General Text-Chunk Localization in Scene Images using a Codebook-based Classifier

Bowornrat Sriman^a, Lambert Schomaker^a, Potchara Pruksasri^b

^aALICE Institute, University of Groningen, Nijenborgh 9, 9747 AG, Groningen, The Netherlands

^bFaculty of Informatics, Mahasarakham University, Mahasarakham 44150, Thailand

B.Sriman@rug.nl, L.Schomaker@ai.rug.nl, potchara.p@msu.ac.th

Abstract

Text localization is a main portal to character recognition in scene images. The detection of text regions in an image is a great challenge. However, many locating methods use a bottom-up scheme that consumes relatively high computation to identify the text regions. Therefore, this paper presents a novel text localization in scene images using a top-down scheme, with a faster computation. While benchmark datasets are usually based on segmented words, there are also script types where character strings are continuous, such as Thai. Models of general text-chunk and non-text have been produced using the SIFT algorithm. Furthermore, we present a codebook-based classifier, which significantly enhances the detection performance. The proposed method has been evaluated for the text-detection and classifying processes with the TSIB (continuous-word language). The results show that our proposed technique improves the text localization particularly for a continuous-word script (Thai) which is rich in ornaments. The proposed technique is also beneficial for locating the text of the ICDAR dataset even though there are many text lines in an image.

Keywords: Text localization, histogram classifier, text and non-text codebooks, continuous-word script, SIFT.

1. Introduction

Today, mobile devices significantly influence daily human life. They are not only used for voice communication but also serve multimedia such as video streaming and images to people. New challenges in image processing such as face detection, visual landmark recognition for navigation, object detection, and text

recognition then arise for the scene images that are captured by mobile devices. Text localization is one of the biggest challenges to be accomplished. Many algorithms have been proposed to conquer the challenge, for example, Conditional Random Field (CRF) [1], [2], connected component [3], [4] and stroke width transform [5]. However, they have not fully succeeded because of different difficulties shown in several studies [6].

Proposed text localization methods are usually based on Western languages, which are punctuated with dots, and all the words in a sentence are separated by spaces. Also, most Western scripts have no tone accents in a word, but some lowercases in English present a dot on the top (i, j) and some have a line that is drawn up/down (f, h, k, l, t, p, q, y, g). Existing localization methods perform quite effectively in detection but using them for continuous-word script such as Thai consumes relatively high computation.

Thai sentences do not have spaces between words. Space is only used to separate sentences, or words and numbers or symbols. Also, Thai alphabet structures are different from other languages scripts because of complex geometric shapes, including lines (vertical/horizontal/tilted), circles and curves. Some letters have crossed lines and branched points. Moreover, some of them look very similar to each other. Finally, a Thai word is composed of consonants, vowels, and tones that can be four vertical zone levels in the written line, which can be easily overlapped between the upper line and lower line. Then, this raises the question of how to find the letters of words in a line of the sentence.

Continuous-word script detection in natural scene image is becoming more interesting and challenging, especially in Asian countries. A model-based text detection is a potential technique to locate character strings in scene

images. Object histogram classifiers or codebooks could be used in the text localization process. However, to acquire a good performance, producing models and relevant detection methods are crucial to solving the problem.

Furthermore, in performing text localization, the results could be text and non-text regions. In the case of missing detection, the challenge of how to classify the candidate regions as text or non-text is also very important. In the literature, Support Vector Machine (SVM), a machine learning is widely used for classifying text or non-text images [7]. Hyung [3] classified text/non-text regions by creating a text/non-text classifier based on normalized gray-scale images (without binarization), and the T-HOG descriptor [8] is a text classifier that is used to characterize single-line texts in the image. However, the accuracy rates of the mentioned two classifiers are not very good depending on the process of model production. Therefore in this paper, we present a new continuous-word script localization and classification method using object histogram classifiers. The proposed technique has been evaluated on the text-detecting processes of a Thai scene image dataset (TSIB) [9]. The results show that our proposed technique effectively performs in text localization in scene images, particularly for continuous-word scripts such as Thai.

