Recognizing Text with Perspective Distortion in Natural Scenes

Trung Quy Phan[1*], Palaiahnakote Shivakumara[2+], Shangxuan Tian[1*] and Chew Lim Tan[1*]
[1]School of Computing, National University of Singapore
[2]Faculty of Computer Science and Information Technology, University of Malaya
[*]{phanquyt, tians, tancl}@comp.nus.edu.sg, [+]shiva@um.edu.my

Abstract

This paper presents an approach to text recognition in natural scene images. Unlike most existing works which assume that texts are horizontal and frontal parallel to the image plane, our method is able to recognize perspective texts of arbitrary orientations. For individual character recognition, we adopt a bag-of-keypoints approach, in which Scale Invariant Feature Transform (SIFT) descriptors are extracted densely and quantized using a pre-trained vocabulary. Following [1, 2], the context information is utilized through lexicons. We formulate word recognition as finding the optimal alignment between the set of characters and the list of lexicon words. Furthermore, we introduce a new dataset called StreetViewText-Perspective, which contains texts in street images with a great variety of viewpoints. Experimental results on public datasets and the proposed dataset show that our method significantly outperforms the state-of-the-art on perspective texts of arbitrary orientations.

1. Introduction

Reading text in natural scene images refers to the problem of recognizing words that appear on, e.g., bill boards and road signs. If such words can be recognized, they can be used for a wide range of applications: content-based image retrieval, sign translation, intelligent driving assistance, and navigation aid for the visually-impaired and robots. Partly due to this reason, scene text recognition has received increased interests from the community, e.g., [1–4], in recent years.

In this paper, we focus on text recognition in street images, which facilitates the application of business name search on online maps [1]. This application is motivated by the availability of ground-level, 360° views of various locations on Google Maps and Microsoft Bing Maps. These geo-tagged images contain useful text information, e.g., business names and addresses. The large-scale nature of street image data provides an exciting opportunity to benefit millions of users.

However, scene text recognition is very challenging due to three main problems. First, the appearances of scene characters are almost unconstrained, i.e., they vary drastically in fonts, colors and sizes. Moreover, the characters typically appear on cluttered backgrounds. Second, scene characters often suffer from various deformations such as uneven illumination, blurring and perspective distortion. Third, in complex scenes such as street images, text may not be the main object. Other objects such as buildings, cars and pedestrians can interfere with the recognition process.

As an illustration of the complexity of street images, the recognition accuracy of Optical Character Recognition (OCR) engines on words cropped from these images is as low as 35% [1]. For comparison, the typical OCR accuracy on scanned documents is more than 90%.

Although there are existing works to recognize text in natural scene images, e.g., [1–4], their scopes are limited to horizontal texts which are frontal parallel to the image plane. However, in practice, scene texts can appear in any orientation, and with perspective distortion. Thus, the important issue of handling perspective texts has been neglected by previous works. In this paper, we attempt to address the recognition of perspective texts of arbitrary orientations in complex scenes (such as street images).

Using a traditional visual feature such as Histogram of Oriented Gradients (HOG) (as employed in [1, 2]) would lead to a low accuracy on perspective texts. The reason is that the feature is not able to handle the different character poses. To deal with this problem, one approach is to train a classifier on discretized poses of individual characters. However, the major drawback of this approach is that it is labor-intensive and time-consuming to collect enough training samples for a large number of character classes (62 classes for English characters and digits), each with, say, 10 discrete poses. In addition, when collecting character samples from natural scenes, it is difficult to control the character poses accurately.

Hence, we take a different approach and use Scale Invariant Feature Transform (SIFT) in a bag-of-keypoints approach. Because SIFT is robust to both rotation and viewpoint change, our system is trained on **only frontal characters** (from commonly used datasets in the literature such as ICDAR 2003 [5]). Our extensive experiments show that this approach achieves good accuracies, while avoiding the high cost of collecting samples of perspective characters.

