Noname manuscript No.
(will be inserted by the editor)

Bi-directional Extraction and Recognition of Scene Text with Layout Consistency

Ryota Hinami · Xinhao Liu · Naoki Chiba · Shin'ichi Satoh

Received: date / Accepted: date

Abstract Text recognition in natural scene images is a challenging task that has recently been garnering increased research attention. In this paper, we propose a method of recognizing text by utilizing the layout consistency of a text string. We estimate the layout (four lines of a text string) using initial character extraction and recognition result. On the basis of the layout consistency across a word, we perform character extraction and recognition again using four lines, which is more accurate than the first process. Our layout estimation method is different from previous methods in terms of exploiting character recognition results and its use of a class-conditional layout model. More accurate and robust estimation is achieved and it can be used to refine character extraction and recognition. We call this two-way process—from extraction and recognition to layout, and from layout to extraction and recognition—"bi-directional" to discriminate it from previous feedback refinement approaches. Experimental results demonstrate that our bi-directional processes provide a boost to the performance of word recognition.

Keywords Scene text recognition – Character recognition – Word recognition – Layout consistency

R. Hinami
The University of Tokyo
E-mail: hinami@nii.ac.jp

X. Liu
Tokyo Institute of Technology
E-mail: liuxinhao@ok.ctrl.titech.ac.jp

N. Chiba
Rakuten, Inc
E-mail: naoki.a.chiba@mail.rakuten.com

S. Satoh
National Institute of Informatics
E-mail: satoh@nii.ac.jp

1 Introduction

As the number of vision systems (e.g., mobile cameras) increases, so too does the need for information retrieval from images. Text in natural scene images contains valuable information pertaining to street signs, store names, product brands, and so on. Recognizing the text in these images is becoming more important than ever before.

Although conventional optical character recognition (OCR) has been successfully commercialized, its application is limited to the realm of business documents and factory automation. Recognizing text in natural scenes is still a challenging problem and has been garnering a significant amount of attention from academia. Such images frequently contain blurred text on complicated backgrounds, and the uncontrolled lighting conditions can cause reflection, highlights, and/or shadows on the text, which makes scene text recognition more difficult than OCR.

Text recognition research can be divided into two categories: text detection and word recognition. The method we propose in this work falls into the category of word recognition, specifically, the recognition of a word from a given cropped rectangular region in an image. Word recognition is regarded as the key part of end-to-end scene text recognition.

Character extraction is a critical stage in the word recognition of scene text. Character extraction is difficult, especially for scene text because scene text is not easily segmented or binarized due to various factors such as noise, blur, and complicated background. Two types of methods are mainly used in character extraction: connected component-based (CC-based) and classifier-based. CC-based methods (e.g., MSER) extract characters by connected component analysis where

text pixels with similar properties are grouped to construct component candidates. Classifier-based (e.g., sliding window) methods find the candidate character by applying a character classifier to sub-windows in multiple scales through all locations of an image. Choosing which of these two methods to select is a system design issue. There is a trade-off between accuracy and processing time. Although sliding window methods are generally more accurate than CC-based methods, they are more computationally expensive.

Character recognition is also a fundamental problem of text recognition. Scene text's variation of font, noise, and distortion makes this problem a very challenging task. Various features and classifiers have been investigated as character classifiers (e.g., HOG feature + SVM classifier, convolutional neural networks). Geometric information (position of characters) is also used to improve the accuracy of character recognition. For example, in scene text recognition, Bissacco et al. [6] added the geometric feature (top and bottom positions relative to the height of text line) in addition to HOG features, while Neumann and Matas [28] corrected the initial recognition results by comparing the height with other characters to differentiate the upper and lower cases. Since vertical position and height are important features that characterize alphabets (e.g., the upper and lower cases of some characters such as 'o' are discriminated only by these features), incorporating precise geometry features is important to improve the accuracy of character recognition. However, in scene text, since it is possible to know the relative vertical position for only a few characters, it is difficult to estimate precise geometry information for character recognition.

In this work, we address the two problems mentioned above—loss of character geometric information and trade-off between two character extraction approaches—by focusing on layout consistency. Following the typography, we exploit four lines of a text string (Fig. 2) as layout consistency. Since fonts are designed on the basis of these four lines (layout consistency), they can be regarded as one of the important style that define the vertical position of characters. We can solve the two problems by performing character extraction and recognition using four lines: 1) we can perform efficient and accurate character extraction even with sliding window because character size and position are constrained by four lines, and 2) geometry information can be incorporated into character recognition by referring to the relative position between the four lines and characters.

Accurate estimation of four lines is indispensable in terms of making use of them for character extraction and recognition. However, since only a few characters can be extracted reliably from one word, information that we can use to estimate layout is limited. We therefore propose a novel layout estimation method that can estimate four lines from a limited number of characters. Our layout estimation method is different from previous methods in that it exploits character *recognition* results and uses a class-conditional layout model that is trained in advance. On the basis of four lines estimated by the proposed method, we refine the result of character extraction and recognition.

There are two main contributions in this paper. The first is accurate and robust layout (four line) estimation methods. The proposed method can estimate layout from only a few candidates by using recognition results and trained statistical layout model. In addition, our method can easily be adapted to other types of layout such as curve; i.e., it is not limited to linear. The second is a layout consistency-based feedback refinement process. Using the four lines estimated by our method, we perform the following two feedback refinements: character *re-extraction* by sliding window with four lines and *re-scoring* of character classification using relative position to the four lines. These two contributions are regarded as top-down (character extraction and recognition → word layout) and bottom-up (word layout → character extraction and recognition) processes, respectively. We combine these two processes into what we call a *bi-directional* process.

2 Related Works

In this section, we review works that are related to our method. We first review scene text recognition, especially word recognition. We then focus on geometric estimation related to our line estimation method. Finally, we review the style consistency (our layout consistency can be regarded as a kind of style consistency) and feedback refinement related to our re-extraction and re-scoring processes.

2.1 Scene Text Recognition

The problem of scene text recognition has been addressed as two tasks: text detection and word recognition. Text detection methods tackle the problem of localizing words in whole natural scene images and word recognition methods recognize the cropped word images obtained in the text detection stage. End-to-end systems that integrate both methods are addressed in [39, 41, 3, 14, 28, 29, 6]. Since our method focuses on the word recognition problem, we review the work related to word recognition here.

Typical word recognition systems starts with candidate character extraction. Character extraction methods can be categorized into two types: connected component-based (CC-based) and classifier-based. For connected component-based methods, MSER [24] is the most popular method in scene text recognition [31,29,28,30] because it achieves high recall compared with the binarization that is often used in document processing. For classifier-based methods, a sliding window consisting of a character classifier is often used [40,39,41,27,34]. Since multi-scale sliding window produces many overlapped candidates, non-maximum suppression is applied to them to reduce the number of candidate regions. Some recent works such as those by Bissacco et al. [6] and Alsharif and Pineau [3] use over-segmentation methods that find character segmentation points with a character classifier, while Weinman et al. [44] fully integrate segmentation and recognition into a probabilistic framework. In these methods, since the number of candidates is not so large compared to multi-scale sliding window, they do not use NMS.

Character candidates extraction is usually performed with high recall and appropriate character regions are selected in the next inference stage. Previous works have addressed character extraction using pictorial structure [39], conditional random field (CRF) [27,34], weighted finite-state transducers (WFST) [31], and hidden Markov models(HMM) [3]. Although the important components of CRF are energy functional and likelihood training, Mishra et al. [27] and Shi et al. [34] use only energy functional. Most models solve the word inference problem by a cost minimization problem with a Viterbi-like algorithm. A language model is often incorporated into this inference to check the consistency of language. Language models have been used since the appearance of OCR literature such as that by Jones et al. [16] using n-grams as an OCR post-processor and by Bazzi et al. [4] integrating n-grams in an OCR system. In scene text recognition, Thillou et al. [37] used n-grams for post-processing and Weinman et al. [42,45] integrated bi-grams for word recognition while Bissacco et al. [6] integrated n-grams. The consistency of not only language but also style is incorporated into this process, as we discuss in the next subsection.

Deep neural networks have recently been used in scene text recognition to improve the recognition accuracy. Wang et al. [41] use convolutional neural networks (CNN) with raw image input as a character classifier. Alsharif and Pineau [3] use a maxout network and further improve the performance of CNN's character classification. Specifically, they use the maxout network for segmentation and character classification together by integrating them into hybrid HMM models. Bissacco et al. [6] use the HOG feature as an input of the deep neural network to find character segmentation points as well as the character classifier.