2. Related Work

Natural scene images contain a variety of colors which make it difficult for a computer to identify an object in a picture. Color-based components have been used to segment the foreground from a background [10], using an ordinary method for segmenting text image based on chromatic and achromatic components; these analyze and compare the performance of four color components such as RGB, YUV, YIQ and HSI. Segmenting text region by HSI together with k-means clustering clearly shows that the hue component is robust to highlights, shading and shadows. A connected component in conjunction with color quantization [11] is used to extract text regions from natural scene images by reducing a bit-color image and computing its algorithm. The experimental results yielded a detection rate of 84.55% and a false-alarm rate of 5.61%. Solving the problem of text retrieval from the complex background by using a touch screen interface was addressed in [12] using ICDAR2003 and KAIST scene text databases [12].

Nevertheless, text localization faces problems with big reflection effects, small text region because of inadequate

resolution, a variation in the stroke width of a character, a small difference between the text and background and an immoderate color change within a single image component. However, the authors insist that the problem of extracting characters from the complex background can be solved. Fang [13] illustrates a method for detecting and tracking road signs from video images with complex backgrounds by extracting color and shape features by two neural networks, respectively. A process characterized by fuzzy-set discipline is used to extract road-sign candidates. In the tracking phase to predict the positions and sizes, a Kalman filter is then introduced. The performance of the method is relatively accurate and robust.

Texts in common scene images are various patterns. A canny edge detector is used to compute edges in the image and it then groups pixels into letter candidates by stroke width transform [5] to extract text from a complex background. The algorithm has been tested on ICDAR 2003, ICDAR 2005 and the publicly available dataset with result of word recall rate is 79.04%, the stroke precision is 79.59%, and the pixel precision ratio is 90.39%. Another image feature that can be used to apply the method to many languages and fonts was introduced: Stroke Gabor Words [14] in which Gabor filters are used to describe and analyze the stroke components in text characters or strings to detect text in natural scene images. The k-means algorithm combines to cluster the descriptive Gabor filters; ICDAR 2003 and [5] are used. A performance evaluation of the algorithm resulted in text region detection with the ground truth precision 0.64, recall 0.76 and f-measure 0.68, which shows that it is able to handle complex backgrounds and variant text patterns.

3. Proposed Method

3.1 SIFT Codebook Histogram Modeling

Locating objects in a picture is often performed in a natural scene OCR process. The object expectancy approach is a potential technique to solve this challenge. Finding only an object that is expected in the image is fast, uses low computation, and is similar to human behavior. Internal models of expected objects are the key to locating the area of the objects. Therefore, a model of the object is needed. In order to detect text regions, text models in the picture must be created. To ensure that the text detection is accurate and can distinguish text from the background, we proposed to create two types of object model: pure text, and

combined text and background. These object models are called the codebooks, and the detail of producing the codebooks is described as follows.

(a) Text and non-text selection

To obtain a codebook, we first need to prepare its constituents. The text codebook will be created using the text objects while the non-text codebook needs the background objects from training images to be built up. Based on the annotation of all the text regions that occur in the images, we extract all the text zones in each image using its annotated data while additional steps can extract the non-text zones. First, the equivalent area of non-text to the text area in each image needs to be calculated to make it balance. We count the number of all the text zones and multiply them by the average dimension (width and height) of all the text zones of each image. The calculated area is called ‘an expected area’.

We next find and extract the largest non-text area that is greater than the expected area from the picture. Finally, we obtain an equivalent area of non-text from the picture. However, some pictures cannot provide the equivalent non-text area because it is too small or less than the expected area. We consider skipping all the text and non-text zones of that image. Formulas 1 and 2 describe the selection.

$$avg_t = (\sum_{i=1}^m (w_i * h_i)) / m \quad (1)$$

$$x_t = avg_t * m \quad (2)$$

Given p , where images by $p = 1..n$ which each p has $1..m$ text zones represented by t . Let w, h be width and height of each text zone respectively. So, the avg is the average area of the text zones in an image. Next, x is the expected area of the non-text that results from the multiplication of avg and m . Performing this selection should ensure that we get equivalent text and non-text areas for producing the codebooks. From the above formulas, we depict the process to obtain the text and non-text areas as shown in Fig. 1.

Fig. 1. (a) An original image from the ground truth. (b) The black rectangles are the text zone from the ground truth. (c) The expected non-text area of the image.