Following recent works [1, 2], the scope of this paper is limited to cropped word recognition with a lexicon, i.e., a list of words of interest. The lexicon serves as a form of context information, and is especially relevant for the application of business name search. Given a street image and its address, the lexicon can be built by collecting the shop names around the address via a search engine [1]. There are also other

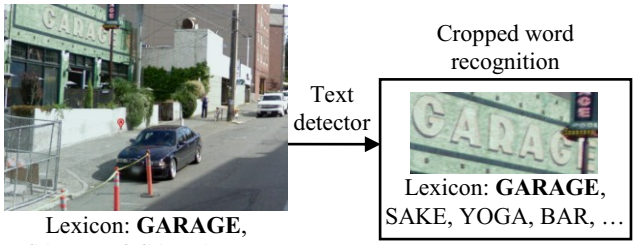


Figure 1: The problem of cropped word recognition. A "cropped word" refers to the region cropped from the original image based on the word bounding box returned by a text detector, e.g., [10]. Given a cropped word image, the task is to recognize the word using the provided lexicon.

applications where such a lexicon is available. Ballan et al. [6] used a list of soccer players' names for text recognition in sports videos. Graves et al. [7] constructed a list of the most common English words for handwriting recognition. Another example is the list of products in a supermarket, which can be used for the application of aiding the visually-impaired. Figure 1 illustrates the problem setting.

Our contributions are as follows. (1) We present an approach to recognize perspective scene texts of arbitrary orientations. This issue is of great practical importance, but has been neglected by most previous works. (2) Our system is trained on only frontal characters, which drastically reduces the cost of collecting training data. (3) For performance evaluation, we introduce a new dataset called StreetViewText-Perspective, which contains texts in street images with a variety of viewpoints. On this dataset, our method compares favorably to the state-of-the-art.

2. Related work

A comprehensive review of text extraction methods is provided in [8]. In general, there are two main steps: text detection and text recognition. The first step aims to locate the text positions in an image, usually by drawing a bounding box around each word. The typical features used for detection are stroke widths [9, 10], Maximally Stable Extremal Regions (MSER) [11, 12] and HOG [1]. In the second step, the detected words are recognized into text strings. We discuss some of the recent works below.

Character recognition. A building block for word recognition is individual character recognition. The typical features used for recognition are HOG [1, 2] and Gabor filters [13]. More recently, unsupervised feature learning has been explored in [14, 15]. However, since these features are not robust to rotation and viewpoint change, they may not work well for perspective characters of arbitrary orientations.

Frontal word recognition. Weinman et al. [13] and Smith et al. [3] proposed a novel similarity constraint to force characters which were visually similar to take the same label. However, these methods were only tested on simple sign images where most of the words appeared on plain backgrounds. Wang et al. [1, 16] adopted an object recognition framework for word recognition. These methods require all characters of a word to be correctly recognized and thus, they cannot handle cases where one or more characters are occluded. Recent works formulate word recognition as an optimization problem by using Conditional Random Field (CRF) [2, 17], Viterbi alignment [15] and weighted finite-state transducers [4]. However, these works were only tested on frontal texts.

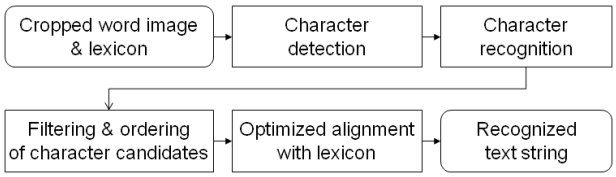


Figure 2: The flowchart of the proposed method.

Perspective text recognition. One approach is to rectify perspective texts prior to recognition, e.g., [11, 18, 19]. However, these methods rely heavily on the quality of the binarized character shapes. Thus, although they work for texts on plain backgrounds, it is unclear whether they can handle texts with cluttered backgrounds (as in street images). Gandhi et al. [20] rectified perspective texts in image sequences by utilizing the motion information. This method requires camera calibration, and does not work for still images. In a recent work, Li et al. [21] recognized perspective characters without rectification. However, this work only focused on character recognition, and did not address word recognition. The dataset was also limited to simple sign images. Therefore, despite its importance, the recognition of perspective texts has not been adequately addressed.

3. Character detection and recognition

An overview of our approach to perspective text recognition is shown in Figure 2. We describe the detection and recognition of characters below. The optimized alignment of the recognized characters with the lexicon will be discussed in the next section.

3.1. Detection of character candidates

In the first step, we use MSERs [22] to detect the potential character locations in a cropped word image (hereafter referred to as *character candidates*). The main idea of MSER is to identify regions which remain stable over a range of thresholds on the intensity values. It has been shown that scene characters can be extracted as MSERs [11, 12]. MSERs are also robust to viewpoint change [23]. Hence, they are suitable for perspective characters.

However, not all the extracted MSERs from a cropped word correspond to characters. Thus, we classify them into text MSERs and non-text MSERs using four features: the relative height, the aspect ratio, the number of holes and the number of horizontal crossings [11, 12]. The text MSERs are retained while the non-text MSERs are discarded.