Some recent work [2,12,14,11] performs word recognition by a holistic approach, in contrast to other character-based recognition approaches. Goel et al. [11] use whole image features to recognize words by comparing them with the synthetic images of lexicon words. Almazan et al. [2] embed word images and text strings in a common vectorial subspace and cast a word recognition task as a nearest neighbor problem. Gordo et al. [12] extend this work by using supervised mid-level features for word image representation. Jaderberg et al. [15] input a whole word image into a deep convolutional neural network trained with a huge amount of synthetic images. In an extension of this work [14], they integrate this word recognition framework into an end-to-end text recognition pipeline. Since these methods can deal with style consistency such as character alignment in a holistic way, they are extremely effective in tasks where a candidate lexicon is available as prior knowledge. However, it is difficult for them to recognize a new word out of lexicon.

2.2 Text Line Estimation

Estimation of text line has been used to improve text recognition accuracy. In handwriting recognition, Bengio and LeCun [5] estimate four tied curves for each word to normalize a word image. The model of the four curves is represented by six parameters and the most likely parameters are inferred by a probabilistic framework with an EM algorithm. Caesar et al. [7] estimate four tied lines based on the contour processing of a binary image with iterative regression analysis.

In scene text recognition, Neumann and Matas in [28] introduced a method to detect top and bottom lines for geometric image normalization while using a typographic model by measuring the height of uppercase and lower-case letters for correctly recognizing interchangeable letters. Neumann and Matas in [29] use four lines of a text string to verify text localization. They use the vertical extrema of character regions to fit four lines. They first estimate the common slope (direction) of four lines by fitting bottom points and then find four lines that minimize the square error. Weinmann et al. in [44] identified a pair of guidelines for a text string to normalize the region before recognizing words. They can infer the extrema in which the guidelines intersect even if characters are inclined through an iterative refinement approach.

These methods use extrema of character regions to estimate lines without recognizing characters. They con-

sequently require many candidate regions to estimate the four lines accurately as the exact typograhic style that we need. Since sometimes only a few characters can be extracted reliably from scene text, these methods are not suited for our purpose. Our method, in contrast, estimates lines using character recognition results. In addition, we use a character-dependent layout model that we train in advance. Our method can therefore extract a lot of information to estimate the word layout (four lines) and achieves accurate and robust layout estimation from only a few candidate regions.

2.3 Style Consistency and Feedback Refinement

Style consistency is important information and has been widely used in text recognition [33, 36, 4]. In terms of scene text, Weinman et al. in [42, 45] incorporate character similarity as style consistency under the assumption that the appearance of characters with the same class is similar if the source (font) is the same. They incorporate this information into a probabilistic framework so that characters with similar appearance are given the same label and character with dissimilar appearance are given different labels. Weinman et al. [43] incorporate the bi-gram model of character width into the inference to find the best parse, wherein correlation between the widths of a character pair is statistically modeled with reference to character class. Novikova et al. [31] use color and vertical position as style consistency. Specifically, they use color information and the distance from the top and bottom points of character to the baseline as a part of the word inference process.

We conclude this section with a brief review of works related to our feedback refinement, i.e., the re-extraction and re-scoring processes. Huang et al.[13] split the connected characters detected by MSER using sliding window with a CNN classifier after estimating the text line from initial candidates. Although this process is related to our re-extraction using sliding window, it cannot recover missing characters, unlike our method. Mishra et al. [27] use character classifier re-scoring based on aspect ratio, which is related to our method using vertical geometry positions to re-score character classification although Mishra et al. does not use word-level knowledge to re-score. Neumann and Matas [28] correct the initial recognition results by feedback loop to differentiate the upper and lower cases of certain letters by comparing the height with other recognized characters.

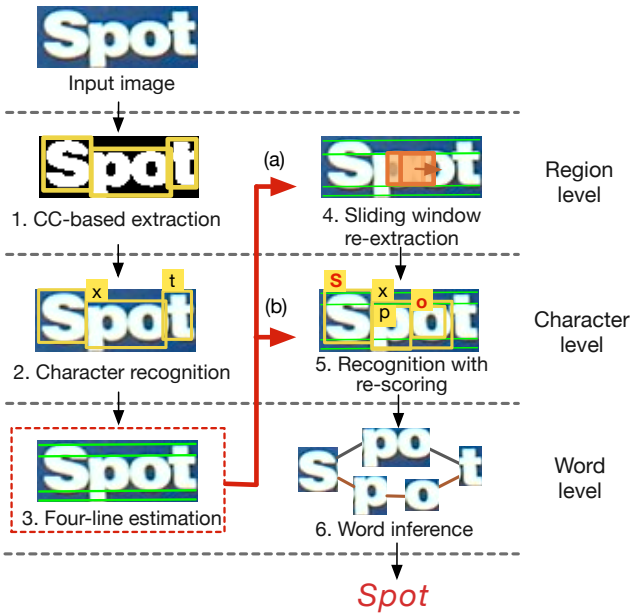


Fig. 1: The pipeline of the proposed word recognition method.

3 Overview of the Approach

Our method begins by extracting of reliable characters and using them to estimate four lines using them (Section 4). Character re-extraction and re-scoring are then performed using the four lines and the optimal word is inferred from obtained candidates with a cost minimization framework (Section 5). Figure 1 shows the pipeline of the proposed method.

We first extract reliable characters whose scores are sufficiently high. To extract a reliable character, we use CC-based methods (MSER and Otsu binarization) on an input word image and apply a character classifier to extracted components. We then estimate the layout consistency (four lines of a text string) on the basis of the reliable characters. Unlike previous estimation methods, which use character *extraction* results to obtain constraints [28], we use character *recognition* results to obtain layout consistency. Each character recognition result predicts the distribution of the line positions by using a character class dependent model that is trained in advance, which allows accurate estimation even if only a few candidates can be extracted.

We then perform character extraction and recognition again under the layout consistency defined by four lines. We use two top-down cues, indicated by the red arrows in Fig. 1. **(a) Re-extraction.** We extract character candidate regions by using layout consistency to concentrate on promising regions. Although we use sliding windows, we can perform a computationally efficient scan with four lines, unlike conventional multi-

Table 1: Character types

Ascender	bdfhiklt
Descender	gjpqy
Small lower case	acemnorsuvwxz
Upper case	ABCDEFGHIJKLM NOPQRSTUVWXYZ
Digit	0123456789

Fig. 2: Four lines of a text string.

scale sliding window. **(b) Re-scoring.** Among multiple character classes for each region, we re-calculate scores on the basis of layout consistency, giving higher scores to character classes consistent with the style and lower scores to classes with less consistency.

Candidate regions obtained by first and second scans are processed jointly in the word inference process. We find the optimal labeling (character or non-character, and its character class as well) that minimize the cost function. We use a cost function similar to the cost function in Mishra et al. [27], which is defined by classification score, spatial constraints, and linguistic model. The labeling that minimizes the cost function is inferred by the TRW-S algorithm [19].

3.1 Character Classifier

Here, we describe the character classifier used in our paper. To recognize characters, we apply a character classifier to every candidate region. It generates classification probabilities (scores) for multiple character classes. We use HOG features [9] along with the aspect ratio of the bounding box of the character. Since alphabets tend to have certain aspect ratios for specific characters, aspect ratio is important information to identify character class. We confirmed a performance increase due to using aspect ratio (accuracy increase from 80.3% to 81.8% on the ICDAR 2003 dataset). An RBF kernel SVM is used for the classifier with the one-versus-all multiclass SVM setting. We give special treatment to *case-identical characters* (c-o-s-v-w-x-z) that are identical in upper and lower cases. Instead of 62 classes corresponding to upper and lower case alphabets and digits, we use 55 classes by merging the upper and lower cases of case-identical characters into one class. These classes are then separated into upper and lower cases in the re-scoring process described in Section 5.2.

4 Estimation of Four Lines

The proposed method starts with the estimation of four lines—ascender line, mean line, baseline, and descender line (Fig. 2)—using the initial recognition result. We first extract reliable characters that are used for line estimation. To estimate the lines from reliable characters, we train the Gaussian distribution model of the line positions depending on each character class in advance. We then estimate the equation of the four lines via maximum likelihood estimation by using the positions of reliable characters and trained models.

