Localizing Text in Scene Images by Boundary Clustering, Stroke Segmentation, and String Fragment Classification

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

***Abstract*— In this paper, we propose a novel framework to extract text regions from scene images with complex backgrounds and multiple text appearances. This framework consists of three main steps: boundary clustering (BC), stroke segmentation, and string fragment classification. In BC, we propose a new bigram-color-uniformity-based method to model both text and attachment surface, and cluster edge pixels based on color pairs and spatial positions into boundary layers. Then, stroke segmentation is performed at each boundary layer by color assignment to extract character candidates. We propose two algorithms to combine the structural analysis of text stroke with color assignment and filter out background interferences. Further, we design a robust string fragment classification based on Gabor-based text features. The features are obtained from feature maps of gradient, stroke distribution, and stroke width. The proposed framework of text localization is evaluated on scene images, born-digital images, broadcast video images, and images of handheld objects captured by blind persons. Experimental results on respective datasets demonstrate that the framework outperforms state-of-the-art localization algorithms.**

***Index Terms*— Bigram color uniformity, boundary clustering (BC), color assignment, Gabor-based text features, string fragment classification, stroke segmentation, text localization.**

I. Introduction

IMAGE-BASED text information serves as an important indicator in many applications. It provides instructions and presentations for navigation, assistive reading, geocoding, and content-based image retrieval, and so on. It is a challenging task to detect and recognize text from camera-captured images due to two main issues: 1) variety of text patterns (sizes, fonts, colors, orientations, etc.) and 2) existence of background outliers resembling text characters, such as windows, bricks, and character-like texture. Most optical character recognition (OCR) systems are designed to transform text images to readable text codes [1], [2], but perform poorly when text is embedded into complex background because of background interferences and low frequency of occurrence of text. As we know, most voice pens for blind assistance require manual localization of text lines in documents and objects in hand. However, in camera-based scene images, manual text localization is impractical, especially for blind people. Therefore, algorithms of automatic text localization are required to filter out background outliers and localize the regions containing text characters or strings in images for further processes of text segmentation and recognition.

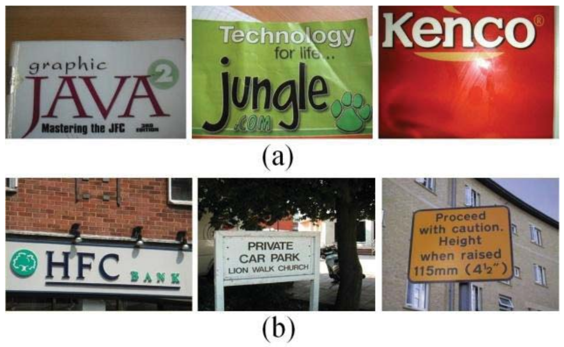


Fig. 1. Examples of text in natural scene images. (a) Text with high resolution and a relatively simple background. (b) Text with a complex background.

Natural scene images with text information are divided into two categories according to the complexity of the background. In the first category, text characters and strings are in high resolution with a relatively simple background. This category of scene images generally contains the close-up shots of specific objects, such as book covers, notice signages, and wrappers [see Fig. 1(a)]. Most images of born-digital documents and handheld objects captured by blind persons belong to this category. We will provide more details on this in Section V. The second category embeds text into more complex backgrounds with various natural objects, such as buildings, trees, lawns, roads, and so on [see Fig. 1(b)]. In both categories, text characters and strings mostly appear in print patterns with regular structure.

We observe that characters and strings in natural scene images are printed in same color for almost all cases. Many previous text localization algorithms applied color-based clustering to group the pixels in similar colors into respective color layers. Thus, text characters and strings could be separated

Manuscript received September 4, 2011, revised April 15, 2012; accepted April 30, 2012. Date of publication May 15, 2012; date of current version August 22, 2012. This work was supported in part by the National Institutes of Health under Grant 1R21EY020990, in part by the National Science Foundation under Grant IIS-0957016, Grant EFRI-1137172, and Grant DTFH61-12-H-00002, in part by ARO under Grant W911NF-09-1-0565, in part by Microsoft Research, and in part by a CITY SEEDs Grant. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Patrick Flynn.

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

Y. Tian was with City College, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA. He is now with City College, City University of New York, New York, NY 10031 USA (e-mail: ytian@ccny.cuny.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2012.2199327

from background objects in different colors. Nikolaou *et al.* [3] proposed an algorithm of color reduction based on color histogram and mean-shift algorithm. It initialized color centers randomly and projected each pixel to the nearest color center. Then mean-shift algorithm was applied to fix the color centers into the mean positions as final color layers. Chen *et al.* [4] established Gaussian mixture model (GMM) in five color channels (red, green, blue, hue, and intensity) to analyze the distributions of text pixels and background pixels. The parameters of these characteristic distributions were then used to label candidate text regions. Cosine similarity [5] and K-means clustering [6] were, respectively, applied to RGB channels to segment text characters in uniform color. In these algorithms, color clustering is a single-variable function that maps each pixel to the nearest color center. From the high-level perspective, it maps each text string in uniform color to the most compatible color layer. However, these algorithms ignore that text string is attached to the neighboring surface in uniform color in most cases. Based on this feature, we design the algorithm of boundary clustering (BC) which will be presented in Section II.

Another significant property of text is stroke width consistency. As the basic element of text, stroke is defined as a connected region in the form of a band of approximately constant width [7]. Compared to character height and width, stroke width reflects the text size from the perspective of character structure. Due to the stroke width consistency of characters and strings, a regional stroke width distribution can be employed to verify whether the localized areas contain text or not. In [8] and [9]. Stroke width was calculated by using the horizontal scan line to record the intensity variations around the edge pixels, usually a pair of impulses on the strokes with equal magnitudes and in opposite directions. A character consists of strokes in multiple orientations, while the horizontal scan line can only derive the width of vertical strokes. Epshtein *et al.* [7] proposed a more robust stroke width operator for text region localization, named as stroke width transform. In stroke-based algorithms, stroke width consistency is used to extract the pixels inside the body of strokes. The connected stroke pixels can be grouped into connected components as candidate text characters for further analysis in text localization.

In natural scene images, text information is usually printed as a text string, which is a group of characters, rather than a single character, and the character members of a string are most likely with similar size, consistent color, and aligned arrangement. Based on these features, layout analysis can be performed to extract text strings without character recognition [7], [10]–[12].

In addition to color uniformity, stroke width consistency, and character alignment, boundary-based and gradient-based structural analysis also plays an important role in text localization [13]–[19]. Text features, such as edge distribution, gradient variation, closed component boundary, and edge-based filter response, were obtained from boundary maps and gradient maps of scene images to detect and verify text regions. They were related to the geometrical structure of text.

In the above algorithms, multiple pixel-based features were employed to distinguish text characters and strings from background outliers. But most of the processes were based on the subjective selection of features and the hard assignment of parameters. For example, it was defined that aspect ratio of text characters should not exceed 5 and stroke width should be no larger than 50. But these estimates cannot accurately model the inner structure of text characters. They fail to filter out the background objects that are also composed of strokes in uniform color and consistent width, such as windows, bricks, and stripe texture. Thus the three common features are not enough for an accurate text model. In this case, some localization algorithms employed Haar-like features or wavelet analysis to train a text classifier in a machine-learning model. Chen *et al.* [10] applied block patterns to gradient-based maps and histograms to train a text classifier in an adaptive boosting (AdaBoost) model. Hanif *et al.* [20] extracted the mean difference, standard deviation, and histogram of oriented gradients features of text characters to generate a text detector under a complex AdaBoost model. In [21], the responses of globally matched wavelet filters from text regions are used as features to train a text classifier based on support vector machine (SVM) model and Fisher model. Pan *et al.* [22] used steerable Gabor filters to extract rotation-invariant features of multiple scripts. Shi *et al.* [23] adopted gradient-based curvatures to perform a structural analysis of handwritten digits under a Bayes discriminant model. Jung *et al.* [24] proposed an algorithm of text line refinement by analyzing the SVM score of text regions. Inspired by these algorithms, we design Gabor-based features from a set of block patterns and feature maps to model the inner structure of text and classify string fragments (see Section IV).

In this paper, we propose a novel framework of automatic text localization to calculate text regions in natural scene images by using features at three levels: pixel, character, and string. At the pixel level, assuming that text characters and strings in scene images mostly appear in uniform color, the edge pixels are clustered into several layers to separate the boundaries of text strokes from those of background outliers with different color pairs. At the character level, assuming that each text character is composed of a single stroke, the pixels inside the body of strokes are segmented from each boundary layer to extract the candidate characters in the form of connected components. At the string level, assuming that scene text is mostly in the form of approximately horizontal strings, layout analysis is first performed to group the horizontally aligned connected components into candidate fragments of text strings, and then a text classifier is learned from training set to predict whether an image patch of the candidate string fragment contains text or not.

We propose novel algorithms to extract more robust features for text localization. Fig. 2 depicts the flowchart of our framework. The main contributions include the following.