In [11, 12], the text MSERs were directly used for text detection. However, in this paper, we use the MSER bounding boxes instead. The reason is that although MSERs provide a useful initial segmentation of the characters, they do not always correspond to the whole characters. Figure 3 shows an

(a) Cropped word image (b) MSERs

(c) Character candidates based on MSER bounding boxes

Figure 3: Character detection based on MSERs. For better illustration, only the non-overlapping MSERs are shown in (b). The handling of overlapping MSERs will be discussed later.

example where the MSERs corresponding to 'E' and 'S' have incomplete shapes. Therefore, using the MSER bounding boxes as character candidates helps to recover some of the missing parts (if any) of the characters.

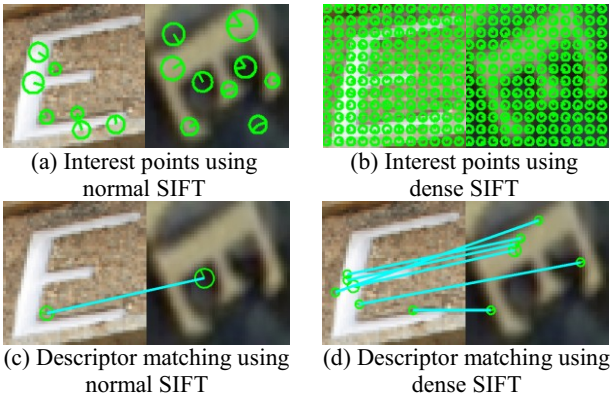
3.2. Estimation of character probabilities

For each character candidate detected in the previous section, we need to estimate the probability that it takes character label l_u. In this paper, we focus on English characters: $l_u \in L$ where $L = \{A, \dots, Z, a, \dots, z, 0, \dots, 9\}$. Formally, we would like to estimate $P(l_u|c_i)$, the probability that c_i, the i^{th} character candidate, takes label l_u.

As mentioned before, our goal is to train the system on only frontal characters (to reduce the cost of collecting training data). This requires the features extracted from the character candidates to be robust to rotation and viewpoint change. Thus, we propose to use SIFT. SIFT has been explored for text recognition in [24, 25] and for word spotting in [26, 27]. The first two works extracted the descriptors only at sparse interest points, which is not sufficient for perspective characters (to be explained later). The last two works were only tested on frontal scanned document images. In contrast, we adopt dense SIFT (which was used for scene classification in [28]) for perspective character recognition.

More specifically, the patch inside a character candidate is normalized to a fixed size of 48 × 48. We use a grid with spacing of 2 pixels. At each grid point, we extract SIFT descriptors at multiple scales. Note that only the locations and the scales are fixed. The dominant gradient directions of the descriptors may vary across different grid points, as well as across different scales at the same grid point. (In the literature, the term "dense SIFT" sometimes refers to an extraction scheme where the orientations of the dense interest points are fixed. However, we allow them to vary to ensure the rotation-invariance of the descriptors.)

The rationale for using dense SIFT is that it provides more information to discriminate among a large number of classes (62 character classes). With the original SIFT, the descriptors are only extracted at sparse interest points. However, scene characters typically suffer from deformations, e.g., blurring and uneven illumination, which reduce the number of detected interest points. More importantly, instances of the same class often suffer from different types of deformations. Thus, the sets of detected interest points are not consistent, which negatively affects the matching of the descriptors. Figure 4 shows that using dense SIFT helps to overcome the above problems.



(a) Interest points using normal SIFT (b) Interest points using dense SIFT

(c) Descriptor matching using normal SIFT (d) Descriptor matching using dense SIFT

Figure 4: Using normal SIFT leads to few descriptor matches. In contrast, dense SIFT provides more information for character recognition. The left image in each pair is from the training set while the right one is from the test set. Note the fact that the right one is a rotated character. For better illustration, in (b), we only show one scale at each point. (Best viewed in color.)

Moreover, instead of matching the descriptors directly, we use a bag-of-keypoints approach [29]. By ignoring the spatial information of the keypoints, this approach allows for more distortion between the training and the testing samples. K-means clustering is used to build a vocabulary of 3,000 visual words from a random subset of (dense) SIFT descriptors extracted from training samples. (Section 6.4 shows how the recognition accuracy varies with respect to the vocabulary size.) With this vocabulary, the descriptors of a character candidate are assigned to the nearest clusters. The feature representation then becomes a histogram which counts the number of descriptors belonging to each visual cluster. We use a standard SVM package [30] with Histogram Intersection Kernel [31] to estimate $P(l_u|c_i) \in [0, 1]$. (The training and the testing data are described in Section 6.)

