basic structure.

We estimate stroke configuration by synthesized characters generated from computer software rather than scene characters cropped from scene images, because synthesized characters can provide an accurate boundary and skeleton related to character structure. The Synthetic Font Training Dataset proposed by [27] is employed to obtain stroke configuration. This dataset contains about 67400 character patches of synthetic English letters and digits in various fonts and styles, and we select 20000 patches to generate character patches. It covers all the 62 classes of characters. Each character image is normalized into 128×128 pixels with no anti-aliasing. In estimating stroke configuration, character boundaries and skeletons are generated to extract stroke-related features, which are used to compose stroke configuration. The implementation contains three main steps as follows.

Firstly, given a synthesized character patch from the training set, we obtain character boundary and character skeleton by applying discrete contour evolution (DCE) [11] and skeleton pruning on the basis of DCE [1]. The DCE simplifies characters into polygons of visual parts with a small number of vertices, and we define the polygon as a character boundary. Then skeleton pruning is performed to obtain a refined character skeleton, as shown in Fig. 8(b-c). Since DCE and skeleton pruning are invariant to deformation and scaling, they provide stable results of boundary and skeleton, which are used as a universal description of the structure of characters in the same category but with different fonts and sizes. However, DCE-based skeleton pruning can only obtain a coarse character skeleton by boundary analysis, but not predict its stroke configurations. Thus we further locate all the strokes that compose the character skeleton in the next two steps.

Secondly, we estimate the stroke width and orientation on sample points of character boundary. N points are sampled evenly from the polygon character boundary, with the polygon vertices reserved. In our experiment, we set $N = 128$. The number of points to be sampled on each side of the polygon boundary is proportional to its length. The sampled points of character boundary are shown in Fig. 8(b) in blue and orange, where the blue points are the reserved vertices, mostly located at the junctions and corners of the character.

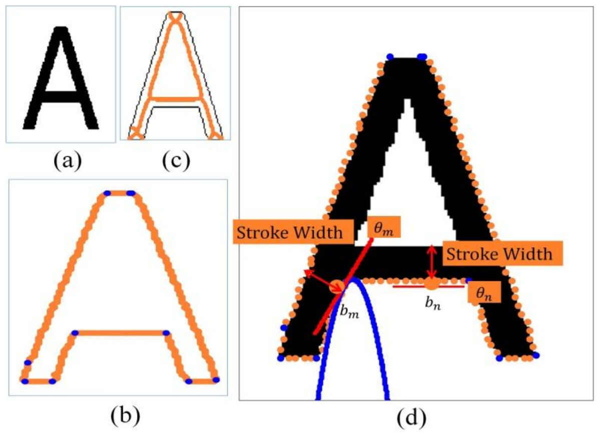


Fig. 8. (a) Character patch. (b) Polygon-based character boundary obtained from DCE, where blue points denote the polygon vertices. (c) Pruned skeleton based on the vertices. (d) Stroke orientations denoted by red line and stroke width denoted by red double arrows are calculated at the two boundary sample points b_m and b_n respectively, where b_n is approximately collinear with neighboring points and b_m fits a quadratic curve (in blue) with neighboring points.

Next, stroke width and orientation at each boundary sample point b is estimated. We take b and its two neighboring sample points to fit a line when they are approximately collinear or else a quadratic curve. Then the slope or tangent direction at b is used as stroke orientation. The stroke width is obtained from probing length along the normal vector at b until another boundary point is encountered (see Fig. 8(d)). From pixel-level perspective, the character boundary is a set of sampled boundary points as b. The character structure modeling is based on the boundary sample points.

Thirdly, we calculate the skeleton-based stroke maps from the consistency of stroke width and stroke orientation. At each boundary sample point, values of stroke width and orientation are compared with its neighboring points. We define width consistency as a difference of stroke widths no larger than 3 and orientation consistency as a difference of stroke orientations no larger than $\pi/8$. These parameters are compatible with the synthesized character patches with size 128×128. The sample points satisfying the stroke-related features construct stroke sections of the character boundary, while the other sample points, around the intersections of neighboring strokes or the ends of strokes, compose junction sections of a character boundary, as shown in Fig. 9(a). The corresponding skeleton points at stroke sections are extracted to construct the stroke configuration, as illustrated by red lines in the Fig. 9(b).