4.1 Extraction of Reliable Character

To obtain layout consistency (four lines of a text string), we use reliably extracted characters, i.e., reliable characters, whose scores are above the predetermined threshold. The idea of using reliable outputs to guide recognition has appeared in various literature on handwriting and speech recognition (e.g., Miller and Viola [25]). We use a connected component-based (CC-based) method to extract initial candidate regions. We extract candidate regions by two methods: MSER and Otsu binarization. Following the approach of Neumann and Matas [28], the MSER region detector [24] is used to extract the set of candidates. We also use Otsu binarization [32], where the connected components of binary images are used as another set of candidates. We combine the results of MSER and Otsu binarization to form the final result of CC-based extraction. We confirmed that the combination of these two methods delivers higher recall rates in the character extraction (see Section 6.3).

We then select reliable characters from candidate character regions by using a character classifier. We apply the character classifier to every candidate, which outputs a character class along with its score. The regions with scores above a certain threshold are selected as reliable characters. These selected regions are then used to estimate four lines of a text string in the next step. Since two or more reliable characters are required for the line estimation, if less than two characters with scores above the threshold are extracted, the two characters with the two highest scores are selected as reliable characters.

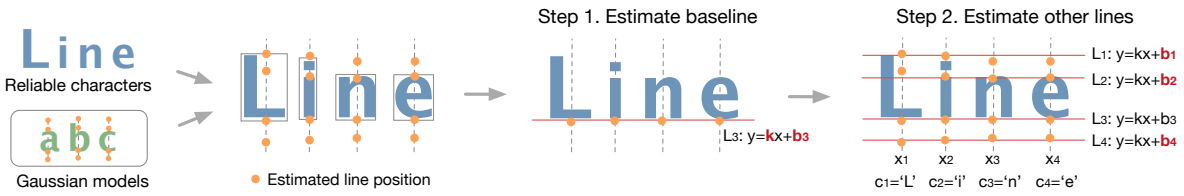


Fig. 3: Estimation of four lines. For each reliable character, distributions of line position are obtained from trained models. Equations of lines are then estimated by maximum likelihood estimation.

4.2 Training of Line Position Models

In this section, we describe how to train the Gaussian distribution model. For training we used the word image datasets with ground truth of character bounding boxes and their classes. We used the training part of the IIIT 5K-Word [26] and ICDAR2013 [18] datasets. In addition to these, we used the synthetic word images as training data because some character classes such as 'q' scarcely appear in public training sets. We used 30 words generated from different fonts and each synthetic word contains 62 characters (0..9, a..z, and A..Z). In total, 2,577 words containing 30∼577 characters for each character class were used for training. The detailed training procedure is shown in Algorithm 1. In summary,

1) Determine the ground truth lines of training images from given training data (lines 5–9).
2) Calculate the line positions relative to each character (lines 10–16).
3) Calculate Gaussian parameters μ_{lc}, σ_{lc} from obtained positions (lines 19–24).

We first determine four lines for each of the training word images. The lines that a character touches depend on the class of the character, as shown in Table 1—for example, upper case characters touch the ascender line and the baseline. On the basis of this fact, lines are fit to the centers of the upper and lower bases of the bounding boxes of characters using the least-mean-square (LMS) algorithm. This requires more than two points available for each line. If not available, corresponding lines will not be drawn and the information will not be taken into account for the corresponding training word image.

We then compute vertical positions of lines relative to each character region. The intersections between four lines and the vertical center line of a character are used as the line positions for the character. These positions are then normalized on the basis of the height of each character region.

We construct a statistical model of line positions using the position values obtained from many word images. We assume normalized line positions yield the Gaussian distribution for each line and character class. Therefore, the means and variances of normalized line positions are obtained as the sufficient statistics of the Gaussian distributions for each of the 62 classes × 4 lines, namely, μ_{lc} and σ_{lc} as the mean and variance for each line l and character class c. l can be 1, 2, or 4 corresponding to ascender line, mean line, and descender line, respectively; also, $l = 3$ for the baseline for completeness. These models are used to estimate the line positions for each extracted region. Note that although Algorithm 1 explicitly use the set of values P_{lc}, we can calculate the mean and variance efficiently by maintaining not P_{lc} but rather just the sum and sum of squares and updating them for each iteration.

Algorithm 1 Training of line position models

Input: Number of training word images N, Ground truth data: character class and bounding box.
Output: $\{\mu_{lc}\}, \{\sigma_{lc}\}$

```
1: initialize {P_ij}_{i=1,j=1}^{i=4,j=62} ⇐ 62 × 4 empty set
2: for i = 1 to N do
3:     C ⇐ set of characters in ith word image
4:     for l = 1 to 4 do
5:         D ⇐ set of characters that touch line L_l
6:         if size(D) < 2 then
7:             continue
8:         end if
9:         L_l ⇐ Least-square fitting(D)
10:        for k = 1 to size(C) do
11:            c ⇐ character class of C_k
12:            b ⇐ bounding box of C_k ([x y width height])
13:            p ⇐ y-position of L_l at x=b_x + b_width/2
14:            p̃ = (b_y − y)/b_height
15:            add p̃ to P_lc
16:        end for
17:    end for
18: end for
19: for c = 1 to 62 do
20:    for l = 1 to 4 do
21:        μ_lc ⇐ mean(P_lc)
22:        σ_lc ⇐ variance(P_lc)
23:    end for
24: end for
```

4.3 Estimation of Line Equations

We now estimate equations of lines from reliable character extraction results by using a trained model of line position. For each character region, we can estimate the positions of lines, as shown in Fig. 3. Let us denotes the equation of l th lines as $L_l : y = kx + b_l$. The estimation of line equations is performed with the following two steps:

1) Estimate the equation of baseline (common slope k and intercept of baseline b_3).
2) Estimate the intercept of other three lines (b_1, b_2, and b_4).

We first determine a baseline L_3. Since most characters touch the baseline, its estimation is expected to be more reliable than others. Neumann and Matas [29] also estimate baseline prior to the other three lines. The vertical location of the baseline at each reliable character is obtained from line position models as μ_{3c}, where c is its character class. The baseline is inferred by fitting these locations using LMS.

Next, other lines are estimated. Since the slope of these line is the same as that of the baseline, we only estimate the intercepts for each of three lines (b_1, b_2, and b_4). We estimate the intercepts by maximizing likelihood under the constraint of the slope k. Note that case-identical characters (c-o-s-v-w-x-z) are excluded from reliable characters to estimate lines other than the baseline. Given N reliable characters in a word, let $c_1, c_2, \cdots, c_N$ be corresponding character classes and $x_1, x_2, \cdots, x_N$ be horizontal locations at the centers of the characters, as shown in Fig. 3 right. Further assuming $p(y_{il}|c_i)$ to be the probability that the intercept of the lth line at x_i is y_{il} when the character class of the ith character is c_i, the intercepts that maximize likelihood can be obtained as

$$\begin{aligned} b_l &= \arg\max_{b_l} \sum_{i=1}^{N} \ln p(y_{il}|c_i) \\ &= \arg\max_{b_l} \sum_{i=1}^{N} -\frac{(\tilde{\mu_{lc_i}} - y_{il})^2}{2\tilde{\sigma_{lc_i}}^2} \\ &= \frac{\sum_{i=1}^{N} \frac{1}{\tilde{\sigma_{lc_i}}^2} (\tilde{\mu_{lc_i}} - kx_i))}{\sum_{i=1}^{N} \frac{1}{\tilde{\sigma_{lc_i}}^2}}, \end{aligned} \tag{1}$$

where $\tilde{\mu_{lc_i}}, \tilde{\sigma_{lc_i}}$ are the mean and variance of the position of the lth line for c_i, normalized by the location and size of the ith reliable character.

By considering the variance σ_{lc_i}, we vary the priority of characters for estimation. For example, in Fig. 3, 'L' is considered more important to estimate the ascender line than the other characters. These estimated lines are used to re-extract missed character candidate regions and to re-score the initial scores from the character classifier. Figure 4 shows examples of line estimation results.