1) We design a color pair clustering algorithm based on GMM and expectation-maximization (EM) algorithm to group the boundary pixels with bigram color uniformity on the border of text and attachment surface.
2) We combine structural analysis of stroke boundary with color assignment for extracting the pixels inside the body

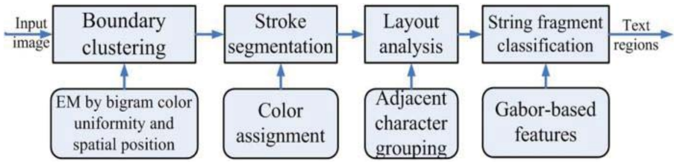


Fig. 2. Flowchart of our framework for text localization in natural scene images with complex background.

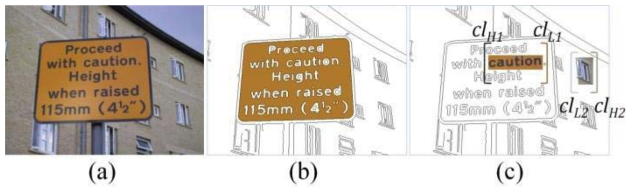


Fig. 3. (a) Scene image. (b) Attachment surface of text. (c) Two examples of color pairs corresponding to the boundaries at the signage board and window grid, respectively.

of strokes on each boundary layer.

3) To classify the string fragments, we model text features by applying block patterns to feature points of gradient maps, stroke distribution maps, and stroke width maps. The feature points are derived from maximum responses of Gabor filters.

The rest of this paper is organized as follows. Section II describes the proposed BC based on bigram color uniformity. Section III presents two new algorithms of stroke segmentation combining structural analysis and color assignment. Section IV describes the Gabor-based text features for training an SVM-based classifier of string fragments. We evaluate the framework and discuss the results according to the experiments on benchmark datasets in Section V. The proposed framework and the results presented in this paper and future research directions are summarized in Section VI.

II. Boundary Clustering

Boundary plays an important in role in the structural analysis and the geometrical model of text. In scene images, object boundary is derived from the color difference of two uniform regions: object and its surrounding backgrounds. Thus color uniformity and spatial positions are employed to analyze the boundaries of text characters, and we propose a clustering algorithm to separate them from the boundaries of background outliers.

A. Bigram Color Uniformity and Spatial Position

Camera-based scene images usually have complex backgrounds filled with nontext objects in multiple shapes and colors. In these images, text strokes, characters, and strings keep conspicuous by consistent colors. Thus many color-based clustering algorithms of text localization and text segmentation are designed [3]–[5]. However, these clustering algorithms ignore the color difference of neighboring pixels around the object boundaries. Compared to absolute color values, color difference is more suitable for the analysis of object shape and texture, because it produces more accurate local texture measure and is more robust to lighting changes [26].

We observe that text information is generally attached to a plane carrier as the attachment surface. The attachment surface consists of pixels with uniform color near the character boundaries but outside the character strokes, as shown in Fig. 3. We define the color uniformity of both text and attachment surface as bigram color uniformity, modeled by a color pair composed of their colors. For text and attachment surface, the color pair reflects their respective color uniformity as well as the color difference between them. Text boundaries on the border of text and its attachment surface are described by characteristic color pairs, and we are able to extract text by distinguishing the boundaries of characters and strings from those of background outliers based on color pairs [see Fig. 3(c)]. Here, we design an algorithm to group object boundaries with similar color pairs into respective maps, which are called boundary layers. Section III will present detailed descriptions of the clustering algorithm.

To reduce mutual interferences between text strings, text boundaries in different positions should be assigned into different boundary layers as possible, even though they have uniform color values and similar color differences. But most previous color-based clustering algorithms did not take into account the spatial positions of text. Our clustering algorithm employs the spatial positions of edge pixels at object boundaries as additional features.

B. EM-Based BC

In natural scene images, an initial map of object boundary is calculated from Canny detection [27]. Edge pixels at boundaries are obtained from either large neighboring color differences that are greater than the threshold of the Canny detector or the eight-neighborhood connection to an existing edge pixel. We describe the edge pixels by characteristic color pairs. In an $n \times n$ neighborhood Nb_n of an edge pixel P_e, we find out the two pixels with maximum color difference among all pairs of pixels. Their color values are used for the observation of the color pair across the two sides of the boundary where the edge pixel is located. We denote the color with lower intensity component by $\boldsymbol{cl_L}$ and higher intensity component by $\boldsymbol{cl_H}$ [see Fig. 3(c)]. In RGB space, color values $\boldsymbol{cl_L}$ and $\boldsymbol{cl_H}$ both have three dimensions. If the boundary belongs to a text character or string, $\boldsymbol{cl_L}$ and $\boldsymbol{cl_H}$ represent the colors of text and attachment surface, respectively. Moreover, the coordinates and of the edge pixel P_e are used for the observation of spatial positions. Then an observation vector x of the edge pixel icean be defined by cascading the color values of the two pixels with maximum color difference in the neighborhood and the spatial coordinates of the central edge pixel. To normalize the dimensions of color pair and spatial position observation, the coordinates sp_x and sp_y are extended into three dimensions as $\boldsymbol{sp_x} = \{sp_x, sp_x, sp_x\}$ and $\boldsymbol{sp_y} = \{sp_y, sp_y, sp_y\}$. Thus edge pixel P_e is described by an observation vector $x = [\boldsymbol{cl_L}, \boldsymbol{cl_H}, \boldsymbol{sp_x}, \boldsymbol{sp_y}]$, which is a 12-D point in the observation space.

To extract text boundaries from scene images, we cluster the observation points of the edge pixels into several groups such that the edge pixels with similar color pairs and spatial positions are assigned into identical boundary layers. In this process, GMM is employed to analyze the distributions of observation points of edge pixels. At first, K, means clustering is applied to calculate K centers of observation points, which are used as initial means $\mu_i(1 \leq i \leq k)$ of the Gaussian mixture distributions. Then the corresponding K variances $\sigma_i(1 \leq i \leq k)$ are calculated from the means of observation points. Thus, we can initialize a group of Gaussian distributions. By labeling each of them with a weight, the expectation of GMM is represented by (1)

$$P(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{i=1}^{K} w_i P_i(x|\mu_i, \sigma_i) \tag{1}$$

where $\mathbf{x}$ represents the observation points, w_i represents the weights of the ith Gaussian distribution in the mixture set, and μ_i and σ_i represent the mean and variance of the ith Gaussian distribution, respectively. Next, over the observation points of edge pixels, EM algorithm is applied to obtain the maximum likelihood estimate of the GMM parameters, including weights, means, and variances of the K Gaussian distributions. In the EM process, the GMM parameters are iteratively updated by (2) from their initial values derived by K-means clustering

$$\begin{aligned} w_i^{t+1} &= \frac{1}{N}\sum_{j=1}^{N} p_i(x_j|\mu_i^t, \sigma_i^t) \\ \mu_i^{t+1} &= \frac{\sum_{j=1}^{N} p_i(x_j|\mu_i^t, \sigma_i^t)x_j}{\sum_{j=1}^{N} p_i(x_j|\mu_i^t, \sigma_i^t)} \\ \sigma_i^{t+1} &= \frac{\sum_{j=1}^{N} p_i(x_j|\mu_i^t, \sigma_i^t)(x_j - \mu_i^{t+1})^2}{\sum_{j=1}^{N} p_i(x_j|\mu_i^t, \sigma_i^t)} \end{aligned} \tag{2}$$

where N is the number of observation points and t denotes the tth iteration. This iterative update is performed until the log likelihood $\log \prod_{j=1}^{N} P(x_j|\boldsymbol{\mu}, \boldsymbol{\sigma})$ is convergent. Then the boundary layer is built from each of the K Gaussian distributions under the parameters derived by EM. For an edge pixel, if it generates the maximum likelihood in the ith Gaussian distribution, it will be assigned into the ith boundary layer B_i by (3)

$$\begin{aligned} \mathbf{X_i} &= \{x_j \in \mathbf{x} | \forall k \in [1, k], p_i(x_j|\mu_i, \sigma_i) \geq p_k(x_j|\mu_k, \sigma_k)\} \\ B_i(p) &= \begin{cases} 1, & \text{if } x_p \in \mathbf{X_i} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

Furthermore, the expected value μ_i of the ith Gaussian distribution provides a mean color pair $\{\boldsymbol{cl}_L^i, \boldsymbol{cl}_H^i\}$ to label all edge pixels at the layer B_i.

Fig. 4(a) illustrates three examples of boundary layers after EM-based clustering. Fig. 4(b) presents the corresponding results of regular color reduction. As shown in the first two examples, color reduction fails to completely extract text from background outliers, leaving the boundaries of tree and plane on the boundary layer of text. It is because color reduction does not employ the spatial position difference between text and background outliers. In the third example, color reduction fails to extract the words "TESCO" and "LIFE," but fuses them into the attachment surface in similar color. Color reduction quantizes the dominant colors through statistics of absolute color values, so neighboring objects with color difference lower than some threshold are very probably regarded as a complete object. However, our proposed clustering algorithm quantizes the color pairs around the edge pixels instead of absolute color values at all pixels. A color pair can be successfully extracted as long as it covers enough edge pixels to compose its boundary layer, even though the difference between the pair of colors is small.