The fact that our method recognizes perspective characters directly is an advantage over methods which rely on rectification such as [11, 19]. The rectification process is error-prone due to the challenges of scene characters, including blurring and cluttered backgrounds.

3.3. Non-maximal suppression

Since multiple MSERs may be detected for the same character [12], we perform non-maximal suppression on the set of character candidates. A character candidate is suppressed if it has a significant overlap with another character candidate and the latter has a higher confidence. The overlap ratio is calculated based on the two MSER areas. The confidence of a character candidate is defined as the maximum character probability:

$$Confidence(c_i) = \max_{l_u \in L} P(l_u|c_i) \quad (1)$$

After suppression, the remaining character candidates are fed into the next step for word recognition.

MSER and SIFT have been used separately for character detection and character recognition in previous works.

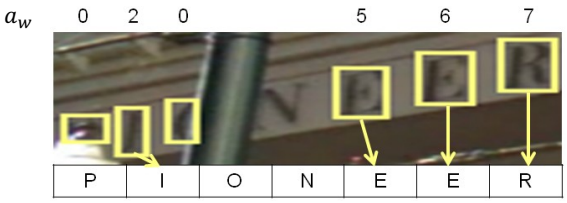


Figure 5: A sample alignment between a set of 6 character candidates (shown in yellow) and the word "PIONEER". The top row shows the value of the alignment vector (of length 6).

However, to the best of our knowledge, this paper is the first attempt to combine them in a coherent way to recognize perspective characters while using only frontal training data.

4. Word recognition

The recognition of perspective texts of arbitrary orientations is much more difficult than that of frontal, horizontal texts due to additional challenges. With arbitrary orientation, it is difficult to distinguish characters such as '6' and '9', and 'u' and 'n', unless there is context information. Furthermore, some characters may be hard to read (due to severe distortions) or even occluded. To deal with these problems, we use a lexicon as the context information. We formulate word recognition as finding the optimal alignment between the character candidates and the lexicon words.

Let W denote the lexicon of an image. Let $C = \{c_1, \dots, c_n\}$ be the set of character candidates. Each character candidate can take a label from $L \cup \{\varepsilon\}$, where $L = \{A, \dots, Z, a, \dots, z, 0, \dots, 9\}$ and ε is the empty label. Let a_w denote an alignment vector of C to a word $w \in W$. $a_w(i) = k$ $(k > 0)$ represents that c_i is aligned with $w(k)$, the k^{th} character of string w. $a_w(i) = 0$ indicates that c_i takes the empty label.

For example, in Figure 5, $a_{PIONEER}(6) = 7$ indicates that c_6 is aligned with the 7th character ('R'). $a_{PIONEER}(1) = 0$ and $a_{PIONEER}(3) = 0$ because c_1 and c_3 take the empty label. Note that for this image, some of the characters are missed ('N') or partially detected ('P' and 'O'). However, the alignment vector still allows for a flexible matching.

We also define $AlignScore(a_w)$ to be the alignment score, which measures how well the labels of the character candidates match the word w (to be explained more later). The optimal word w^* can then be found as follows:

$$w^* = \underset{w \in W}{argmax} \quad MaxWordScore(w) \tag{2}$$

where

$$MaxWordScore(w) = \underset{a_w \in A_w}{max} \quad AlignScore(a_w) \tag{3}$$

A_w denotes the set of all the possible alignments between the character candidates and word w.

Intuitively, Equations (2) and (3) mean that for each word in the lexicon, we compute its maximum alignment score. Then, among all the lexicon words, the one with the highest maximum alignment score is returned as the optimal word.

4.1. Ordering of character candidates

Our alignment algorithm requires the character candidates to be ordered into a sequence. For simplicity, we assume that text is written from left to right or from top to bottom. If a word is nearer to the horizontal orientation, the character candidates are ordered by the x-coordinates. Otherwise, they are ordered by the y-coordinates. A word is classified as either nearer to the horizontal orientation or nearer to the vertical orientation based on the angle of the major axis of its bounding quadrilateral. (For perspective words, we use quadrilaterals to mark the word locations (Section 5).)