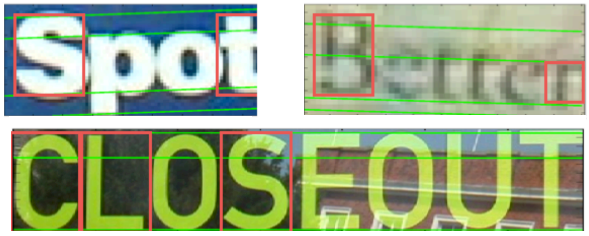


Fig. 4: Examples of line estimation results. The red rectangles are reliable characters used for the estimation of four lines.

4.4 Curved Text

Although we typically assume that characters in text are aligned in a straight fashion, scene images sometimes contain curved text. To extend the proposed method for curved text, we construct a four-concentric circle model in addition to the existing four-line model with adaptive model selection.

We first judge from reliable characters whether a given word is straight or curved. Since the slope of a curved text line is gradually changed from left to right, we check if the text is curved by the following steps. First, for each reliable character, the vertical location of the baseline is estimated from a trained Gaussian model. Adjacent locations are connected by lines and their slopes are calculated. We then calculate the subtraction between two slopes for every adjacent pair of lines. If these difference values are all positive or all negative, text is inferred as curved and a circle model is used for layout estimation.

Next, we explain how we estimate four concentric circles. First, a circle corresponding to the baseline (basecircle) is estimated using LMS. The other three circles are then estimated. Since we assume the four circles are concentric, the centers of the circles are the same as that of the basecircle. The radius of each circle is obtained by maximum likelihood estimation. Examples of curve estimation results are shown in Fig. 5. On the basis of four estimated circles, re-extraction and re-scoring are performed similar to the straight cases. Our evaluation of this system is described in Section 6.5.

The limitation of our method is that it does not work well for largely oriented text, since we used only the vertical extrema of a connected component as the intersection point; for oriented characters, vertical ex-

Fig. 5: Examples of four concentric circles estimation for curved text. The red rectangles are reliable characters used for estimation.

Fig. 6: Examples of the effect of sliding window re-extraction. The top row shows the extraction results by CC-based extraction (MSER and Otsu). The middle row shows the four lines estimated by our method. The bottom row shows the extraction results with sliding window re-extraction along four lines.

trema is not always the point that intersect with four lines. Weinman et al. [44] address this problem and infer the exact extrema that the guidelines intersect by using an iterative refinement method. However, the method they used cannot estimate guidelines from few characters (e.g., "Dry" in Fig. 7 of that work [44]) because it estimates the top and bottom lines without recognizing character class. We assume that accurate lines can be estimated even from largely oriented text with few characters by combining their iterative process with our method using the character recognition result.

5 Feedback Refinement with Four Lines

5.1 Sliding Window-based Re-extraction

The connected component-based method (Section. 4.1) sometimes misses characters, such as those shown in the top row of Fig. 6. The sliding window-based method, in contrast, can find such characters, but this comes at the cost of a huge number of candidate regions. Moreover, since windows need to be scanned at various scales, aspect ratios, and locations in a given image, a huge number of false alarms is produced and computational costs may become high.

Although we also use sliding window to re-extract missed characters, our method significantly reduces the number of windows to be checked by imposing layout consistency. We can reduce the number of windows for two reasons: (1) our method scans windows in limited space excluding reliable character regions, as shown in Fig. 7, and (2) we slide windows in the horizontal direction only because vertical positions are determined by word layout (four lines).

Here, we describe the details of the sliding window along four lines. Figure 8 illustrates the sliding window re-extraction. We change the vertical position and height of the windows in accordance with the following three cases: (1) windows to detect ascenders, upper cases, and digits between ascender line and baseline, (2) windows to detect small lower cases between mean line and baseline, and (3) windows to detect descenders between mean line and descender line. The width of the window is determined by selecting an aspect ratio from a set of predetermined ratios. The windows are then scanned along lines. The regions where reliable characters are detected will be skipped to further reduce the number of scans. In this way, characters that are missed in the first extraction are re-extracted, as shown in the bottom row of Fig. 6. Our two-way character extraction—a hybrid of CC-based and sliding window extraction—significantly reduces the computational cost while achieving high recall.

The final set of candidate character regions is obtained by combining MSER, Otsu binarization, and sliding window re-extraction. Previous sliding window-based methods [41,27,34] typically apply non-maximum suppression (NMS) to reduce the number of candidate regions before they are fed into a word inference stage. However, when we tried using NMS after our sliding window re-extraction, it was not as effective as in previous works because our method already sufficiently reduces the number of character candidate regions by layout consistency, and thus the difference in the performance (word recognition accuracy and processing time) does not change much between with and without NMS. We therefore decided not to use NMS following the method without sliding window, the same as Novikova et al. [31] who use MSER. False extractions are discarded by solving the cost minimization problem described below.

5.2 Re-scoring with Four Lines

Character classification scores are modified on the basis of on layout consistency before the word inference process. The new scores are then calculated on the basis of goodness of fit between the estimated lines and the

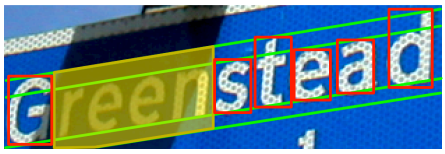


Fig. 7: Sliding window re-extraction. The yellow region is the space to be scanned, the green lines are the four estimated lines of a text string, and the red rectangles are bounding boxes of reliable characters.

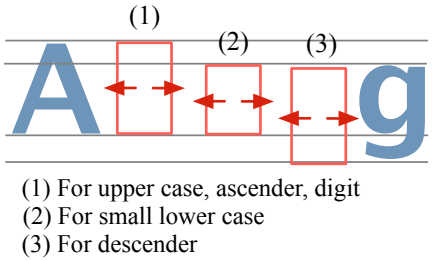


Fig. 8: The vertical position of windows in sliding window re-extraction.

trained models of line positions, as follows:

$$S(x_i = c_j) = p(c_j|x_i) \prod_{l=1}^{4} \exp\left(-\frac{(\mu_{\tilde{lc_j}} - y_{il})^2}{{\sigma_{\tilde{lc_j}}}^2}\right), \quad (2)$$

where $p(c_j|x_i)$ is the original classification score for i th candidate classified as class c_j. The scores for case-identical characters, i.e., $S(x_i = c_j^U)$ for the score of the upper case and $S(x_i = c_j^L)$ for that of the lower, are then calculated as

$$S(x_i = c_j^U) = p(c_j|x_i) \prod_{l=1}^{4} \exp\left(-\frac{(\mu_{\tilde{lc_j^U}} - y_{il})^2}{{\sigma_{\tilde{lc_j^U}}}^2}\right), \quad (3)$$

$$S(x_i = c_j^L) = p(c_j|x_i) \prod_{l=1}^{4} \exp\left(-\frac{(\mu_{\tilde{lc_j^L}} - y_{il})^2}{{\sigma_{\tilde{lc_j^L}}}^2}\right). \quad (4)$$

This re-scoring process enables more accurate character recognition because geometric information (vertical location) ignored in the first scan can be taken into account. For example, for a window between ascender line and baseline, the score of the window classified as label 'C' does not decrease because estimated lines match with the trained line positions of 'C'. In contrast, the score of 'c' decreases because estimated lines do not match with the models. As an additional benefit, windows located far from four lines will be assigned low scores for all character classes, which helps remove false positives in the word inference stage.

5.3 Cost Minimization

The application of character recognition to each candidate character region generates character classes with scores. However, the given set of character regions often contains false positives, and corresponding character classes include classification failure. We select appropriate character regions as well as promising character classes by an energy minimization framework.

We first construct a graph with candidate character regions as nodes and place edges between sufficiently close nodes. Each node is represented by a random variable X_i that takes a label x_i corresponding to a character class including non-character label ϵ. We then define a cost function by making use of character recognition scores, spatial constraints, and linguistic knowledge and infer the most likely word by minimizing the cost function.