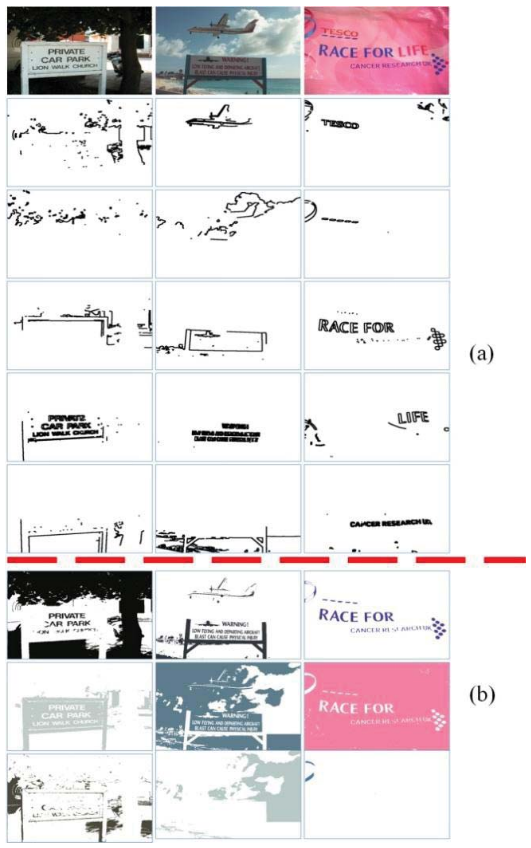


Fig. 4. (a) Examples of boundary layers from scene images, edge pixels with similar color pairs and spatial positions are grouped into the same layer. Boundaries at different veridical positions are assigned into different boundary layers because of y-coordinate spatial information in the clustering process. (b) Results of color reduction based on the clustering of absolute color values, where white region represents background, and color region consists of the pixels that are grouped to the layer.

Since all the involved text strings in our experiments are horizontal, the spatial positions of text boundaries can be estimated only in y-coordinates. Thus the dimension of an observation point is reduced to 9 as $x = [\boldsymbol{cl_L}, \boldsymbol{cl_H}, \boldsymbol{sp_y}]$. In our experiments on scene images, the number of Gaussian mixtures is $K = 5$, which generates the best results of BC. If K is too small, text boundary cannot be effectively extracted from complex background. If K is too large, the algorithm will lose the tolerance to color variation within a character

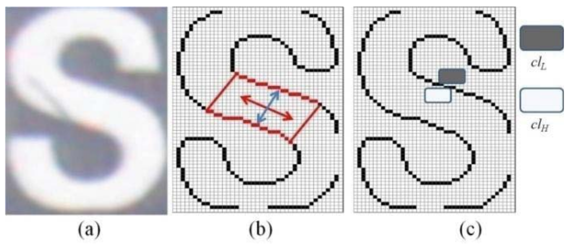


Fig. 5. (a) Image patch of a text character. (b) Stroke is marked by a red character boundary, and the red arrow denotes stroke orientations and blue arrow indicates the stroke width. (c) Color assignment based on the mean color pair $\{\boldsymbol{cl_L}, \boldsymbol{cl_H}\}$ in the current boundary layer.

or string. In that case, text boundary is probably broken into several fragments and assigned into different boundary layers.

III. Stroke Segmentation

Text boundaries provide preliminary clues of string locations and character structure. In [14], character boundaries, as a set of connected edge pixels, are directly used for structural analysis and text segmentation. However, due to the background interferences in scene images, text boundaries are probably broken into tiny segments or connected into the boundary of a nontext background object. To localize text accurately, we use the mean color pair in each boundary layer to label a set of connected components as candidate text characters by color assignment. The process of character labeling uses stroke as basic unit because character is composed of strokes with similar width and different orientations.

Here we define a stroke as a connected image region with uniform color and half-closed boundary, which keeps consistent distance in one direction while stays extensible in the perpendicular direction. As shown in Fig. 5(b), this consistent distance is defined as stroke width and the extensible direction is defined as stroke orientation. We apply color assignment and structural analysis to obtain the strokes of text characters and organize them into aligned connected components.

Color assignment assigns each pixel to the closer color value of the mean color pair $\{\boldsymbol{cl_L}, \boldsymbol{cl_H}\}$ on a boundary layer by (4), where $\mathbf{c}$ denotes the original color of a pixel in RGB space

$$\mathbf{c}^* = \begin{cases} \boldsymbol{cl_L} & \|\boldsymbol{cl_L} - \mathbf{c}\| \leq \|\boldsymbol{cl_H} - \mathbf{c}\| \\ \boldsymbol{cl_H} & \|\boldsymbol{cl_L} - \mathbf{c}\| > \|\boldsymbol{cl_H} - \mathbf{c}\|. \end{cases} \quad (4)$$

In color assignment, the pixels at strokes and those at attachment surfaces should be assigned different colors from the mean color pair [see Fig. 5(c)]. Thus text characters composed of strokes can be segmented in the form of connected components. However, attachment surfaces and background outliers are also segmented out during color assignment. To eliminate the background interferences, we combine color assignment with structural analysis of stroke boundary and propose two algorithms of color assignment.

A. Direct Color Assignment (DCA)

The pixels far from object boundaries should be skipped during color assignment, because they are neither text nor an attachment surface. To predict whether a pixel P is located at

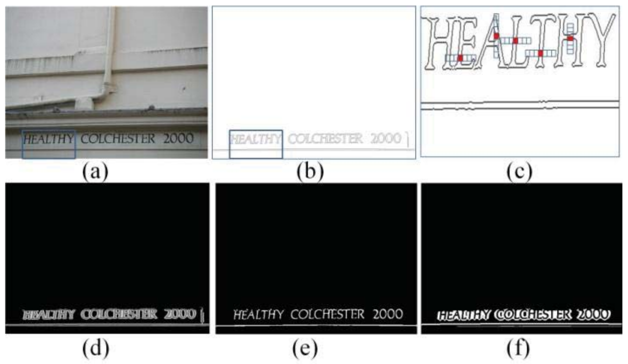


Fig. 6. (a) Scene image. (b) One of the boundary layers of the scene image. (c) Some examples of neighboring windows, where the red points denote the central pixels. (d) Resulting map of DCA, where pixels inside the strokes and the attachment surfaces are labeled by the mean color pair of current boundary layer, and the skipped background pixels are black. (e) Segmented strokes. (f) Segmented attachment surfaces. Both (e) and (f) are binary maps obtained from the binarization of (d).

the neighborhoods of object boundaries, we define a constraint based on a pair of neighboring windows at P in (5)

$$\left(\sum_{t=-R_s}^{R_s} B(x+t, y)\right) \times \left(\sum_{t=-C_s}^{C_s} B(x, y+t)\right) = 0 \quad (5)$$

where B represents a boundary layer in which edge pixel is set as value 1 and background pixel is set as value 0, and x and y represent the coordinates of a central pixel P at boundary layer B. A horizontal window $(2R_s + 1) \times 1$ and a vertical window $1 \times (2C_s + 1)$ are respectively generated at P, as shown in Fig. 6(c), where P is denoted by the red points. If P satisfies the constraint in (5), no edge pixel falls into its two neighboring windows. Thus P will be regarded as background pixel and be skipped during color assignment. In our experiments, we set and $R_s = 6$ and $C_s = 4$.

This algorithm is effective on extracting text from a simple background without many natural objects, that is, the first category of scene images as introduced in Section I. The two sliding windows generate valid regions for color assignment in each boundary layer. Within the valid regions, the pixel belongs to either text stroke or attachment surface. Thus both of them can be extracted as foreground connected components by the binary labeling of color pairs directly, as shown in Fig. 6. The task of distinguishing text from attachment surface will be finished by layout analysis (see Section IV-A).

B. Inferred Color Assignment (ICA)

In the scene images where text characters and nontext objects are not isolated apart from each other, it is difficult to derive a valid region without any boundaries of background outliers. In this case, naive binarization based on only the color pair cannot segment strokes and attachment surfaces, because DCA usually fails to completely separate text boundaries from nearby background interferences in similar colors. As shown in Figs. 7(b) and 8(a), the boundaries of the text “HFC” are connected to the boundaries of background objects due to the interference of lamp shadow on the attachment surface.

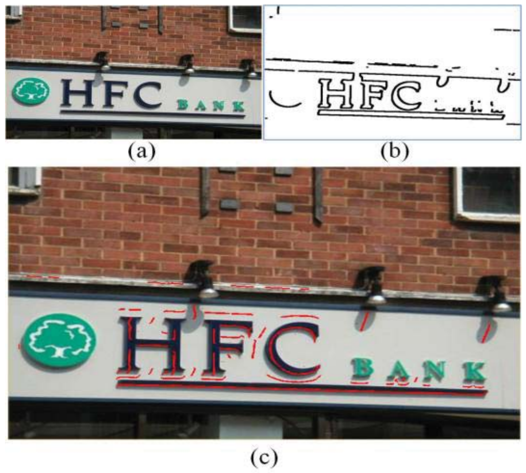


Fig. 7. (a) Original scene image. (b) Boundary layer of the image. (c) PORs marked in red display the preliminary stroke localization in the boundary layer.