The characters from the same class always have similar stroke configurations, so the basic structure of a character class can be described by the mean value of all stroke configurations from character samples of the class. We use a stroke alignment method to estimate a mean value of stroke configuration so that it is able to handle various fonts, styles and sizes. An objective function of stroke alignment is defined as Eq. (3).

$$E = \sum_i \left(D\left(\bar{S}, T_i(S_i)\right) + g(T_i) \right) \tag{3}$$

$$D(S_m, S_n) = \sum_p \|S_m(p) - S_n(p)\|^2 \tag{4}$$

where D represents the distance between the stroke configurations of two character samples as Eq. (4), and $\bar{S}$ represents mean value of the stroke configurations. T_i represents the transformation applied on the strokes of the i-th stroke configuration S_i. This transformation consists of transition and scaling of the stroke configuration (not including rotation). $g(T_i)$ represents the amplitude of the transformations, which is used to suppress large transformations. This problem is solved on the basis of the alignment method in [9], to calculate a proper $\bar{S}$ to minimize E in Eq. (3).

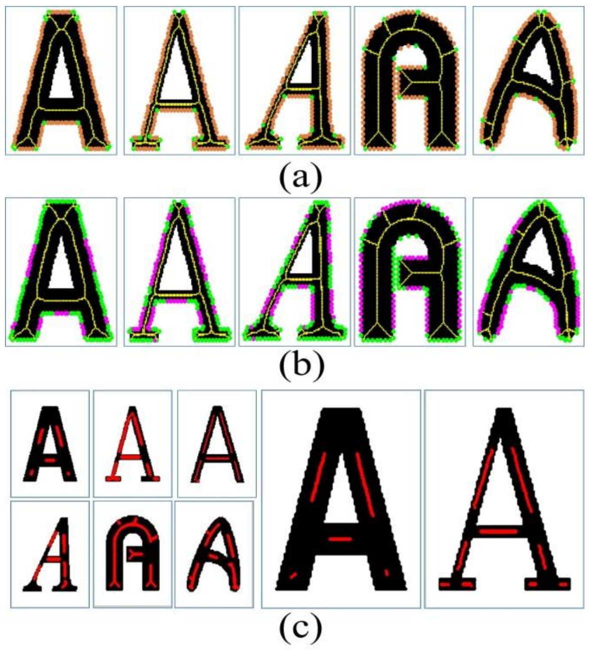


Fig. 9. For character samples 'A' in different fonts and styles. (a) Top row: boundary samples points in blue and orange, where the blue ones represent polygon vertices; Bottom row: stroke sections (pink points) and junction sections (green points) at the boundaries. (b) Stroke configurations of character 'A' in different styles and sizes are approximately consistent. The two large characters are from the first two small ones.

For each of the 62 character classes, we align stroke configurations from all its character patches, and calculate the mean value. As illustrated in Fig. 10(a), stroke configurations of four character classes, including 'S', 'A', '4', and 'W', respectively describe their character structures. The intensity in stroke configuration indicates the frequency of occurrences of character strokes (e.g. darker indicates higher frequency). The mean values of stroke configurations are manually labeled as skeletons along the dark stroke components, as shown in Fig. 10(b). These skeletons will be employed to analyze character structure. It shows that text characters in natural scene mostly appear in regular fonts, which is also one of our assumptions about scene text. Next, stroke configuration is divided into several partitions, and each partition is a sub-patch containing 2 or 3 neighboring stroke components, as the cyan boxes in Fig. 10(c). Thus we can generate a stroke configuration map of sub-patches for each character class.