(b) Feature extraction

All the text and non-text zones from the previous section are considered the constituents, and their attributes (features) will be used to generate parts of the codebook. We choose the SIFT [15] algorithm for extracting the features from each zone because SIFT is well-known and widely used to extract the feature of the objects in the scene image. The SIFT features are also known as keypoints, that consist of the coordinate (x, y) , *scale*, *orientation* and *128 descriptors*. Several parameters can be adjusted to obtain a robust keypoint, for example, steps per scale octave or initial Gaussian blur level (sigma). As a result, the SIFT algorithm generates text and non-text features from the zones, and we combine all the text features to the first database as well as the non-text features to the second database.

(c) K-Means clustering

The size of the keypoint database is typically large because of the unique keypoint's location (x, y) but some descriptors are rather similar. The similarity of the descriptor may exist because the keypoints are generated from similar parts of objects. So, k-means clustering is applied to group similar keypoints to reduce the diversity. We found that grouping keypoint with a large number ($k=3,000$) provided a high accuracy of the codebook [9], but the computational time also increased when using too large codebook. Based on this, we decided to group the keypoints $k=4,000$, for which the computational time is acceptable. We employed k-means clustering [16] to partition the keypoints into 4,000 clusters for each database. Receiving the clusters of the text and non-text, all the cluster's centroids are called prototypical keypoints (PKP), that represent parts of the codebook. These PKPs produce two preliminary codebooks. The first codebook (CB1) is the pure text, which consists of 4,000 PKPs of all the text zones as shown in algorithm1. The second codebook (CB2) is the combination of text and non-text which consists of 8,000 PKPs (4,000 each) in total.

Algorithm 1: Pure text codebook (CB1)

Input: set of training text images, $Img = \{Img_1, Img_2, \dots, Img_m\}$

Output: set of PKP of training text images,

$PKP = \{pkp_1, pkp_2, \dots, pkp_k\}$. Where m = number of total images; k = number of keypoint's clusters ($k=4,000$).

$Kps \leftarrow \text{getDescriptor}(Img_{1..m})$

$C = \text{kmean}(Kps, k)$

$PKP \leftarrow \text{getCentroid}\{C_{1..k}\}$

(d) PKP improvement

Both the preliminary codebooks in the previous section consider whether they are text or non-text, based on the creation of text and non-text resources. In the second codebook (CB2), however, it is not guaranteed that the PKPs within the codebook are exact text or non-text PKPs. All the PKPs of the second codebook need to be improved to what they exactly are (text or non-text). For this reason, we apply an additional improvement to the second codebook by matching every PKP back to the text and non-text zones. The matched number between the codebook's PKPs and the zones are represented as matching histograms as shown in algorithm 2.

Algorithm 2: Improved codebook (CB3)

Input: set of text zones, $Txt = \{Txt_1, Txt_2, \dots, Txt_m\}$,
set of non-text zones, $NTxt = \{NTxt_1, NTxt_2, \dots, NTxt_n\}$,
the combined codebook (CB2), where m, n = number of
text and non-text images.

Output: improved codebook (CB3).

$$HisTxt = \sum_{i=1}^m match(CB2, Txt_i)$$

$$HisNTxt = \sum_{i=1}^n match(CB2, NTxt_i)$$

$$CB3 = sort(HisTxt - HisNTxt, desc)$$

All the histograms generated from matching with text (Txt) and non-text ($NTxt$) zones are combined into two final histograms ($HisTxt$, $HisNTxt$). After that, we subtract the non-text histogram from the text histogram. The value of the subtraction could be positive or negative for each histogram bin. We consider that the positive index can confirm that the PKP is text because the number of matches for this PKP to the text zones is greater than the non-text zones. Finally, we sort the subtracted histogram and assign the positive index as text and the negative index as non-text PKPs. The second codebook's PKPs are then rearranged into the new improved codebook (CB3).