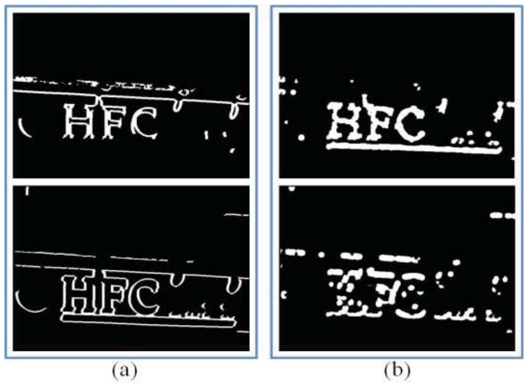


Fig. 8. Binary maps of segmented strokes in the top and segmented attachment surfaces in the bottom, resulting from (a) DCA and (b) ICA.

To skip these unexpected pixels, we propose an algorithm of ICA, in which the gradient cohesion and width consistency of character strokes are employed to localize pixels inside the body of strokes.

We extract a set of pixels to compose the stroke central lines, points of reference (PORs). We use horizontal and vertical probes to detect the PORs at each boundary layer. For a pixel R, two horizontal rays are generated to probe toward left and right, respectively, until edge pixels B_l and B_r are encountered within a range. Pixel R will be labeled as a POR if B_l and B_r satisfy the following constraints: 1) gradients at B_l and B_r have approximately equal magnitude and opposite orientations and 2) length difference between the two line segments and does not exceed 2 pixels, which ensures the approximate lengths of the two line segments and the proper number and distribution of PORs for further feature extraction. Then the length sum of the two line segments is used as the width of the stroke where R is located. If R is not a POR under the horizontal probe, two vertical rays will be generated to repeat the above process toward up and down, and we employ the same constraints in the vertical probe.

Each POR is assigned the closer color value of the color pair by (4). As shown in Fig. 7(c), PORs are distributed at the body of strokes or at the gap of neighboring stroke. Rare PORs exist in the background because most interference boundaries are not satisfied with stroke features. Next, color assignment is performed for all the other pixels according to their spatial relationships with the PORs. For a pixel P_o, we find out its nearest POR P_R. If the length of a line segment P_oP_R is less than the stroke width marked by P_R, the P_o is considered as a pixel within the same stroke as P_R, and we paint it with the color of P_R. After the color assignment, the segmentation results of strokes and attachment surfaces are generated by labeling the two color values of the color pair as foreground, respectively, as shown in Fig. 8(b). Then connected components are obtained from the two resulting binary maps according to PORs.

Compared to the DCA, some background outliers that are incompatible with stroke-width-based text features can be removed in the process of inferred color assignment, as shown in Fig. 8. In our experiments, the framework is respectively equipped with DCA and ICA for performance evaluation.

IV. String Fragment Classification (SFC)

In the process of stroke segmentation, text and attachment surface are extracted as foreground connected components. Layout SFC are further performed to verify text among the connected components. The attachment surface usually appears as background board, and text usually appears in the form of text string, which is a group of aligned character members with similar size and distance between them. We find out the groups of connected components that probably compose text strings. Each group is considered as a fragment of a text string, defined as a string fragment. Then we propose Gabor-based text features to train an SVM-based classifier of string fragments to determine whether a candidate string fragment is text patch or not.

A. Layout Analysis

Layout analysis based on text string focuses on the relationships between characters and their neighboring siblings. We use connected components obtained from stroke segmentation as the basic element to extract the string fragments. Different algorithms of layout analysis are adopted according to the character size. We use character height 12 as the threshold to distinguish large-size and small-size characters.

For the large-size characters that are distant from neighboring siblings, we apply adjacent character grouping [11] to combine sibling connected components based on the structural analysis of sibling characters for layout analysis. This algorithm is applicable to text strings with small number of characters and high resolutions. Adjacent character grouping is designed to extract the string fragments with no less than three characters. To ensure the robustness of our framework, we extend the algorithm by grouping two neighboring connected components under specific conditions. Compared to the adjacent grouping based on three or more characters, two-component grouping is more likely to obtain false-positive string fragments from background outliers. Thus we set more rigorous conditions for two-component grouping as follows.

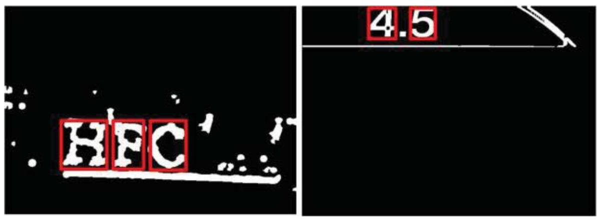


Fig. 9. Some examples of adjacent character grouping, where string fragments are obtained from grouping adjacent characters marked by red boxes.

First, the centroid of a connected component should be inside the horizontal range of the other connected component. If y_o is the y-coordinate of the centroid of one connected component, y_1 and y_2 are the upper side and bottom side of the bounding box of the other connected component, respectively, they should satisfy $y_o \geq y_1 + (y_2 - y_1)/3$ and $y_o \leq y_2 - (y_2 - y_1)/3$. It ensures that the two connected components stay in horizontal alignment. Second, the height ratio of the two connected components should be greater than 0.83 and less than 1.2. The width ratio of the two connected components should be greater than 0.5 and less than 2. It ensures that they have similar sizes. Third, the distance between the two connected components should not exceed twice the width of the wider one, and also it should not be less than 12. The satisfied connected components are grouped together to obtain string fragments, as shown in Fig. 9.

For small-size characters, stroke segmentation usually fails to separate them since they might be integrated into a single connected component, and we have no access to the details of the character structure. In this case, a single connected component is considered as a string fragment directly if its height is less than 12 and width-to-height ratio is greater than 4. But this method will bring in more background interferences because it omits the certification of character alignment. In our experiments, this method is used only in the process of text localization in born-digital images, which will be described in detail in Section V-C.

A string fragment is an image patch with compatible size to accommodate all its candidate character members. To prepare for the classification of the candidate string fragments, we normalize the image patches into a fixed size 48 × 96. In this process, the large-size string fragments obtained from adjacent character grouping are directly scaled into image patches with width 96 pixels and height 48 pixels, and the small-size string fragment from a single connected component should be sliced vertically into overlapped partitions with width-to-height ratio 2:1 and then scaled into image patches with width 96 pixels and height 48 pixels.

B. Training Set

In this section, we will describe the establishment of a training set for learning the string fragment classifier. First, we perform adjacent character grouping [11] on scene images with text information to obtain a set of image patches, in which string fragments are taken as positive samples, and nontext outliers are taken as negative samples. Besides the image patches generated from scene images, we also generate synthetic text characters and string fragments to produce additional positive samples. Also we add image patches of nontext objects and textures that resemble text characters as additional negative samples. We collect around 2000 positive samples and 2000 negative samples, respectively, for training. Next, all these image patches are adjusted to the standard size for training the string fragment classifier. For each patch, if the width-to-height ratio is greater than 1:1 but less than 3:1, it is normalized into width 96 pixels and height 48 pixels. If the width-to-height ratio is greater than 3:1, it is sliced vertically into overlapped patches with width-to-height ratio 2:1, and then normalized into width 96 pixels and height 48 pixels. If the width-to-height ratio of an image patch is less than 1:1, we splice its three copies in vertical to generate a new image patch of width 96 pixels and height 48 pixels.

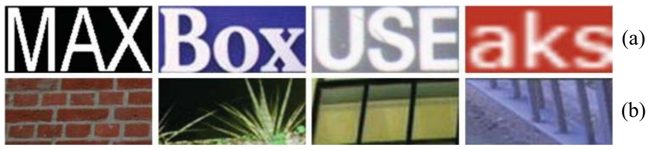


Fig. 10. Examples of (a) positive samples and (b) negative samples in the training set of string fragments.

Based on the training set, Gabor-based text features are extracted to train a classifier of string fragment in an SVM model. To ensure maximum Gabor responses at stroke components, stroke pixel intensity is higher than the background pixel intensity in positive samples. Detailed description will be given in Section IV-C. Fig. 10 presents examples of both positive samples and negative samples from the training set.

C. Gabor-Based Features

To eliminate the false-positive string fragments, we employ Gabor-based features to model the inner structure of text. In previous literatures of text segmentation and text recognition [19], [25], [28], Gabor filter was applied to model text appearances by finding maximum responses on stroke components. Gabor filter can be used to analyze the combinations and distributions of stroke components, which are related to the structural features of text. In this framework, we employ Gabor filter responses to detect out pixels of interest (POI) for text feature extraction.