We use a cost function similar to the one by Mishra et al. [27]. The cost function $E(x)$ is represented as a sum of unary and pairwise terms, as follows:

$$E(x) = \sum_{i=1}^{n} E_i(x_i) + \sum_{\varepsilon} E_{i,j}(x_i, x_j), \quad (5)$$

where $\mathbf{x} = \{x_i | i = 1, 2, ..., n\}$ is the set of all random variables, $E_i(x_i)$ is the unary cost, $E_{ij}(x_i, x_j)$ is the pairwise cost, and ε represents the set of all neighboring pairs of nodes, which is determined by the structure of the constructed graph. We use the same pairwise cost as Mishra et al. [27], using a bi-gram model and spatial constraint. We also use a similar formulation for unary cost as Mishra et al. [27] but with a layout-based score (2) instead of raw classifier confidence:

$$E_i(x_i = c_j) = \begin{cases} 1 - S(x_i = c_j) & if\ c_j \neq \epsilon, \\ \lambda max_j S(x_i = c_j) & if\ c_j = \epsilon, \end{cases} \quad (6)$$

where $S(x_i = c_j)$ is the score calculated in (2) and λ is the parameter of penalty for the node taking a null label, which is set to 10 in our experiments. Given these unary and pairwise terms, the word recognition result is acquired by minimizing the cost function. To do this, following Mishra et al. [27] and Shi et al. [34], we use the TRW-S algorithm [19].

6 Experiments

6.1 Datasets

We used several challenging public datasets—ICDAR 2003 [23], 2013 [18], 2015 [17], Street View Text (SVT) [39,40], and IIIT 5K-Word [26]—to evaluate the word recognition performance. The ICDAR datasets were scene

text recognition datasets created for the ICDAR competition. The ICDAR 2015 dataset [17] was created for the task of "Challenge on Incidental Scene Text" introduced in ICDAR 2015, which is a more challenging dataset than previous ICDAR datasets such as ICDAR 2003 and 2013. It contains more than 1,500 images acquired with wearable devices; thus, the images are often arbitrarily rotated and include bad conditions (e.g., blur, noise), unlike the focused scene text in previous ICDAR datasets. The detailed evaluation protocol of word recognition is described in Section 6.4. For ICDAR 2003, a 50-word lexicon is provided for each image by Wang et al. [39].

The SVT dataset contains images taken from Google Street View. Following the experimental protocol of previous works [39,40], we used the SVT-WORD subset that contains 647 words, where a 50-word lexicon is associated with each word.

The IIIT 5K-Word dataset is the largest dataset for cropped word recognition, consisting of 2,000 word images for training and 3,000 word images for testing. Each test image has an associated 50- and 1000- word lexicon for evaluation on a closed-vocabulary task.

We also used the training part of the ICDAR 2003 and IIIT 5K-Word datasets for training character classifiers because ground truth bounding boxes and character labels are provided on these datasets. Chars74K [10] was also used for training the character classifier.

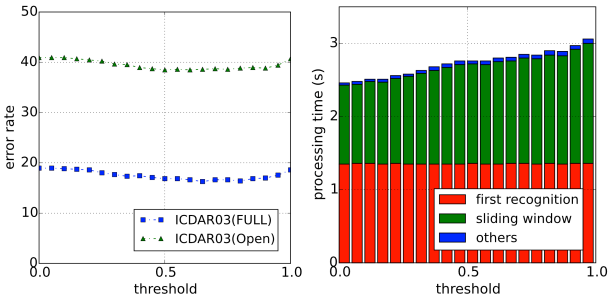
6.2 Evaluation of Character Classifier

In this section, we present a detailed implementation and evaluation of the character classifier used in our experiments. HOG features were computed with a block size of 2×2 cells and a cell size of 7×7 pixels using nine bins after resizing each image to a 28×28-pixel window. When the input image height was larger than 64 pixels, the input image was smoothed by a 4×4 median filter before resizing. We used the standard LIBSVM package [8] for training and testing the SVM classifier. The parameters of RBFSVM were determined by five-fold cross validation on the training data; C=6.72 and γ=1.5 were obtained. We combined the ICDAR 2003 [23] training dataset, Chars74k [10] dataset, and IIIT 5K-Word [26] training dataset as training data, resulting in a total of 27,715 characters. In all reported experiments, we used a 55-class SVM trained on these datasets unless otherwise specified.

To evaluate the performance of the character classifier, we tested the recognition rate on the ICDAR 2003 testing dataset. Table 2 shows the error rates of the character recognition compared to other methods. For this comparison, the result using a 62-class SVM (lower and upper cases of case-identical characters are trained separately) is also shown. The results indicate that our character classifier is comparable to the latest methods such as TSM [34] and Co-HOG [38,35] but behind Conv-Maxout [3] and CNN [41]. When we changed the training set to the combination of ICDAR 2003 training and Chars74K (same as Tian et al. [38] and Su et al. [35]), the error rate of our method on the 62-class SVM increased to 19.4% but was still comparable to the methods with Co-HOG [38,35]. We assume that the difference of performance between our method (HOG + non-linear SVM) and HOG + NN is due to the difference of classifier: non-linear SVM and nearest neighbor.

Table 2: Error rates of character recognition on ICDAR 2003 dataset.

Method	Error (%)
Conv-Maxout [3]	14.5
CNN [41]	16.1
HOG + non-linear SVM (ours, 55-class)	16.89
HOG + non-linear SVM (ours, 62-class)	18.23
Conv. Co-HOG [35]	19
Co-HOG + linear SVM [38]	20.6
TSM (49-class) [34]	22.14
HOG + NN [40]	48.5
ABBYY [1]	73.4



(a) (b)

Fig. 9: Effect of threshold of reliable characters

Table 3: Character extraction results on IIIT 5K-Word dataset.

Extraction method	Recall (%)
MSER + Otsu + re-extraction	**81.25**
MSER + Otsu	72.96
MSER	68.24
Otsu	55.29

6.3 Evaluation of Character Extraction

After character candidate regions were obtained by MSER and connected component analysis, we applied a char-

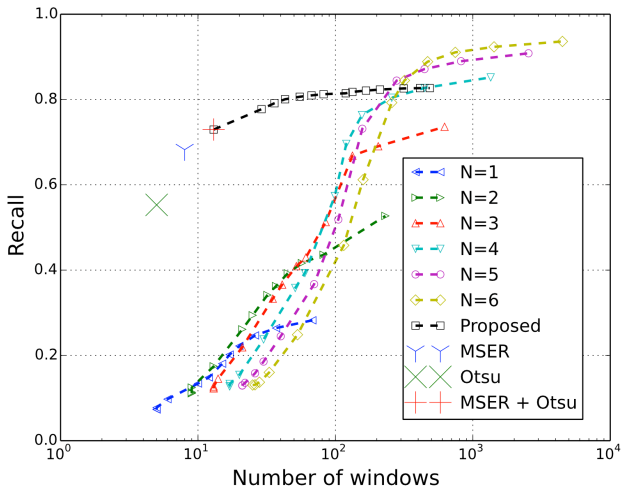


Fig. 10: Relation between the number of scanned windows per word and recall of character extraction on IIIT 5K-Word dataset. N is the number of vertical scales on conventional sliding window.

acter classifier to obtain character classes with scores. Reliable characters that had scores above a threshold (set to 0.8) were then selected. Although this threshold was manually determined, we confirmed that the word recognition performance is not very sensitive to this value. Figure 9 shows the effect of the threshold value on word recognition performance (both open and closed vocabulary tasks) and runtime tested on ICDAR 2003 dataset. The error rate is minimized with the threshold of 0.5∼0.9 and the runtime of sliding window increases gradually as the threshold increases.

We then estimated four lines of a text string from reliable characters using the trained models of line positions. We then performed sliding window-based re-extraction. We used 1.1, 1.4, 1.7, and 2.0 as the set of possible aspect ratios. The step size of the scan was fixed to 1/10 of the height of the window.

To evaluate the performance of character extraction from cropped words, we measured the recall of character extraction on the IIIT 5K-Word testing dataset. We computed the intersection over union (IoU) measure of a detected window compared to the ground truth with a threshold of 60%. Table 3 summarizes the results. Otsu, MSER, and MSER + Otsu show the recall using only first CC-based extraction and MSER + Otsu + re-extraction shows the recall using both first CC-based extraction and second sliding window re-extraction. The combination of MSER and Otsu achieved a recall of 72.96%, which is far better than using only MSER or Otsu. The recall was further improved by 8.29% (30.66% relative error reduction) by combination with our sliding window re-extraction. We also measured the percentage of the words wherein all characters are correctly extracted by the proposed method (Otsu + MSER + re-extraction). We confirmed that all characters in 72.93% of words (2,188 words / 3,000 words) are correctly extracted by the proposed methods, which shows that the character extraction errors are concentrated on the other 27.07% difficult words.