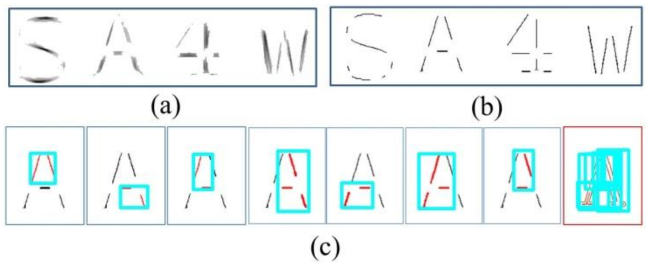


Fig. 10. (a) Stroke configuration of four character classes. (b) Extracted stroke points, where each segment represents a mean stroke of the basic character structure. (c) 7 partitions of character 'A' are marked in red, generating sub-patches on its stroke configuration.

To determine that a given character patch belongs to a character class, it is first partitioned into sub-patches according to stroke configuration map of the character class. Then we calculate our proposed character descriptor from each sub-patch, and cascade all of them into feature vector of the character patch. This feature vector is used for training the binary character classifier.

In the learning of character classifier for text retrieval, only the patches from queried character class are considered as positive samples, and the patches of character classes or non-text outliers are regarded as negative samples. Thus the number of negative samples is much larger than that of positive samples. To handle the imbalanced data, we adopt cascade Adaboost learning scheme [24] to train the binary character classifier for each character class. Whenever a queried word is input into text retrieval application, it is first divided into single queried characters. Then their corresponding character classifiers are invoked to confirm the character classes. If most of the queried characters exist, the text retrieval application will provide positive response, otherwise provide negative response.

V. Demo System

We have developed demo systems of scene text extraction in Android-Mobile platforms. We integrate the functional modules of scene text detection and text recognition. It is able to detect regions of text strings from cluttered background, and recognize characters in the text regions.

Compared with a PC platform, the mobile platform is portable and more convenient to use. Scene text extraction will be more widely used in mobile applications, so it is indispensible to transplant demo system into the popular

Android mobile platform. However, two main challenges should be overcome in developing the scene text extraction application in mobile platform. First, our proposed method is implemented by C++ programming, while Android platform is based on Java engine. Fortunately, Android NDK is provided to compile C++ code into the Android platform. Second, due to the limitations of computing speed and memory allocation in mobile device, we attempt to make our implementations efficient enough for real applications. To improve the efficiency, we skip layout analysis of color decomposition in text detection, but directly apply the canny edge map for layout analysis of horizontal alignment. It lowers the accuracy of text detection, but is still reliable for text extraction from nearby object in enough resolutions. In addition, code optimization is performed. In our test, each frame spends about 1 second in completing the whole process of scene text extraction. One of demo systems runs on Samsung Galaxy II smart phone with Android 2.3 as shown in Fig. 11. It captures natural scene by the phone camera, and extract text information online from the captured scene images, which are frame-by-frame processed.

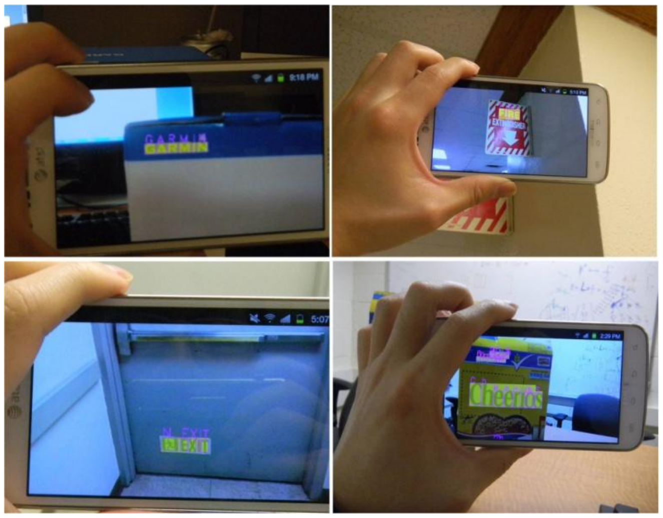


Fig. 11. Our demo system of scene text extraction in Android platform. The text strings "GARMIN", "FIRE", "EXIT", and "Cheerios" are extracted from complex background. Although some characters are incorrectly recognized in current demo, we will introduce lexicon analysis to improve the recognition performance.