To model the stroke width and stroke orientation from a pixel-based perspective, Gabor filter is adaptively generated at each pixel of the string fragment for maximum Gabor responses. First, we calculate the edge map and distance transform (DT) map of string fragments. For a pixel P, the DT map can indicate its nearest edge pixel P_e and their distance d_p. If P is located inside a stroke, the line PP_e should be perpendicular to the stroke orientation. Next, we adopt the model of Gabor filter by (6) to calculate a compatible Gabor filter at each pixel, in which the wavelength λ is set as the distance from the nearest edge pixel d_p and the orientation θ is set as the stroke orientation corresponding to perpendicular direction to the line PP_e [see Fig. 11(b) and (c)]. The other parameters are kept constant, $\gamma = 0.2$ and

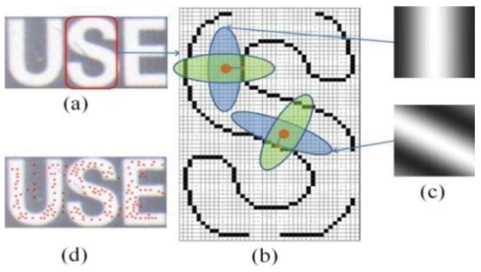


Fig. 11. (a) Image patch of a string fragment. (b) Two compatible Gabor filters (marked in blue) are generated at the two red points according to the stroke orientation and the distance to the nearest edge pixel. Two anticompatible Gabor filters are obtained by $\pi/2$ rotation marked in green. (c) Two compatible Gabor filters. (d) Extracted feature points marked in red, which are distributed inside the body of strokes and in the gap of neighboring strokes.

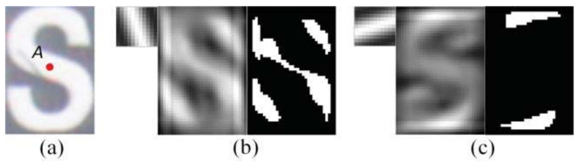


Fig. 12. (a) Image patch of a string fragment with a red pixel A. (b) Compatible Gabor filter at pixel A with corresponding Gabor response and binary threshold map. (c) Anticompatible Gabor filter with corresponding Gabor response and binary threshold map.

$\sigma = 2$. The pixel P is called the source pixel of the Gabor filter. The compatible Gabor filter is expected to produce the maximum Gabor response on the stroke of its source pixel, because it is generated along the stroke orientation without crossing the stroke boundaries

$$g(x, y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$
$$x' = x\cos\theta + y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta. \quad (6)$$

We rotate the compatible Gabor filter by $\pi/2$ at each pixel to obtain an anticompatible Gabor filter and corresponding Gabor response map. The anticompatible Gabor filter perpendicular to stroke orientation stretches across the stroke width segment into the background region. On the contrary to compatible Gabor filter, it is expected to provide minimum filter response at the pixel, as shown in Fig. 12. We calculate the absolute difference between the two Gabor response maps, where the local maximum pixels at the map of absolute difference are extracted as POI, as shown in Fig. 11(d). POI can be considered as the sample point of the stroke. Feature extraction focuses only on the POIs in the feature maps of gradient, stroke distribution, and stroke width. Thus the extracted feature is named as Gabor-based text feature.

The POIs model the inner structure of text by positions and orientations of the strokes where they are located. We make statistics of stroke orientations based on the POIs in the collected training set of string fragments. A block pattern is employed to divide the image patches of both positive samples and negative samples into three horizontal partition regions. By quantizing stroke orientations into eight bins within the range $(-\pi/2, \pi/2]$, we count the number of POIs at each stroke orientation to obtain POI histograms, as shown in Fig. 13. This figure shows that in positive samples of text string fragments, the dominant stroke orientation is $\pi/2$, that is, vertical stroke has the largest frequency of occurrences in text characters and strings. Also, the stroke orientations in positive samples are approximately symmetrical with respect to the vertical direction, because most characters in print patterns are composed of closed and symmetrical strokes. In contrast, the distribution of stroke orientations in negative samples is more uniform without dominant orientation. Furthermore, in positive samples, statistical results vary among different partition regions. We can see that the vertical stroke is the most dominant in middle partition regions. However, negative samples generate similar patterns of histograms in the three partition regions. The pattern of histograms proves that the POI and the block pattern can generate structural features to distinguish text string fragments from nontext outliers.

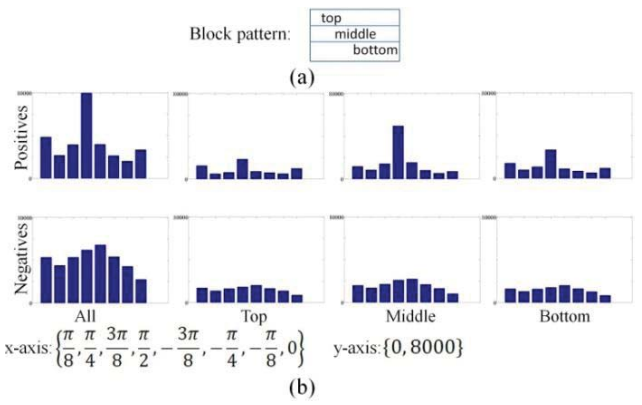


Fig. 13. (a) Block pattern used in the statistics of stroke orientations. (b) Histograms of stroke orientations from partition regions of positive samples (top row) and negative samples (bottom row), where the first column denotes the statistical results of the whole image patches without block pattern partition.

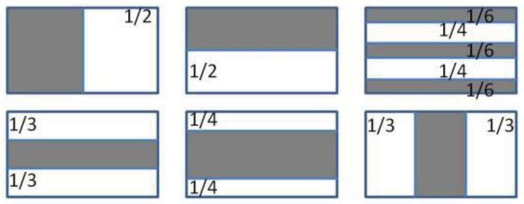


Fig. 14. Six block patterns used for extracting text features, where the side length of the partition region is given by the percentage of the block side.

To extract Gabor-based text features for string fragment classification, we employ six block patterns [10] to provide the maps of image patch partition, including the one presented in Fig. 14. These block patterns are related to the intensity and gradient distributions in image patches of string fragments. They model the inner structure of string fragments by proportionally dividing the image patches into multiple partition regions along horizontal and vertical directions.

The feature values are calculated by the absolute difference between the measurement of POIs in white partition regions

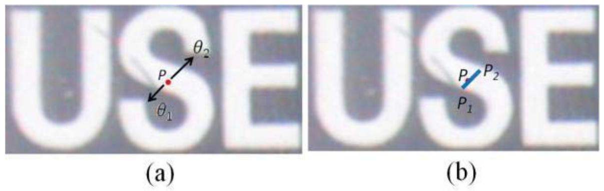


Fig. 15. (a) Probe rays along opposite orientations to search for edge pixels. (b) Two encountered edge pixels and are used to estimate the stroke width of character "S" where feature point is located.

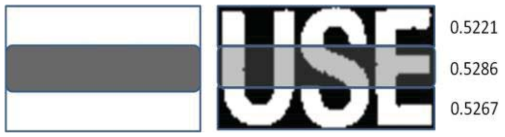


Fig. 16. Stroke distribution estimates by one of the block patterns, and the right values denote the ratio of foreground pixels at corresponding partition regions.

and that in gray partition regions. To balance the feature values, a weight is assigned to each partition region. We design two methods of weight assignment. The first method assigns the same weights to all partition regions of the block patterns. This method is applied to all six block patterns. The second method first ensures the same weight sum for gray partition regions and white partition regions. Then each partition region in the respective group is assigned a weight inversely proportional to its area. Taking the third block pattern in the top row of Fig. 14 for example, the three gray partition regions are assigned weight 0.5/3, and the two white partition regions are assigned weight 0.5/2. This method is not applied to the two block patterns with less than three partition regions, because it provides the same results as the first method. Taking the weights into account, 10 types of block patterns are employed to extract text features from feature maps. For each partition region of a block pattern, the measurement consists of two calculation metrics, which are sum and mean of all the pixel values within this partition region. The 10 block patterns and two calculation metrics are applied to extract text features from the POI-based maps of gradient, stroke width, and stroke distribution. Each feature map can generate $10 \times 2 = 20$ values to compose a feature vector.

1) Gradient: POIs are distributed within the body or gap of strokes, where the gradient magnitudes decrease but gradient orientations are the same as stroke boundaries. The gradient variations should keep consistent at all partition regions of block patterns, if they are obtained from text characters with identical font and size. Horizontal and vertical Sobel operators are applied on the POIs to calculate the gradient values in the two directions. Then we transform them into two feature maps, gradient magnitude and gradient orientation. By using the block patterns and calculation metrics, we generate feature vector in $2 \times 20 = 40$ dimensions on the gradient feature maps of a string fragment.