To determine the efficiency of our two-stage extraction method, we compared it with conventional multi-scale sliding window extraction. We measured the recall of character extraction and evaluated the efficiency by recall for the number of scans because computation time is mainly occupied by computing features and applying classifiers. In the conventional sliding window method, the height of the window was varied by the scale s as follows: $s = 2^{-\frac{n}{4}}$ $(n = 0, 1, ..., N-1)$, where N is the number of scales. The height of the window was set to s times the height of the word image. The step size was varied among 13 different values (from 0.01 to 10) times the height of the window for both (proposed and conventional) methods. Note that we do not apply NMS following the proposed method, because the performance with NMS highly depends on the character classifier and the threshold of NMS and thus fair comparison is difficult with NMS.

Figure 10 shows the relation between the number of scanned windows per word and the recall of character extraction (IoU >60% is used). The conventional sliding window method needed about 250 scans per word to achieve 80% recall while the proposed method achieved 80% with about 40 scans including the results of MSER and Otsu. These results demonstrate that our method achieves high computational efficiency while retaining high recall. Although the conventional sliding window method can exceed 90% recall with more than 1,000 scans for a given word image, it does so at a huge computational cost (more than 25 times that of our method).

6.4 Cropped Word Recognition

In this section, we provide a detailed evaluation of cropped word recognition. To determine the pairwise cost, we use a large English dictionary with around 0.5 million words provided by Weinman et al. [45]. We evaluate our performance on both a closed (limited) vocabulary task and an open (unlimited) vocabulary task. In the closed vocabulary task, we selected the word with the smallest edit distance in the lexicon as the final result. The lexicon consists 50 words in the tasks of ICDAR03(50), SVT, and IIIT5K(Small). All words in the test set are used as lexicon in ICDAR(FULL) and IIIT5K(Medium)

has a 1,000-word lexicon. Open vocabulary tasks measure the performance by case sensitive accuracy without edit distance correction. We evaluate ICDAR 2013 and 2015 in an open vocabulary task following the evaluation of the competition. We also evaluate ICDAR 2003 with an open vocabulary task (referred to as ICDAR03(Open)) in addition to ICDAR03(50) and ICDAR03(FULL). IIIT5K(Large) can also be considered a case insensitive open vocabulary task because lexicon with ground truth is not given. In the open-vocabulary task, to compare accuracy with another method that uses a strong language model, we implemented a simple post-processing method based on a 4-gram-based language model similar to the secondary language scoring in Bissacco et al. [6]. The language model is trained with 0.5 million words independent from the ground truth lexicon.

Following previous papers [39,27,31,34], words with two or fewer characters or with non-alphanumeric characters were excluded on ICDAR 2003. However, on ICDAR 2013/2015, we followed the exact evaluation protocol of the competition for a fair comparison.

6.4.1 Evaluation of Sliding Window Re-extraction

We have shown in Section 6.3 that sliding window re-extraction improves the recall of character extraction efficiently. To investigate the effect of re-extraction on the accuracy of word recognition, we evaluated the accuracy of cropped word recognition by changing extraction methods. We performed evaluation of the following four tasks: ICDAR03(50), ICDAR03(FULL), ICDAR03(Open), and ICDAR13.

Table 4 shows the performance using different extraction methods. The combination of MSER and Otsu achieved much higher performance than either MSER or Otsu alone. Moreover, sliding window re-extraction further improved MSER+Otsu extraction. Figure 11 shows sample results to demonstrate the effect of sliding window re-extraction. Missed and connected characters in the MSER+Otsu extraction were successfully extracted by the proposed method, demonstrating the effectiveness of the proposed re-extraction based on layout consistency.

6.4.2 Evaluation of Text Line-Based Re-scoring

To confirm the effect of the re-scoring on the performance of character classification, especially for case-identical characters, we evaluated the performance with and without the re-scoring on a case sensitive word recognition task on the ICDAR 2003 and 2013 datasets. We evaluated on following four tasks: ICDAR03(50), ICDAR03(FULL), ICDAR03(Open), and ICDAR13. In the method without the re-scoring, we used a 62-class SVM as a character classifier.

Table 6: Word recognition results on ICDAR 2013.

Method	Normalized edit distance	Error (%)
PhotoOCR [6]	**122.7**	**17.17**
Proposed	329.5	41.00
PicRead [31]	332.4	42.01
NESP [21]	360.1	35.80
ABBYY [1]	539.0	54.70

Table 7: Word recognition results on ICDAR 2015.

Method	Normalized edit distance	Error (%)
MAPS [20]	1128.0	67.07
NESP [21]	1164.6	68.32
DSM	1178.8	74.15
Proposed	1232.7	69.81

The recognition rates on the ICDAR 2003/2013 datasets are shown in Table 5. Although the performance of the closed-vocabulary task was increased slightly by the re-scoring, the performance of the open-vocabulary task was increased significantly. This indicates that our re-scoring stage corrects falsely recognized characters effectively although its effect becomes weak when vocabulary is limited.

To investigate the contribution of a re-scoring stage to each character class, we measured the character wise recognition performance on the IIIT 5K-Word dataset. Figure 12 shows the character wise recognition rate with and without re-scoring. The rate increase by re-scoring is also shown and the graph is arranged in the order of this amount. Character classes that do not contain enough test data items (less than 15) are excluded from the graph. As shown, the re-scoring has a positive effect on the classification, especially for case-identical characters (c-o-s-v-w-x-z). On the other hand, performance of numeric characters decreases because their position relative to other characters varies widely and this variation cannot be learned with limited training data. Although performance decreased for some characters, the overall character level recognition error rate was reduced from 26.79% to 24.35% (9.11% relative error reduction).

Figure 13 shows examples of the results to demonstrate the effect of the re-scoring. These results indicate that the re-scoring stage is effective to eliminate false positives and to distinguish confusing characters such as case-identical characters.

Table 4: Word recognition error rates on ICDAR 2003 dataset with different extraction methods.

Method	ICDAR03(50)	ICDAR03(FULL)	ICDAR03(Open)	ICDAR13
MSER + Otsu + re-extraction	**12.21**	**15.58**	**38.57**	**41.00**
MSER + Otsu	13.72	17.91	45.93	41.83
MSER	15.70	22.56	57.90	50.68
Otsu	21.51	26.28	54.65	54.16

Without re-detection	CILANEIS	PWP	tOLATON	GzRAG	factry
With re-detection	CLEANERS	PULP	KOLESTON	GARAGE	factory

Fig. 11: Differences of word recognition results with and without re-extraction. These examples are from ICDAR 2003/2013, SVT, and IIIT 5K-word datasets. The recognition results are obtained under an open vocabulary condition.