In blind-assistant application, the demo system has been used to assist blind or visually impaired people to recognize hand-held object. Another demo system consists of camera, processing device, and Bluetooth earplug. In this system, our proposed method is implemented and deployed into the processing device. Camera is used as input device for capturing natural scene, and Bluetooth earplug is used as output device for broadcasting the recognized text information. As shown in

Fig. 12, this demo system reads the text labels in the hand-held objects and informs blind users of the extracted text codes. With the approval of Human Subjects Institutional Review Board, this prototype system has been evaluated by 10 blind subjects. Each blind person spent about 2 hours to evaluate the system and collected a dataset of 15 different hand-held objects.

The demo system in mobile platform gives us some insight into algorithm design and performance improvement of scene text extraction. First, the assumptions of horizontal alignment in text layout analysis make sense in mobile applications. Although some text detection algorithms attempts to extract text strings in arbitrary orientations [28, 29], they usually bring in more false positive text regions and lower the efficiency. However, the user can rotate the lightweight mobile devices to adaptively fit the non-horizontal text strings. Second, the accuracy of scene text detection could be improved by using the intersections of extracted text regions from consecutive frames captured by the camera at an identical scene.

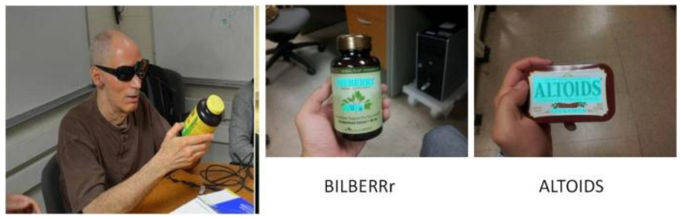


Fig. 12. Demo system in a Window-PC platform assists blind user read text information from hand-held objects, including the detected text regions in cyan and the recognized text codes.

VI. Quantitative Experimental Analysis

Scene text extraction consists of detection and recognition. However, the main technical contributions of this paper are the two scene character recognition schemes compatible with mobile applications. We perform experiments to evaluate the two schemes over benchmark datasets.

A. *Datasets*

To evaluate the proposed character descriptor and the character stroke configuration, we employ three public datasets of scene text characters, in which we conduct scene character recognition. The first one is Chars74K EnglishImg Dataset published in [6]. It contains all the 62 character classes with the approximately balanced number of samples. The samples in this dataset are divided into two categories, GoodImg and BadImg, according to the recognition difficulty. The second one is Sign Dataset published by Weinman *et al.* [27]. It captures 96 camera-based signs with 1209 scene characters. Most of the characters appear in regular font and style consistent with documents. The third one is ICDAR-2003 Robust Reading Dataset. It contains about 11600 character samples which are cropped from text regions of natural scene images. ICDAR-2003 Dataset [14] is very challenging because large amounts of non-text background outliers interfere with the cropped character samples, and many character samples have a small size that does not have enough resolution for recognition. In Sign Dataset and ICDAR-2003 Dataset, the number of character samples from different categories is unbalanced.

B. Scene Character Recognition for Text Understanding

In performance evaluations of text understanding, we use accuracy rate (AR) as evaluation measure, which is defined as the ratio between the number of correctly recognized text characters and the total number of text characters.