2) Stroke Width: We propose a ray-probing algorithm to estimate stroke width at each POI. At first, we obtain stroke orientation at a POI P from its compatible Gabor filter. Then, two rays starting from P probe along two directions that are perpendicular to P's stroke orientation until they reach edge pixels P_1 and P_2, respectively, as shown in Fig. 15. The segment length between the two encountered edge pixels is considered as the stroke width across P. We calculate stroke width across each POI to produce a feature map. Next, the block patterns and calculation metrics are applied on the stroke width map to obtain a feature vector of 20-D. Furthermore, to measure the stroke width consistency, Gaussian distribution $<\mu, \sigma>$ is generated from the maximum likelihood of the estimated stroke width at POIs. The coefficient of variance in the form $\sigma/|\mu|$ is used as the feature value of stroke width consistency. To keep consistent feature dimensions with gradient-based features, we extend the single-value variance measure into a vector of 20-D.

3) Stroke Distribution: When the image patch of string fragment is partitioned by horizontal section lines, the strokes of text characters are regularly organized in the line spaces. For example, the handwriting worksheet uses a four-line block pattern to assist novices in tracing the character structure. According to this characteristic, we model the character structure by a stroke distribution of 18 partition regions from the six types of block patterns. First, an image patch of string fragment is binarized by Otsu's method [29]. Then we calculate the ratio between the number of foreground pixels and the total number of pixels in each of the partition regions. The ratio is used as the feature value of stroke distribution (see Fig. 16), and an 18-D feature vector is derived from each string fragment.

By cascading the vectors obtained from the feature maps of gradient, stroke width, and stroke distribution, a 98-D feature vector is generated for each candidate string fragment, which can be considered as a point in the feature space. We calculate feature vectors of all the samples in the training set and normalize the feature values in each dimension into the range. Then a linear SVM model is applied to generate a classifier of string fragment.

V. Experiments

We carry out experiments to evaluate the performance of the Gabor-based features in SFC and the performance of our framework in text localization on scene images. In addition, we localize text regions on born-digital images, broadcast video images, and images of handheld objects captured by blind persons to show the robustness of our framework in dealing with multiple background outliers.

A. Evaluation of Gabor-Based Features

Gabor-based features are evaluated to train a robust classifier of string fragments. The five features of string fragment include gradient magnitude, gradient orientation, stroke width, stroke distribution, and stroke width consistency measure. We estimate the performance of each feature in the SVM model, and obtain the most robust classifier of string fragments.

Experiments are performed on two groups of image patches. First, the feature is evaluated within the collected training set

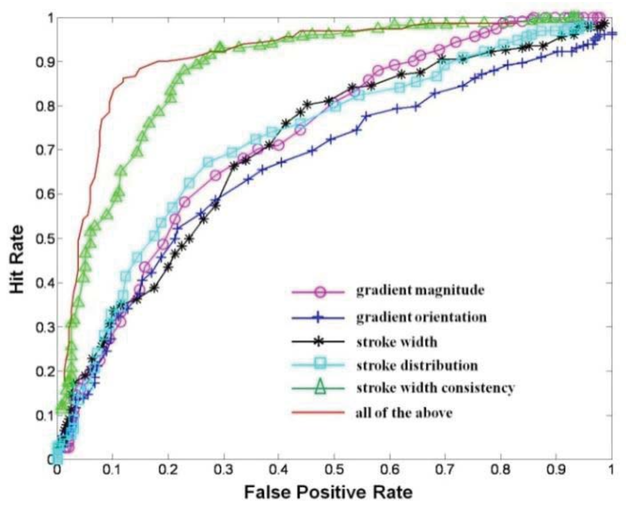


Fig. 17. Evaluation results of the five Gabor-based features on the collected training set of string fragments. Here, hit rate is the ratio of correctly classified samples in the positive set, and false-positive rate is the ratio of incorrectly classified samples in the negative set.

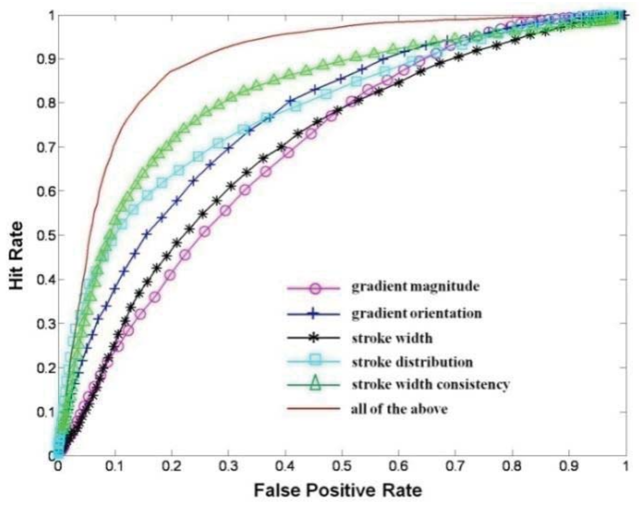


Fig. 18. Evaluation results of the five Gabor-based features on image patches obtained from ICDAR 2003 Robust reading dataset of scene images. Hit rate is the ratio of correctly classified samples in the positive set, and false-positive rate is the ratio of incorrectly classified samples in the negative set.

of string fragments. The 2000 positive samples and 2000 negative samples are equally divided into two subsets, respectively, one of which is used for classifier training and the other is used for evaluation. Next, the classifier is learned from the whole training set, and then evaluated on around 18 000 image patches, which are obtained from layout analysis on the scene images of ICDAR 2003 Robust reading dataset. We have manually labeled the text patches and nontext patches for classifier evaluation.

Figs. 17 and 18 illustrate the evaluation results of the two experiments, respectively, where hit rate represents the ratio of correctly classified samples in the positive set, and false-positive rate represents the ratio of incorrectly classified samples in the negative set. The two figures demonstrate that stroke width consistency is more robust than the other features of text. Gradient magnitude and stroke width achieve comparable performance with stroke distribution and gradient orientation in the first experiment, but they become inferior in the second experiment. It is inferred that gradient orientations and stroke distributions are normalized into the ranges $(\pi, \pi]$ and [0,1], respectively, so they are more robust to the variations of large number of samples in the second experiment. In our framework, all the features are combined to model string fragments, because both figures show that the best performance of SFC is achieved when combining all the features.

B. Evaluation of Text Localization on Scene Images

We evaluate the performance of our proposed framework on two benchmark datasets of scene images, ICDAR 2003 Robust reading dataset [30] and ICDAR 2011 Robust reading dataset [31]. Both of them were collected for robust reading competitions and annotated text regions. ICDAR 2003 Robust reading dataset contains around 500 scene images and 2258 ground truth text regions in total. In our experiments, the scene images contain nontext or only a single character is excluded. Thus 487 scene images are used for performance evaluation. The involved image sizes range from 640×480 to 1600×1200. ICDAR 2011 Robust reading dataset contains 229 scene images with 848 ground truth text regions in total. We evaluate the framework on 228 images containing text strings with no less than two character members. The image size ranges from 422×102 to 3888×2592. The proposed framework is applied on the above datasets for text localization. The localization processes are carried out in each scene image and its inverse image, and the results are combined to calculate the localized text regions. In all our experiments on text localization, the involved scene image is normalized into longer side 640 and shorter side 480 (640×480) when length ratio of the longer side and the shorter side is less than 2, and it is normalized into 640×240 when the length ratio is greater than or equal to 2. Evaluation results are obtained from the comparisons between a group of localized text regions and ground truth text regions. We denote their overlaps as the hit regions which are the correctly extracted text regions. We define the area of a text region as the number of pixels in the region. Based on these measures, *precision* is defined as the ratio between the area of hit regions and the area of the detected regions, and *recall* is defined as the ratio between the area of hit regions and the area of the ground truth regions. Then they are combined by harmonic mean to obtain f-measure as in (7)

$$f = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{7}$$

We perform two rounds of text localization by using DCA and ICA, respectively, in the step of stroke segmentation. The results are presented in Table I. The ICA framework achieves better recall but lower precision. ICA algorithm filters out the boundaries unsatisfied with stroke structure, but cannot eliminate the background outliers completely. More text characters are separated from background interference by structural analysis of stroke boundary in ICA, so the recall is improved. However, stroke structure also exists in nontext objects,

TABLE I
COMPARISON BETWEEN OUR FRAMEWORK AND THE TEXT LOCALIZATION ALGORITHMS PRESENTED IN [7], [11], [32], AND [33] ON THE ROBUST READING DATASE

	Precision	Recall	f-measure
Ours (DCA)	**0.73**	**0.67**	**0.66**
B. Epshtein	0.73	0.60	0.66
Ours (ICA)	**0.65**	**0.74**	**0.64**
H. Becker	0.62	0.67	0.62
C. Yi	0.71	0.62	0.62
A. Chen	0.60	0.60	0.58
Ashida	0.55	0.46	0.50
HWDavid	0.44	0.46	0.45
Wolf	0.30	0.44	0.35
Q. Zhu	0.33	0.40	0.33

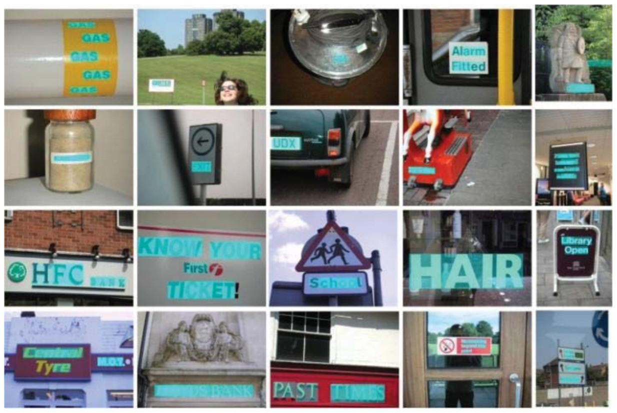


Fig. 19. Example results of text localization in the ICDAR 2003 Robust reading dataset, where the text regions are marked by cyan boxes.