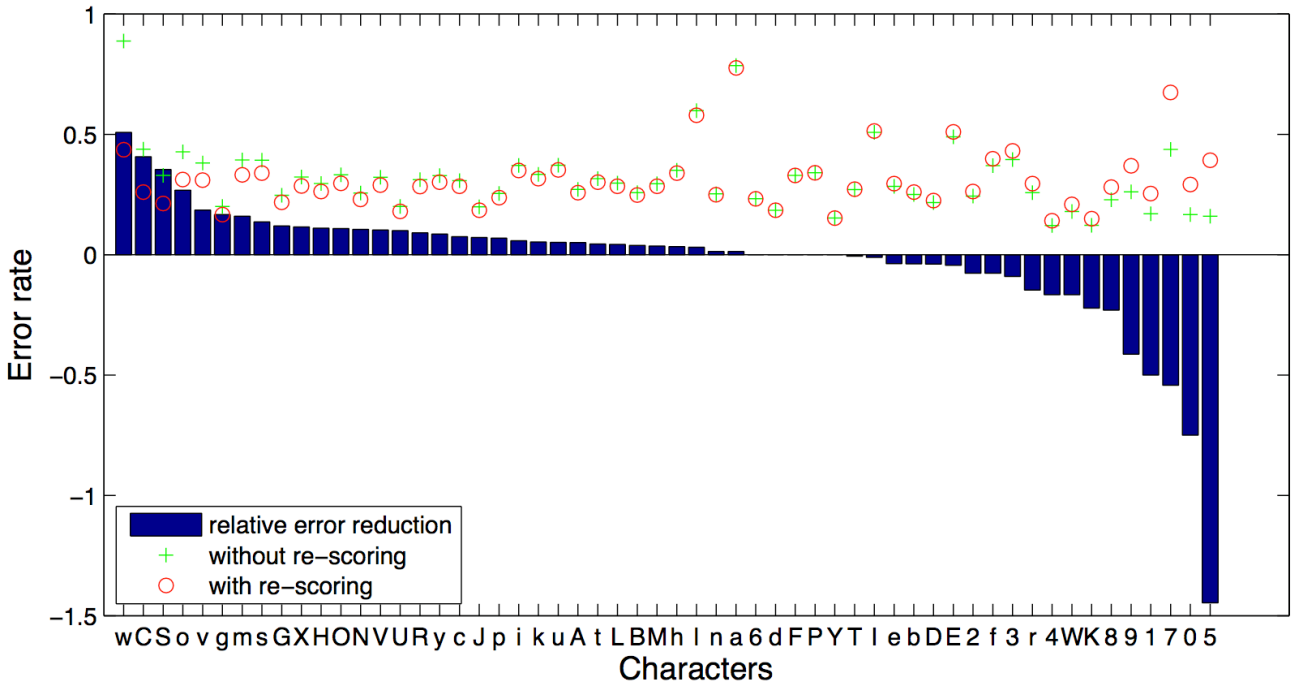


Fig. 12: Character wise recognition error rate on IIIT 5K-Word dataset. The error rates with and without re-scoring are shown by points and the relative error reduction by re-scoring is shown by bars. The graph is arranged in the order of relative error reduction.

6.4.3 Comparison with Other Methods

We compared our method with several competing word recognition methods. We first discuss the evaluation on the ICDAR 2013/2015 dataset. We followed the same evaluation protocol as other competitors for fair comparison. The results on the ICDAR 2013 dataset are shown in Table 6. The proposed method with re-scoring was ranked second among all methods tested according to the official metrics of the normalized edit distances to the correct answers, bested only by PhotoOCR [6]. Although Jaderberg et al. [14] also report the accuracy (90.8%), their evaluation protocol is different: ground truth lexicon is given and words with two or fewer characters or with non-alphanumeric characters were excluded.

Table 5: Word recognition error rates on ICDAR 2003 to show the effect of text line-based re-scoring.

Method	ICDAR03(50)	ICDAR03(FULL)	ICDAR03(Open)	ICDAR13
With re-scoring	**12.21**	**15.58**	**38.57**	**41.00**
Without re-scoring	13.02	16.40	52.91	47.58

Without re-scoring	wivenhOLe	GIFrs	vour	shoP	drts	ENr
With re-scoring	Wivenhoe	GIFTS	your	Shop	arts	ENT

Fig. 13: Differences of word recognition results with and without re-scoring. These examples are from ICDAR 2003/2013, SVT, and IIIT 5K-word datasets. The recognition results are obtained under open vocabulary condition.

Table 8: Word recognition error rates of proposed method and other methods on ICDAR 2003, SVT, and IIIT 5K-Word.

Method	ICDAR03(FULL)	ICDAR03(50)	SVT	IIIT5K(Large)	IIIT5K(Medium)	IIIT5K(Small)
Proposed	15.58	12.21	23.49	**51.53**	19.77	12.60
Wang et al. [41]	16	10	30	-	-	-
Novikova et al. [31]	17.2	-	27.1	-	-	-
Almazan et al. [2]	-	-	12.99	-	24.40	11.43
Gordo et al. [12]	-	-	-	-	13.4	6.7
Alsharif and Pineau [3]	11.4	6.9	25.7	-	-	-
Jaderberg et al. [14]	**1.4**	**1.3**	**4.6**	-	**7.3**	**2.9**
Bissacco et al. [6]	-	-	9.61	-	-	-
Yao et al. [46]	19.67	11.52	24.11	61.7	30.7	19.8
Lee et al. [22]	24	12	20	-	-	-
Shi et al. [34]	20.70	12.56	26.49	-	-	-
Mishra et al. [26]	-	19.72	26.43	55.70	42.50	31.75
ABBYY [1]	45	44	65	-	-	75.67

The results on the ICDAR 2015 dataset are shown in Table 7. Our method is behind the other three submitted methods, although there is no big difference in error rates. MAPS [20] and NESP [21] both perform segmentation first and the binarized image is processed using OCR software. DSM used a deep structured model with convolutional neural networks for local character scoring. It seems our method cannot extract reliable characters with simple MSER and Otsu binarization, because ICDAR 2015 datasets contain many ill-conditioned words with noise and blur. To improve the accuracy of our method for ill-conditioned text, we should use a more sophisticated character extraction method such as the binarization techniques used in NESP and MAPS.

The performances on the ICDAR 2003, SVT, and IIIT 5K-Word datasets are shown in Table 8. Our method outperforms Mishra et al. [27], Shi et al. [34], Yao et al. [46], Novikova et al. [31], and ABBYY [1] and is competitive with Wang et al. [41] and Lee et al. [22]. However, our method is still behind Almazan et al. [2], Gordo et al. [12], Alsharif and Pineau [3], Bissacco et al. [6], and Jaderberg et al. [14]. These methods take a whole word-based approach [2,12,14] and/or use deep neural networks [6,3]. This is discussed in detail later in this subsection.

Our method outperforms Mishra et al. and Shi et al. [26,34], which uses the similar cost function, which suggests that our candidate regions and accompanying classification score are better than these methods. There are two main reasons for this: (1) our two-stage character extraction achieves high recall with a small number of false alarms, and (2) the unary cost is computed by using four estimated lines, which improves character classification accuracy while reducing false positives.

Examples of successful recognition results are shown in Fig. 14(a). As we can see, the proposed method was able to recognize scene text with low resolution, different fonts, and distortions. Figure 14(b) shows the cases of incorrect recognition. The proposed method with the four-line model could not recognize words with a curved

Fig. 14: Examples of word recognition results of proposed method with four-line model. (a) Successfully recognized words. (b) Incorrectly recognized words.

text line, largely inclined characters, and unconstrained style.

Limitations of whole word-based approach. The performance of our method is behind Gordo et al. [12], Almazan et al. [2], and Jaderberg et al. [14], which recognize text from the whole word images. However these methods have some limitations. First, they face difficulty for cases without a lexicon, i.e., an open-vocabulary condition. Gordo et al. [12] and Almazan et al. [2] deal with the word recognition problem as a retrieval problem from a given limited lexicon. Jaderberg et al. [14] train the model with a pre-defined lexicon (about 90K words). The result is that words out of lexicon cannot be recognized by these methods. Since scene texts often contain words that are not listed in the dictionary and are impossible to know in advance, a method that can handle the open-vocabulary task is useful.

Second, these methods cannot handle significant changes in layout because they jointly and implicitly handle layout consistency (character alignment) in a holistic way. Since we explicitly handle layout consistency, we argue that our method is advantageous for use with significant changes in layout (e.g., from linear to curved) by explicit adaptation, whereas the holistic approaches require *re-learning*.

Deep neural networks. Bissacco et al. [6], Alsharif and Pineau [3], Wang et al. [41], and Jaderberg et al. [14] exploit deep neural networks and outperform our method. This is mainly due to the high performance of the CNN-based character classifier. For example, Alsharif and Pineau use convolutional maxout networks as a character classifier and achieve a significantly higher character classification performance, as shown in Table 2. Since our method is compatible with any character classifier, we believe it can achieve an even better performance if we replace our character classifier with a more powerful one. Moreover, since our re-extraction and re-scoring is separated from the final inference process, our method can be incorporated with other methods (if not the whole word-based method) to refine the candidate regions.

In addition, these methods require a huge amount of training images. Bissacco et al. use 2.2 million hand-labeled character images and Jaderberg et al. [14] use 9 million synthetic word images. Wang et al. [41] and Alsharif and Pineau [3] use about 130k character images while we use 27,715 characters. Although training images can be prepared by synthetic generation, as Wang et al. [41] and Jaderberg et al. [14] do, training deep neural networks with large datasets imposes a much lengthier training time and high computational power.

6.5 Curved Text

In this section, we provide an evaluation of the curved text recognition discussed in Section 4.4. We first judge if a given word is straight or curved. The selected style is then estimated and re-extraction and re-scoring are performed using the estimated style.