Table I presents the AR of our proposed character descriptor in Chars74K dataset by integrating BOW-based and GMM-based feature representations. In this dataset, the fixed number of samples from each category is selected for evaluating character recognition performance. Each character category takes 15 character patches as training samples to learn the character recognizer, and another 15 characters as testing samples to evaluate the recognition performance. Each of the 62 categories has 30 samples, which are used to evaluate character recognition performance in a cross validation process. Our method obtains better performance of scene character recognition than previous algorithms. In addition, we further evaluate the two feature representations of our character descriptor independently. BOW-based feature representation obtains 0.53 and GMM-based feature representation 0.47. This may be due to the fact that BOW-based representations cover keypoints from all four detectors, while GMM-based representations rely only on DD and RD keypoints. In DD and RD, it is unavoidable that some keypoints are not located on characters.

TABLE I
ACCURACY RATES OF SCENE CHARACTER RECOGNITION IN CHARS74K DATASET, COMPARED WITH PREVIOUSLY PUBLISHED RESULTS [6, 26]

Chars74K Dataset	AR
Ours	0.60
Ours (BOW-based representation only)	0.53
Ours (GMM-based representation only)	0.47
ABBYY	0.31
HOG+NN	0.58
SYNTH+FERNS	0.47
NATIVE+FERNS	0.54
MKL	0.55

NN: Nearest neighbor; SYNTH: synchronic character patch for training for training; NATIVE: Nature scene characters for training; FERNS: Random ferns algorithm; MKL: Multiple-kernel learning

Table II presents the AR of our proposed character descriptor in ICDAR-2003 dataset. This dataset provides the standard splits of training samples and testing samples, and we filter out the samples that do not belong to the 62 categories of English letters and digits or do not have enough resolution, and select the resting test samples for evaluating character recognition. In [26], the HOG features and Nearest Neighbor classification obtains the best performance in the ICDAR-2003 Dataset. Besides, synthetic images (SYNTH) and scene images (NATIVE) are respectively adopted as training samples, and the Random Ferns model is used to extract character structure features. The experimental results in Table II show that our proposed descriptor outperforms the SYNTH+FERNS and comparable with NATIVE+FERNS in [26].

Chars74K and ICDAR-2003 datasets provide enough character samples in each category to train character recognizer. But the third dataset, Sign Dataset, has a limited number of samples and imbalanced character categories, which cannot generate an effective character recognizer for performance evaluation of text understanding. Besides, the character recognizer obtained from other datasets cannot ensure fair performance comparison. Thus we skip the Sign Dataset in the performance evaluation of text understanding.

TABLE II
ACCURACY RATES OF SCENE CHARACTER RECOGNITION IN ICDAR-2003 DATASET, COMPARED WITH PREVIOUSLY PUBLISHED RESULTS [26].

ICDAR-2003 Dataset	AR
Ours	0.628
HOG+NN	0.515
SYNTH+FERNS	0.520
NATIVE+FERNS	0.640

NN: Nearest neighbor; SYNTH: synchronic character patch for training for training; NATIVE: Nature scene characters for training; FERNS: Random ferns algorithm; MKL: Multiple-kernel learning

C. Scene Character Recognition for Text Retrieval

In character structure modeling, the proposed character descriptor is applied to extract structure features from stroke configuration of the characters to learn a binary classifier for each character class. We evaluate these binary classifiers by queried character classification in the above three datasets.

In each character class, two measurements, accuracy rate (AR) and false positive rate (FPR), are calculated to evaluate the performance of queried character classification. FPR represents the ratio between the number of incorrectly predicted negative samples and the total number of negative samples. We obtain ARs and FPRs by querying each of the 62 character classes, and then calculate the average as evaluation results. In Table III, a character classifier is trained for each character category by using Chars74K samples, which is then evaluated over the three datasets to obtain the results.

TABLE III
ACCURACY RATES (AR) AND FALSE POSITIVE RATES (FPR) OF QUERIED CHARACTER CLASSIFICATION IN THE THREE DATASETS

Dataset	AR	FPR
Chars74K	0.726	0.078
Sign	0.868	0.075
ICDAR-2003	0.536	0.180

The character classes with more discriminative structure features obtain higher ARs and lower FPRs. Chars74K Dataset has the approximately balanced number of character samples among all the categories, so it can generate an applicable comparison of the robustness of the 62 binary classifiers. Fig. 13 illustrates the ARs of all character classes in Chars74K Dataset. Some categories, such as “I” and “l”, do not have discriminative structure model, resulting in lower classification ARs. Some categories like ‘A’, ‘E’ and ‘Y’ have relatively higher ARs because their structure model is more discriminative.