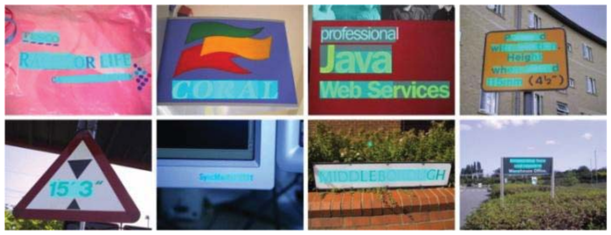


Fig. 20. Some example results of text localization in the ICDAR 2011 Robust reading dataset, where the text regions are marked by cyan boxes.

and the survived boundaries and the corresponding connected components in ICA usually possess similar appearance and alignment. Thus, more false-positive string fragments from layout analysis lower the precision.

Fig. 19 illustrates some examples of text localization where the text regions are marked in cyan boxes. Table I presents the performance comparisons between our framework and the localization algorithms involved in ICDAR Robust Reading Competition [32], [33]. It shows that the proposed framework outperforms most previous localization algorithms.

The same experimental process is performed on the ICDAR 2011 Robust reading dataset to evaluate our framework with DCA and ICA, respectively. The results are obtained by using the same evaluation measures, as presented in Table II. Some detected text regions are presented in Fig. 20 with cyan boxes.

Furthermore, we evaluate the performance of BC and SFC respectively on this dataset. First, adjacent character grouping

TABLE II
RESULTS OF TEXT LOCALIZATION ON THE ICDAR 2011 ROBUST READING DATASET OF SCENE IMAGES

	Precision	Recall	f-measure
Ours (DCA)	**0.81**	**0.72**	**0.71**
Ours (ICA)	**0.67**	**0.80**	**0.68**
ACG	0.63	0.79	0.64
BC + ACG	0.66	0.78	0.66
ACG + SFC	0.72	0.77	0.69

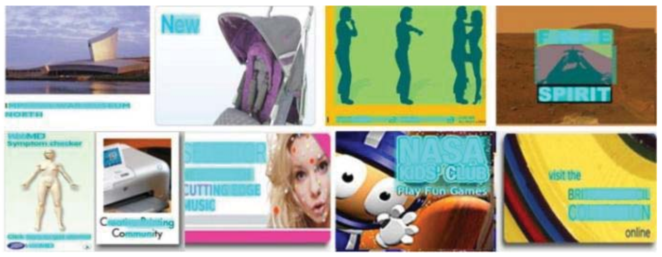


Fig. 21. Example results of text localization in the born-digital images.

(ACG) [11] is evaluated independently on the edge images to localize aligned connected components or boundaries with similar sizes and generate string fragments by grouping. Then we apply BC and SFC as preprocessing and postprocessing of ACG respectively. Boundary clustering categorizes boundaries with similar color pairs to filter out false combination of adjacent components in ACG, and filters out the false positive string fragments generated by ACG. The evaluation results are shown in Table II. Many background interferences are filtered out in BC and SFC, so they improve the precision. However, some text characters with nonregular patterns or incomplete boundaries might be also removed. Thus the recall has a little decline.

C. Evaluation on Born-Digital and Broadcast Video Images

We further evaluate our framework to extract text information from born-digital images and broadcast video images. Born-digital images are electrical documents with colorful captions and illustrations. Mostly they exist in web pages, book covers, and posters. In born-digital images, text characters and strings are more colorful, so the initial number of Gaussian mixtures in BC is set as $K = 7$. Besides, born-digital images have higher-frequency of occurrences of text and smaller character sizes than scene images. Thus, in layout analysis, we consider some connected components directly as string fragments and slice the corresponding image patches vertically to overlapped partitions with width-to-height ratio 2:1.

A dataset of born-digital images is released for ICDAR 2011 Robust Reading Competition [34]. It contains 420 born-digital images with ground truth text regions. The average image size is around 352×200. We evaluate our framework by using the same measures on this dataset. The framework with DCA generates precision 0.64, recall 0.67 and f-measure 0.61, and the framework with ICA generates precision 0.55, recall 0.70 and f-measure 0.56. Fig. 21 presents some examples of localized text regions in born-digital images.

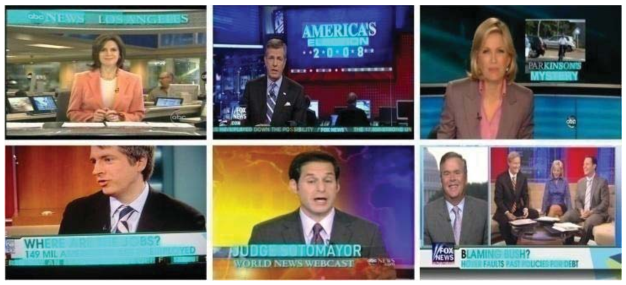


Fig. 22. Example results of text localization in broadcast video images.

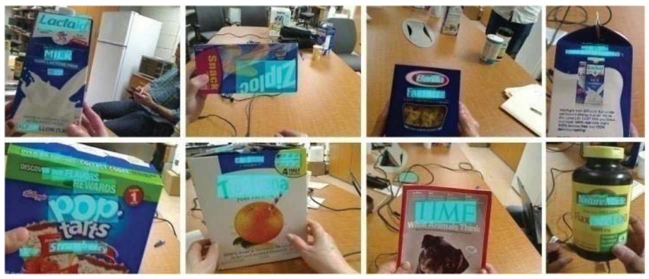


Fig. 23. Example results of localized text regions in cyan from the blind-captured images of text captions.

Moreover, our framework is evaluated on broadcast video images. In most video images, text serves as titles and captions to introduce the content of television programs. It is distributed on the top or bottom of the screen. The characters and strings also have the features of bigram color uniformity, stroke width consistency, and character alignment. Different from scene images, most text information in broadcast video image is subsequently added for audience reading, so they generally encounter fewer background interferences and pattern variations. Fig. 22 depicts some results of localization in broadcast video images.

D. Evaluation on Scene Images of Blind-Captured Objects

The proposed framework can be applied to produce reading-assistant devices to help blind people to distinguish objects in their hands. Most reading-assistant devices, for example, voice-reading pen, require manual localization of text regions in documents or books, because the integrated OCR cannot automatically find out text regions without background outliers. Thus our framework can be employed to carry out the challenging task.

To evaluate the performance of blind assistance, we collected a dataset of text captions on objects in hands. Ten blind persons are recruited to test thirteen objects (grocery boxes, medical bottles, book covers, etc.,) by wearing a camera attached in a pair of sunglasses to capture the handheld objects. In total, we collected 112 blind-captured images with 324 ground truth text regions. These labeled text regions only cover the headings or subheadings in the objects since they provide sufficient information to blind people what the objects in their hands are. Some examples of localized text regions by the proposed framework are shown in Fig. 23.

By using the same evaluation measures used in the above experiments, we obtain precision 0.52, recall 0.62, and f-measure 0.52 on this dataset. Furthermore, we define that a ground truth text region is hit if three-quarter of its area is localized, and a localized text region is hit if half of its area is within the ground truth. In this experiment, our framework hits 164 of the 311 localized regions and 195 of the 324 ground truth regions. The text regions are then input into an OCR system for recognition. The recognized text codes are displayed to blind users as audio outputs. From the above experimental results, we find that the precisions of text localization in born-digital images and blind-captured object images are lower than that in scene images. A reason is that the two types of images usually have lower image resolutions and more compact distributions of text and background objects.

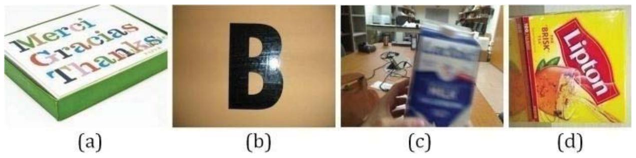


Fig. 24. Scene images where our framework fails to localize text regions. (a) Multicolored character and nonhorizontal string. (b) Text string composed of single character. (c) Low resolution. (d) Nonhorizontal string.

E. Some Limitations of the Proposed Framework

The proposed framework is based on color uniformity and horizontal alignment of text strings with more than 2 characters, so it cannot handle a text string with nonuniform colors, single character or text string whose angle with the horizontal is larger than 20 degrees. In addition, the framework requires enough resolution of the text to be localized. The characters and strings cannot be too small or too blurred. Fig. 24 depicts some challenging scene images in which the framework fails to localize text regions accurately.