Figure 15 shows a comparison of the curved word recognition results by the four line-based method, which could not recognize curved words, and the four circle-based method, which could. We then measured the over-

Table 9: Comparison of word recognition performance between method with adaptive model selection and only four-line model. (a) Tested on full datasets. (b) Tested on the subset that contains curved cases only. (c) Tested on the subset that contains linear cases only.
(a) All cases

Dataset	Four-line model	Adaptive model
IIIT5K (Small)	12.23	11.13
IIIT5K (Medium)	19.10	17.77
IIIT5K (Large)	52.00	51.67
SVT	23.49	23.80
ICDAR03(50)	12.21	12.44
ICDAR03(FULL)	15.58	15.93

(b) Curved cases only

Dataset	Four-line model	Adaptive model
IIIT5K (Small)	49.10	29.70
IIIT5K (Medium)	61.21	38.79
IIIT5K (Large)	89.09	83.03

(c) Linear cases only

Dataset	Four-line model	Adaptive model
IIIT5K (Small)	49.84	49.84
IIIT5K (Medium)	16.54	16.61
IIIT5K (Large)	10.05	10.09

all performance on the IIIT 5K-Word, SVT, and ICDAR 2003 datasets. We counted the number of curved words manually for each datasets: IIIT 5K-Word and SVT contain 165 words (5.5%) and 10 words (1.55%), respectively, while ICDAR 2003 contains none. The method with adaptive model selection is compared with the method using only a four-line model. The performances on several datasets are shown in Table 9. In IIIT5K, we evaluate the performance tested on the subsets that contain curved cases or linear cases only, shown in Table 9(b), (c). The error rates were decreased in the IIIT 5K-Word full dataset and a significant error decrease can be seen in Table 9 (b), which shows the errors evaluated on curved cases only. Accuracies were largely unchanged in ICDAR 2003, which includes no curved cases, indicating that style selection is successful and performance does not decrease even if other styles are considered (Table 9(c) also supports this). Since curved cases in SVT were very difficult to recognize because of complicated background or sharp curves, we cannot recognize only one word among them, even with our adaptive model.

6.6 Computational Time

We analyzed the computational time of different components of the proposed method. The processing time for each procedure is summarized in Table 10. The processing time is dominated by the character classifier, which accounts for 96% of the overall time including first recognition and re-extraction. Although our additional stages of the pipeline appear to involve costly operations, every stage except for first recognition and re-extraction is computationally efficient. Moreover, our method reduces the number of classifier evaluations, as discussed in Section 6.3 (Fig. 10). If our character classifier is replaced with a more efficient one, the processing time will be reduced significantly and our method could be scalable to large datasets.

Table 10: Average computational time (per word) of different components of the proposed method.

Procedure	Processing time
Connected component-based extraction	0.031 sec
First recognition	1.30 sec
Line estimation	0.0016 sec
Sliding window re-extraction	1.59 sec
Line-based re-scoring	0.005 sec
CRF	0.080 sec

7 Conclusion

In this paper, we presented a method that can improve performance by conducting character extraction and word recognition in a bi-directional way. Our key novelty resides in the estimation of four lines of text string by character recognition results, not character extraction results. We utilize the estimated four lines to refine the character extraction and recognition results on the basis of the layout consistency of a text string. The benefits of this are that 1) we can take into account accurate geometric information to recognize character by re-scoring process and 2) the hybrid of CC-based and sliding window-based extraction achieves both computational efficiency and high recall of character extraction.

Although the final accuracy of our method was inferior to that of state-of-the-art methods, each component is easily convertible to another stronger component because our method does not depend on any specific character classifier, character extraction method, or cost function. Moreover, since our re-extraction and re-scoring is separated from the final inference process, our method can be incorporate with other methods independent of other components to refine the candidate regions.

Four lines	SrRE	RANAPA	ROSSQA	GWELSC
Four circles	STREET	GRANARY	PROSPECT	CHELSEA

Fig. 15: Differences of word recognition result between four-line model and four-circle model. These examples are from IIIT 5K-word datasets. The recognition results are obtained under an open vocabulary condition.

References

1. Abbyy finereader 9.0. http://www.abbyy.com.
2. J. Almazan, A. Gordo, A. Forn, and E. Valveny. Word Spotting and Recognition with Embedded Attributes. *PAMI*, pages 1–17.
3. O. Alsharif and J. Pineau. End-to-End text recognition with hybrid HMM maxout models. *ICLP*, oct 2013.
4. I. Bazzi, R. Schwartz, and J. Makhoul. An omnifont open-vocabulary ocr system for english and arabic. *PAMI*, 21(6):495–504, 1999.
5. Y. Bengio and Y. LeCun. Word normalization for on-line handwritten word recognition. In *IAPR*, 1994.
6. A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photocr: Reading text in uncontrolled conditions. In *ICCV*, 2013.
7. T. Caesar, J. M. Gloger, and E. Mandler. Estimating the baseline for written material. In *ICDAR*, 1995.
8. C.-C. Chang and C.-J. Lin. LIBSVM : A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2011.
9. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
10. T. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *VISAPP*, 2009.
11. V. Goel and W. Ecole. Whole is greater than sum of parts : recognizing scene text words. In *ICDAR*, 2013.
12. A. Gordo. Supervised mid-level features for word image representation. In *CVPR*, 2015.
13. W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural Network induced mser trees. In *ECCV*, 2014.
14. M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading Text in the Wild with Convolutional Neural Networks. *IJCV*, pages 1–20, 2014.
15. M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. *NIPS deep learning workshop*, pages 1–10, 2014.
16. M. A. Jones, G. A. Story, and B. W. Ballard. Integrating multiple knowledge sources in a bayesian ocr postprocessor. *ICDAR*, 1991.
17. D. Karatzas, L. Gomez-bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 Competition on Robust Reading. In *ICDAR*, 2015.
18. D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. I. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. de las Heras. ICDAR 2013 robust reading competition. In *ICDAR*, aug 2013.
19. V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, oct 2006.
20. D. Kumar, M. Prasad, and A. Ramakrishnan. Maps: midline analysis and propagation of segmentation. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, page 15. ACM, 2012.
21. D. Kumar, M. A. Prasad, and A. Ramakrishnan. Nesp: Nonlinear enhancement and selection of plane for optimal segmentation and recognition of scene word images. In *IS&T/SPIE Electronic Imaging*, pages 865806–865806. International Society for Optics and Photonics, 2013.
22. C.-Y. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu. Region-based discriminative feature pooling for scene text recognition. In *CVPR*, 2014.
23. S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. In *ICDAR*, number Icdar, 2003.
24. J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, sep 2004.
25. E. G. Miller and P. A. Viola. Ambiguity and constraint in mathematical expression recognition. In *AAAI/IAAI*, pages 784–791, 1998.
26. A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012.
27. A. Mishra, K. Alahari, and C. V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *CVPR*, 2012.
28. L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *ACCV*, 2010.
29. L. Neumann and J. Matas. Text localization in real-world images using efficiently pruned exhaustive search. In *ICDAR*, 2011.
30. L. Neumann and J. Matas. Real-time scene text localization and recognition. In *CVPR*, 2012.
31. T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *ECCV*, 2012.
32. N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 1975.
33. P. Sarkar and G. Nagy. Style consistent classification of isogenous patterns. *PAMI*, 2005.
34. C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang. Scene text recognition using part-based tree-structured character detection. In *CVPR*, 2013.
35. B. Su, S. Lu, S. Tian, J.-H. Lim, and C.-L. Tan. Character recognition in natural scenes using convolutional co-occurrence hog. In *ICPR*, 2014.
36. J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.
37. C. Thillou, S. Ferreira, and B. Gosselin. An embedded application for degraded text recognition. *EURASIP Journal on applied signal processing*, 2005:2127–2135, 2005.

38. S. Tian, S. Lu, and B. Su. Scene text recognition using co-occurrence of histogram of oriented gradients. In *ICDAR*, 2013.
39. K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011.
40. K. Wang and S. Belongie. Word spotting in the wild. In *ECCV*, 2010.
41. T. Wang, D. J. Wu, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *ICPR*, 2012.
42. J. Weinman and E. Learned-Miller. Improving recognition of novel input with similarity. *CVPR*, 2006.
43. J. J. Weinman. Typographical features for scene text recognition. *ICPR*, 2010.
44. J. J. Weinman, Z. Butler, D. Knoll, and J. Feild. Toward integrated scene text reading. *PAMI*, 2014.
45. J. J. Weinman, E. Learned-miller, and A. R. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *PAMI*, 2009.
46. C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets : A learned multi-scale representation for scene text recognition. In *CVPR*, 2014.