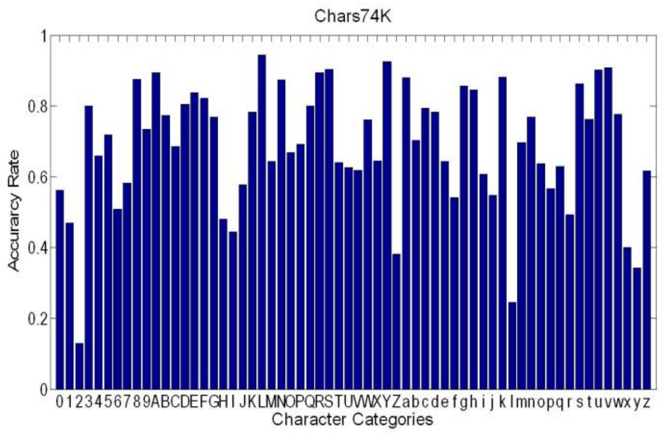


Fig. 13. AR of query-based character recognition at all the 62 character classes of the Chars74K Dataset.

ARs of all character classes in Sign Dataset are obtained as Fig. 14. In this dataset, there is large number difference between the numbers of character samples from different categories. The categories ‘7’, ‘8’, ‘Q’, ‘q’ and ‘x’ do not have samples at all, so their ARs are 0. Some categories have only 1 or 2 samples, so their ARs are most either 100% or 0%.

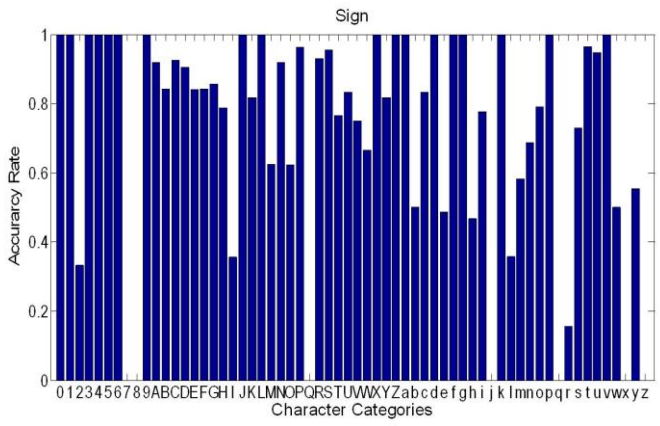


Fig. 14. AR of query-based character recognition at all the 62 character classes of the Sign Dataset.

In ICDAR-2003 Dataset, the numbers of character samples in different classes are also unbalanced. Fig. 15 depicts the ARs of all character classes in this dataset. The categories with more samples mostly obtain larger ARs. It is because these characters are more probably cropped from nearby text signage with enough resolutions and less background interferences.

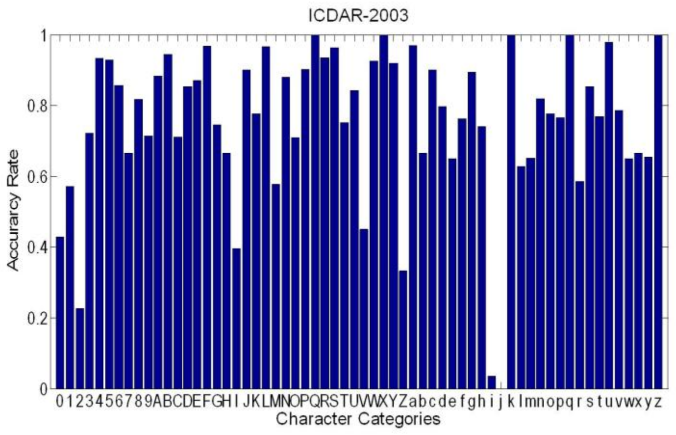


Fig. 15. AR of query-based character recognition at all the 62 character classes of the ICDAR-2003 Dataset.