(e) Generating the object classifier

Besides the codebooks, object verification models are also important to our proposed method. The verification model is a histogram model that can judge the candidate detected area as text or non-text. Both the text and non-text histograms generated from the previous section will be combined and normalized into one (1.0). Using this process, we receive two verification models called classifiers. To evaluate the accuracy rate of our classifiers, we test matching our codebooks back to the text and non-text zones

of the training data sets and compute the distance between the matching result in histogram formation and the classifiers. The shorter the distance, the more likely it is that the classifying zone is a member of that classifier. The accuracy rate of our classifiers is shown in Table 1.

Table 1. Classification results and the timing information in millisecond per zone.

Distance method	Classification (%)	ms/vector
Jaccard	94.29	70.88
Dice	94.23	69.88
Chisquare	94.10	74.50
Kullback	93.97	68.63
Bhattacharyya	93.59	69.25
Manhattan	93.39	67.63

The table shows that different distance measurements affect the accuracy of the classifier. The Jaccard, Dice and Chisquare methods perform better than others at 94.29%, 94.23% and 94.10% respectively. Unfortunately, they are not so fast when compared to the Manhattan method at 93.39%; even though this produces the lower classification result, its time-consumption in millisecond per zone is faster than others by noise condition at 66.88 ms/vector. We, therefore, selected the Manhattan algorithm for our research.

3.2 Text Localization

In this section, we present our proposed text localization method using the codebooks that are generated by the procedure in the previous section. Based on the expectancy approach, we expect that the text codebook should match all the texts in the testing image while the non-text model should match the complex background of the image. The visualization of the localization method is shown in Fig. 2.

(a) Locating matched keypoints

The process starts by using the SIFT feature extraction method to extract SIFT features (keypoint vectors) from a test image (Fig. 2a). A set of extracted keypoints is matched to prepared codebooks (CB1, CB2, CB3). The matching process computes the distance between the keypoint of the image and PKPs of the codebook (1-to-N). Using a distance measurement algorithm, all the computed distance results are captured and sorted in order to find two minimum distances of the candidate PKPs. These candidates are next calculated to determine an explicit nearest candidate by

multiplying the distance ratio by the second candidate. If the distance of the first candidate is less than the multiplication, it is clearly distinct from the second one and it can be judged that the first PKP matched with the given testing key point. In contrast, if the first candidate's distance is higher than the calculation, these neighbors are considered too similar and ambiguous. We then ignore both the candidate's PKPs. We repeat matching for all keypoints of the testing image. After the matching process, all the matched keypoints of text are plotted by black circles on a 2D spatial image (Fig. 2b), which is exactly the same size of the testing image based on their coordinates (x,y).

Fig. 2. Detection process.

(b) Dissolving patched object area

There are many matched keypoints scattered on the blank image. The plotted points represent the keypoints that are supposed to be the text in the image. To identify the detected area, merging adjacent keypoints together is the key. We propose the dissolving method to do this. First, we apply the horizontal blur to the image and adjust the blurring width as a parameter. The different image dimensions give different blurring results when there is a specific width size in the pixel unit. We then normalize the parameter to the percentage compared to the image dimension. The blurred points may be separated from others in the same text line. The vertical merging should be performed. We next apply the Gaussian blur to the image.

All the separate lines will be dissolved to others within the Gaussian blurring level (sigma). By this process, we receive the dissolved image (Fig. 2c), which the dark area is supposed to be the text objects.

(c) ROI bounding box identification

At this point we have an approximate location of the text, but its area is still unclear. The dissolved image is then converted into black and white based on the normal threshold. The black and white image (Fig. 2d) illustrates more accurate text areas in the picture. In many cases, the black areas are not linked to other text areas. To link them, we apply a convex hull algorithm [17], which returns the 2-D convex hull of the points (x,y), where x and y are column vectors, to solve this problem. Then all the bounding boxes are extracted as the candidate region of interest (ROI) of the text. Fig. 2e shows an example of candidate ROIs of the detecting method.

(d) Candidate ROI classification

As aforementioned, we already created the object classifiers for text and non-text. All candidate ROIs, which are extracted from the testing image, will be examined to make a final decision of its type with the classifiers. The candidate ROI that is judged as text represents the final result of the localization while the ROI classified as non-text will be eliminated. Fig. 2f shows an example result of our proposed method.