In our proposed framework, the involved constraints, assumptions, and parameters are all designed for general text appearance and structure in most natural scenes. We would make the framework be more adaptive to those specific and challenging situations.

VI. Conclusion

In this paper, we designed a novel framework to localize text regions under complex background and multiple text patterns. To eliminate background outliers and model text structure, three steps were involved in the localization process, which were BC, stroke segmentation, and string fragment classification. Text in scene images was modeled by pixel-level bigram color uniformity, character-level stroke structure, and string-level text layout. In each step, we presented novel algorithms to extend and integrate the common features related to outer appearance and inner structure of text. Traditional color-based pixel clustering was transformed into color-pair-based BC, and the character boundaries of text string in similar color were grouped into identical boundary layers. Two algorithms were designed for the structural analysis of stroke component to extract character candidates in each boundary layer. Then we performed layout analysis on the candidate characters to calculate the fragments of text strings. An SVM-based classifier of string fragments was learned from Gabor-based features to filter out the false-positive string fragments obtained from layout analysis. The experiments showed that the proposed framework was able to localize text regions among various background outliers and text patterns, including scene images, born-digital images, broadcast video images, and blind-captured object images. In performance evaluation on benchmark dataset, the framework outperformed state-of-the-art localization algorithms in precision and recall.

In the future work, we will design a more sophisticated algorithm to model the structure of text characters and strings. We will also extend the framework to localize nonhorizontal text strings in deformed surfaces, and design word recognition algorithm to read text information from text regions.

References

[1] T. M. Breuel, "The OCRopus open source OCR system," in *Proc. IS&T/SPIE 20th Ann. Symp.*, 2008, pp. 0F1–0F15.

[2] R. Smith, "An overview of the tesseract OCR engine," in *Proc. Int. Conf. Document Anal. Recogn.*, 2007, pp. 629–633.

[3] N. Nikolaou and N. Papamarkos, "Color reduction for complex document images," *Int. J. Imaging Syst. Technol.*, vol. 19, no. 1, pp. 14–26, 2009.

[4] X. Chen, J. Yang, J. Zhang, and A. Waibel, "Automatic detection and recognition of signs from natural scenes," *IEEE Trans. Image Process.*, vol. 13, no. 1, pp. 87–99, Jan. 2004.

[5] C. Mancas-Thillou and B. Gosselin, "Spatial and color spaces combination for natural scene text extraction," in *Proc. IEEE Conf. Image Process.*, Oct. 2006, pp. 985–988.

[6] Y. Song, A. Liu, L. Pang, S. Lin, Y. Zhang, and S. Tang, "A novel image text extraction method based on K-means clustering," in *Proc. 7th Int. Conf. Comput. Inform. Sci.*, 2008, pp. 185–190.

[7] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in nature scenes with Stroke width transform," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jun. 2010, pp. 2963–2970.

[8] V. Dinh, S. Chun, S. Cha, H. Ryu, and S. Sull, "An efficient method for text detection in video based on stroke width similarity," in *Proc. Asian Conf. Comput. Vision*, 2007, pp. 200–209.

[9] K. Subramanian, P. Natarajan, M. Decerbo, and D. Castanon, "Character-Stroke detection for text-localization and extraction," in *Proc. Int. Conf. Document Anal. Recogn.*, 2005, pp. 33–37.

[10] X. Chen and A. Yuille, "Detecting and reading text in natural scenes," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jun.–Jul. 2004, pp. 366–373.

[11] C. Yi and Y. Tian, "Text string detection from natural scenes by structure-based partition and grouping," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2594–2604, Sep. 2011.

[12] Y. Zheng, H. Li, and D. Doermann, "A parallel-line detection algorithm based on HMM decoding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 777–792, May 2005.

[13] H. Anoual, S. Elfkihi, and A. Jilbab, "Features extraction for text detection and localization," in *Proc. 5th Int. Symp. I/V Commun. Mobile Netw.*, 2010, pp. 1–4.

[14] T. Kasar, J. Kumar, and A. Ramakrishnan, "Font and background color independent text binarization," in *Proc. 2nd Int. Workshop Camera-Based Document Anal. Recogn.*, 2007, pp. 3–9.

[15] W. Kim and C. Kim, "A new approach for overlay text detection and extraction from complex video scene," *IEEE Trans. Image Process.*, vol. 18, no. 2, pp. 401–411, Dec. 2009.

[16] C. Liu, C. Wang, and R. Dai, "Text detection in images based on unsupervised classification of edge-based features," in *Proc. Int. Conf. Document Anal. Recogn.*, vol. 2, pp. 610–614, Aug.–Sep. 2005.

[17] M. Lyu, J. Song, and M. Cai, "A comprehensive method for multilingual video text detection, localization, and extraction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 2, pp. 243–255, Feb. 2005.

[18] T. Phan, P. Shivakumara, and C. L. Tan, "A Laplacian method for video text detection," in *Proc. 10th Int. Conf. Document Anal. Recogn.*, 2009, pp. 66–70.

[19] M. Wan, F. Zhang, H. Cheng, and Q. Liu, "Text localization in spam image using edge features," in *Proc. Int. Conf. Commun. Circuits Syst.*, 2008, pp. 838–842.

[20] S. M. Hanif and L. Prevost, "Text detection and localization in complex scene images using constrained AdaBoost algorithm," in *Proc. 10th Int. Conf. Document Anal. Recogn.*, Jul. 2009, pp. 1–5.

[21] S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. D. Joshi, "Text extraction and document image segmentation using matched wavelets and MRF model," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2117–2128, Aug. 2007.

[22] W. Pan, C. Suen, and T. Bui, "Script identification using steerable Gabor filters," in *Proc. Int. Conf. Document Anal. Recogn.*, Aug.–Sep. 2005, pp. 883–887.

[23] M. Shi, Y. Fujisawab, T. Wakabayashia, and F. Kimura, "Handwritten numeral recognition using gradient and curvature of gray scale image," *Pattern Recogn.*, vol. 35, no. 10, pp. 2051–2059, 2002.

[24] C. Jung, Q. Liu, and J. Kim, "Accurate text localization in images based on SVM output scores," *Image Vision Comput.*, vol. 27, no. 9, pp. 1295–1301, 2008.

[25] A. K. Jain and S. Bhattacharjee, "Text segmentation using Gabor filters for automatic document processing," *Mach. Vision Appl.*, vol. 5, no. 3, pp. 169–184, 1992.

[26] L. Li and M. K. H. Leung, "Integrating intensity and texture differences for robust change detection," *IEEE Trans. Image Process.*, vol. 11, no. 2, pp. 105–112, Feb. 2002.

[27] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986.

[28] J. Weinman, E. Leanred-Miller, and A. Hanson, "Scene text recognition using similarity and a lexicon with sparse belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 10, pp. 1733–1746, Oct. 2009.

[29] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

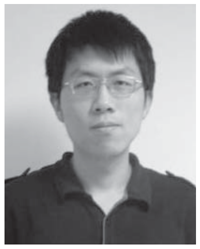
[30] *ICDAR2003 Robust Reading Competition*. (2003) [Online]. Available: http://algoval.essex.ac.uk/icdar/Datasets.html

[31] *ICDAR2011 Robust Reading Competition*. (2011) [Online]. Available: http://robustreading.opendfki.de/wiki/SceneText

[32] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "ICDAR 2003 robust reading competitions," in *Proc. Int. Conf. Document Anal. Recogn.*, 2003, pp. 682–687.

[33] S. M. Lucas, "ICDAR 2005 text locating competition results," in *Proc. Int. Conf. Document Anal. Recogn.*, 2005, pp. 80–84.

[34] *ICDAR2011 Robust Reading Competition* (2011) [Online]. Available: http://www.cvc.uab.es/icdar2011competition/



Chucai Yi (S'12) received the B.S. and M.S. degrees from the Department of Electronic and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2007 and 2009, respectively. He is currently pursuing the Ph.D. degree in computer science from the Graduate Center, City University of New York, New York.

His current research interests include text detection and recognition in natural scene images, object recognition, image processing, and machine learning.

Yingli Tian (M'99–SM'01) received the B.S. and M.S. degrees from Tianjin University, Tianjin, China, in 1987 and 1990, and the Ph.D. degree from the Chinese University of Hong Kong, Hong Kong, in 1996.

She was a Faculty Member with the National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Beijing, China. She joined Carnegie Mellon University, Pittsburgh, PA, in 1998, where she was a Post-Doctoral Fellow with the Robotics Institute. She was a Research Staff Member with IBM T. J. Watson Research Center, Yorktown Heights, NY, from 2001 to 2008. She is currently an Associate Professor with the Department of Electrical Engineering, City College of New York and the Department of Computer Science, Graduate Center, City University of New York. Her current research interests include a wide range of computer vision problems, such as motion detection and analysis, assistive technology, human identification, facial expression analysis, and video surveillance.