VII. CONCLUSIONS

We have presented a method of scene text recognition from detected text regions, which is compatible with mobile applications. It detects text regions from natural scene image/video, and recognizes text information from the detected text regions. In scene text detection, layout analysis of color decomposition and horizontal alignment is performed to search for image regions of text strings. In scene text recognition, two schemes, text understanding and text retrieval, are respectively proposed to extract text information from surrounding environment. Our proposed character descriptor is effective to extract representative and discriminative text features for both recognition schemes. To model text character structure for text retrieval scheme, we have designed a novel feature representation, stroke configuration map, based on boundary and skeleton. Quantitative experimental results demonstrate that our proposed method of scene text recognition outperforms most existing methods. We have also implemented the proposed method to a demo system of scene text extraction on mobile device. The demo system demonstrates the effectiveness of our proposed method in blind-assistant applications, and it also proves that the assumptions of color uniformity and aligned arrangement are suitable for the captured text information from natural scene.

In future work, we will improve the accuracy rate of text detection, and add lexicon analysis to extend our system to word-level recognition. To improve the accuracy and practicality of scene text extraction, we will design more representative and discriminative features to model text structure. We will collect a database of specific scene text words as stronger training set, for example, a set of word patches “EXIT” or “SALE” cropped from scene images. In addition, we will combine scene text extraction with other techniques like content-based image retrieval to develop more useful vision-based assistant system.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments and insightful suggestions that improved the quality of this manuscript.

References

[1] X. Bai, L. Latecki, and W. Liu, "Skeleton prunning by contour partitioning with discrete curve evolution," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007.

[2] R. Beaufort, and C. Mancas-Thillou, "A weighted finite-state framework for correcting errors in natural scene OCR," Proceedings of International Conference on Document Analysis and Recognition, 2007.

[3] X. Chen, J. Yang, J. Zhang, and A. Waibel, "Automatic Detection and Recognition of Signs From Natural Scenes," IEEE Transactions on Image Processing, vol. 13, no. 1, pp. 87-99, 2004.

[4] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng, "Text detection and character recognition in scene images with unsupervised feature learning, " ICDAR 2011.

[5] N. Dalal, and B. Triggs, "Histograms of Oriented Gradients for Human Detection," Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[6] T. de-Campos, B. Babu, and M. Varma, "Character recognition in natural images," VISAPP, 2009.

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

[8] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, Sep. 2010.

[9] T. Jiang, F. Jurie, C. Schmid, "Learning shape prior models for objects matching," CVPR, 2009.

[10] S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. Johsi, "Text Extraction and Document Image Segmentation Using Matched Wavelets and MRF Model," IEEE Transactions on Image Processing, 2007.

[11] L. Latecki, and R. Lakamper, "Convexity rule for shape decomposition based on discrete contour evolution," Computer Vision and Image Understanding, 1999.

[12] Y. Liu, J. Yang, and M. Liu, "Recognition of QR code with mobile phones, " Control and Decision Conference, pp. 203-206, 2008.

[13] S. Lu, L. Li, and C. Tan, "Document image retrieval through word shape coding," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.

[14] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "ICDAR 2003 Robust Reading competitions," Proceedings of International Conference on Document Analysis and Recognition, 2003.

[15] A. Mishra, K. Alahari, and C. Jawahar, "Top-down and bottom-up cues for scene text recognition, " IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[16] N. Nikolaou, and N. Papamarkos, "Color Reduction for Complex Document Images," International Journal of Imaging Systems and Technology, 2009.