4. Evaluation and Results

4.1 Dataset

In this paper, we focus on improving general text-chunk localization in scene images using the proposed method. Image datasets are needed, and Thai text dataset is selected to be used in this study. The dataset was already introduced by [9], and is called the Thai Scene Image dataset or in brief TSIB. The TSIB consists of 1,566 images of scene images captured by a smartphone at 1,280x720 pixels. These images are divided into two groups for training and testing. All the text zones in each image are annotated as ground truth files. The training set is used for creating text and non-text codebooks, as well as the object histogram classifier. In order to reveal the performance of our improvement, the testing dataset is manually selected and separated into three sets. The single text line (TSIB-1) dataset consists of 403 images, and the multiple text line (TSIB-2) consists of 324 images. The combination of TSIB-1 and TSIB-2, named TSIB-3, consists of 727 images

in total. According to the set of TSIB images, the zone of text-chunk are extracted. The number of the text and non-text resources showed in Table 2 and Fig. 3 demonstrated examples of text (Fig. 3a) and non-text (Fig. 3b) zones.

Table 2. The text and non-text resources of TSIB datasets

Type	Text chunk Images			Zones	
	Single	Multiple	Mix	Training	Testing
Text	403	324	727	1,850	1,258
Non-text	-	-	-	642	590

Fig. 3. Examples of text (a) and non-text (b) zones

4.2 Evaluation Method

We evaluated our proposed method using the definitions of precision and recall: $precision = |TP|/|E|$, and $recall = |TP|/|T|$ where TP is defined as the set of true positive detections while E and T are the total numbers of estimated rectangles and the area of ground truth. The F-measure is computed, which based on the above precision and recall rate: $f = 2 * precision * recall / (precision + recall)$.

We prepare two codebooks (pure text and combination codebooks) as mentioned in the previous section. All the testing datasets are evaluated with codebooks with different parameters. We separate the testing into two steps. First, we perform the text detection with our proposed method. Second, we improve the detection using our proposed classifier and then calculate the performances of precision, recall and the F-measure.

4.3 Results

Fig. 4 shows the harmonic mean score of the detection from three experiments with different Gaussian thresholds (sigma = 3.2, 4.8, 6.4, 8.0 and 9.6) and fixed horizontal blur (10% of image width). The blue line is the detection result of the TSIB-2 dataset matched to the first codebook (CB1) with the non-classifier. The red line is the detection result of the TSIB-2 dataset matched to the second codebook (CB2) with the non-classifier. The black line is the detection result

of the TSIB-2 dataset matched to the improved codebook (CB3) with the classifier. The graph illustrates that the accuracy decreases when the Gaussian threshold is higher.

Table 3 presents the detection results of TSIB-1, TSIB-2 and TSIB-3 by matching them to the first codebook (CB1) with the non-classifier, the improved codebook (CB3) with the non-classifier and the improved codebook (CB3) with the classifier respectively. Fig. 5 shows the detection result after using the classifier. The examples of the text detection results of the TSIB and ICDAR [18] datasets are depicted in Fig. 6a and Fig. 6b respectively.

Fig. 4. Harmonic mean curves for the three experiments.

Table 3. The detection result of the separated three sets of TSIB with different methods by illustrated as precision (p), recall (r) and f-measure (f)

Experiments	p	r	f
TSIB-1+CB1+no-classifier	0.30	0.75	0.43
TSIB-1+CB3+no-classifier	0.52	0.79	0.63
TSIB-1+CB3+classifier	0.71	0.79	0.75
TSIB-2+CB1+no-classifier	0.33	0.48	0.39
TSIB-2+CB3+no-classifier	0.56	0.55	0.55
TSIB-2+CB3+classifier	0.66	0.54	0.60
TSIB-3+CB1+no-classifier	0.32	0.57	0.41
TSIB-3+CB3+no-classifier	0.54	0.64	0.58
TSIB-3+CB3+classifier	0.69	0.63	0.66

Fig. 5. Examples of text detection after applying the proposed classifier.

Fig. 6. Examples of text detection for the TSIB dataset (a) and ICDAR2003 dataset (b).

5. Conclusions

In this paper, we have presented our proposed method for general text-chunk localization in scene images. Thai text codebooks and classifiers have been generated and applied to the method. The text-locating performances show our proposed technique perform effectively. Although it works well with the Thai text datasets, there are, however, some issues that can be further improved. First, the classifiers are simply generated from the histograms of the matching between the codebooks and training images without any intelligent algorithm, but the performances are still good. However, to improve the classifier, some machine learning algorithms such as Support Vector Machine (SVM) or Artificial Neural Network (ANN) can be applied. Second, our text-locating method using the calculation of bounding boxes from the convex hull generation causes inaccurate ROIs in the text. Results of

codebook approach promising, but color information is not used in SIFT. For the future work, text and non-text classification on relative illumination independence using autocorrelation on color histograms, which exploits color could be better than SIFT.

References

- (1) Y.-F. Pan, X. Hou, and C.-L. Liu, "A hybrid approach to detect and localize texts in natural scene images," *Image Processing, IEEE Transaction on*, vol. 20, no. 3, pp. 800-813, March 2011.
- (2) Y. F. Pan and X. Hou and C. L. Liu, "Text localization in natural scene images based on conditional random field," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, July 2009, pp. 6-10.
- (3) H. I. Koo and D. H. Kim, "Scene text detection via connected component clustering and nontext filtering," *Image Processing, IEEE Transaction on*, vol. 22, no. 6, pp. 2296-2305, June 2013.
- (4) N. Ezaki, M. Bulacu, and L. Schomaker, "Text detection from natural scene images: towards a system for visually impaired persons," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, Aug 2004, pp. 683-686.
- (5) B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 2963-2970.
- (6) J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: a survey," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 7, no. 2-3, pp. 84-104, 2005. [Online]. Available: <http://dx.doi.org/10.1007/s10032-004-0138-z>
- (7) K. Kim, H. Byun, Y. Song, Y. Choi, S. Chi, K. Kim, and Y. Chung, "Scene text extraction in natural scene images using hierarchical feature combining and verification," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, Aug 2004, pp. 679-682.
- (8) R. Minetto, N. Thome, M. Cord, N. J. Leite, and J. Stolfi, "T-hog: An effective gradient-based descriptor for single line text regions," *Pattern Recogn.*, vol. 46, no. 3, pp. 1078-1090, Mar. 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.patcog.2012.10.009>

- (9) B. Sriman and L. Schomaker, "Object attention patches for text detection and recognition in scene images using sift," in *Proceedings of the International Conference on Pattern Recognition Applications and Methods*, 2015, pp. 304-311.
- (10) J. Park, H. Yoon, and G. Lee, "Automatic segmentation of natural scene images based on chromatic and achromatic components," in *Proceedings of the 3rd International Conference on Computer Vision/Computer Graphics Collaboration Techniques*, ser. MIRAGE'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 482-493. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1759437.1759482>
- (11) C. Li, X. Ding, and Y. Wu, "Automatic text location in natural scene images," in *Document analysis and Recognition*, 2001. *Proceedings. Sixth International Conference on*, 2001, pp. 1069-1073.
- (12) M. S. C. Jehyun Jung, SeongHun Lee and J. H. Kim, "Touch tt: Scene text extractor using touchscreen interface," in *ETRI Journal*, vol. 33, no. 1, 2011, pp. 78-88.
- (13) C.-Y. Fang, S.-W. Chen, and C.-S. Fuh, "Road-sign detection and tracking," *Vehicular Technology, IEEE Transactions on*, vol. 52, no. 5, pp. 1329-1341, Sept. 2003.
- (14) C. Yi and Y. Tian, "Text detection in natural scene images by stroke gabor words," in *Document Analysis and Recognition (ICDAR)*, 2011 *International Conference on*, Sept. 2011, pp. 177-181.
- (15) D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004. [Online]. Available: <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>
- (16) E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications biometrics," *Biometrics*, vol. 21, pp. 768-769, 1965.
- (17) Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., "The Quickhull algorithm for convex hulls," *ACM Trans. On Mathematical Software*, vol. 22, no. 4, pp. 469-483.
- (18) S. Lucas, "Icdar 2005 text locating competition results," in *Document Analysis and Recognition*, 2005. *Proceedings. Eighth International Conference on*, vol. 1, Aug 2005, pp. 80-84