[17] L. Neumann and J. Matas, "Real-time scene text localization and recognition," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[18] E. Ohbuchi, H. Hanaizumi, and L. Hock, "Barcode reader using the camera device in mobile phones," International Conference on Cyberworlds, pp. 260-265, 2004.

[19] A. Shahab, F. Shafait, and A. Dengel, "ICDAR 2011 Robust Reading Competition Challenge 2," Proceedings of International Conference on Document Analysis and Recognition, 2011.

[20] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang, "Scene Text Recognition using Part-based Tree-structured Character Detection," CVPR 2013.

[21] P. Shivakumara, W. Huang, and C. Tan, "An Efficient Edge based Technique for Text Detection in Video Frames," IAPR Workshop on Document Analysis Systems, 2008.

[22] D. Smith, J. Feild, and E. Learned-Miller, "Enforcing Similarity Constraints with Integer Programming," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[23] R. Smith, "An Overview of the Tesseract OCR Engine," Proceedings of International Conference on Document Analysis and Recognition, 2007.

[24] P. Viola, and M. Jones, "Robust real-time face detection," International Journal of Computer Vision, 2004.

[25] K. Wang, and S. Belongie, "Word spotting in the wild," Proceedings of European Conference on Computer Vision, 2010.

[26] K. Wang, B. Bbenko, and S. Belongie, "End-to-End scene text recognition," Proceedings of International Conference on Computer Vision, 2011.

[27] J. Weinman, E. Learned-Miller, and A. Hanson, "Scene text recognition using similarity and a lexicon with sparse belief propagation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009.

[28] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting Texts of Arbitrary Orientations in Natural Images," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2012.

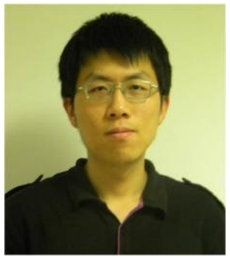
[29] C. Yi, and Y. Tian, "Text string detection from natural scenes by structure-based partition and grouping," IEEE Transactions on Image Processing, 2011.

[30] C. Yi and Y. Tian, "Localizing Text in Scene Images by Boundary Clustering, Stroke Segmentation, and String Fragment Classification, " In *IEEE Transactions on Image Processing*, Vol. 21, No. 9, 2012.

[31] C. Yi, X. Yang and Y. Tian, "Feature Representations for Scene Text Character Recognition: A Comparative Study," ICDAR 2013.

[32] J. Zhang, and R. Kasturi, "Extraction of Text Objects in Video Documents: Recent Progress," In the Eighth IAPR Workshop on Document Analysis Systems (DAS), 2008.

[33] Q. Zheng, K. Chen, Y. Zhou, G. Cong, H. Guan, "Text localization and recognition in complex scenes using local features," *ACCV* 2010



Chucai Yi (S'12) received his B.S. and the M.S. degrees in Department of Electronic and Information Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2007 and 2009, respectively. From 2009 he is a Ph.D. graduate student in Computer Science at the Graduate Center, the City University of New York, New York, NY, USA.

His research focuses on text detection and recognition in natural scene images. His research interests include object recognition, image processing, and machine learning. His current work is to develop computer vision algorithms and systems to help people with severe vision impairment to independently find doors, rooms, elevators, stairs, bathrooms, and other building amenities in unfamiliar indoor environments.

YingLi Tian (M'99–SM'01) received her BS and MS from TianJin University, China in 1987 and 1990 and her PhD from the Chinese University of Hong Kong, Hong Kong, in 1996. After holding a faculty position at National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Beijing, she joined Carnegie Mellon University in 1998, where she was a postdoctoral fellow of the Robotics Institute. Then she worked as a research staff member in IBM T. J. Watson Research Center from 2001 to 2008. She is currently a professor in Department of Electrical Engineering at the City College of New York and Department of Computer Science at the Graduate Center, the City University of New York. Her current research focuses on a wide range of computer vision problems from motion detection and analysis, assistive technology, to human identification, facial expression analysis, and video surveillance. She is a senior member of IEEE.