4.2. Alignment score

As mentioned before, the alignment score measures how well the labels of the character candidates match a word. It is computed based on the individual scores of the character candidates. Let $Score(c_i, l_u)$ be the score of assigning label l_u to character candidate c_i:

$$Score(c_i, l_u) = \begin{cases} P(l_u|c_i), & if\ l_u \neq \varepsilon \\ 1 - Confidence(c_i), & if\ l_u = \varepsilon \end{cases} \tag{4}$$

If a character candidate takes a non-empty label, we directly use the corresponding SVM probability. Otherwise, if it takes the empty label, we use a penalty score (inspired by [2]). The purpose of the penalty score is to discourage character candidates with high confidence from taking the empty label.

The alignment score of the whole word is the sum of the individual scores of the character candidates:

$$AlignScore(a_w) = \sum_{i=1}^{n} Score\left(c_i, w\left(a_w(i)\right)\right) \tag{5}$$

Recall that $a_w(i)$ is the index of string w that c_i is aligned to. Thus, $w\left(a_w(i)\right) \in L$ is the label assigned to c_i.

4.3. Optimized alignment algorithm

Equation (2) is implemented by looping through the words in the lexicon. For each word w, we need to compute $MaxWordScore(w)$ (Equation (3)). The rest of this section describes our optimized alignment algorithm for doing this. Since w is fixed in Equation (3), we drop w in some of the below notations for clarity.

Let $F(c_i, p)$ be the optimal alignment score of character candidates $c_i, c_{i+1}, \dots, c_n$, with c_i aligned at index p of w. F can be computed using dynamic programming. The initialization is as follows:

for $i = n$ down to 1
  for $p = len(w)$ down to 1
    $F(c_i, p) = Score\left(c_i, w(p)\right) + \sum_{k=i+1}^{n} Score(c_k, \varepsilon)$ (6)
  end for
end for

Intuitively, Equation (6) means that only c_i is assigned a non-empty label while $c_{i+1}, \dots, c_n$ are assigned the empty label. Hence, we use the score of c_i and add the penalty scores of $c_{i+1}, \dots, c_n$. After that, we update F backwards:

for $i = n$ down to 1
  for $p = len(w)$ down to 1

$$\textbf{if } F(c_i, p) < \max_{j,q} \begin{pmatrix} F(c_j, q) + \sum_{k=i+1}^{j-1} Score(c_k, \varepsilon) \\ + Score(c_i, w(p)) \end{pmatrix} \quad (7)$$

    update $F(c_i, p)$ to $\max_{j,q}(.)$
  end if
  end for
end for

where $j \in [i+1,\ n]$ and $q \in [p+1,\ len(w)]$. c_j can be thought of as the first character candidate with a non-empty label after c_i. Intuitively, the right hand side (RHS) of Equation (7) means that we loop through the combinations of j and q. For each combination of j and q:

- We use $F(c_j, q)$, the optimal alignment score for $c_j, \dots, c_n$, as the starting point. (Note that because we compute F backwards, $F(c_j, q)$ has already been computed, and thus we can use its value.)
- We then add the penalty scores of assigning the empty label to $c_{i+1}, \dots, c_{j-1}$.
- Finally, we add the score of c_i.

After we have looped through all the combinations of j and q, if the RHS of Equation (7) is greater than the initialized value of $F(c_i, p)$ (in Equation (6)), we update the value of the latter to the former.

In our implementation, to reduce the computational time, we restrict the range of p based on c_i's relative position in the image. For example, if c_i is near the left boundary of the image, p's range can be restricted to only the first few indices of w. q's range can also be restricted in a similar way.

When all F's have been computed, $MaxWordScore(w)$ in Equation (3) is obtained by:

$$MaxWordScore(w) = \max_{i,p} \left(F(c_i, p) + \sum_{k=1}^{i-1} Score(c_k, \varepsilon) \right) \quad (8)$$

The intuition of Equation (8) is that we loop through the combinations of i and p. For each combination of i and p, c_i acts as the first character candidate with a non-empty label (among all the detected character candidates). Hence, we use the score of c_i and add the penalty scores of $c_1, \dots, c_{i-1}$.

Our optimized alignment algorithm has a few advantages over existing works. First, it explicitly allows the empty label, and thus is able to handle cases where one or more characters are missed or occluded. This is an advantage over [1], which does not allow skipping characters. Second, many methods, e.g., [1, 4], require normalization for word length to avoid bias towards shorter words. In contrast, because the magnitude of our alignment score depends on the number of character candidates (and not on the lexicon word length), no normalization is required.

5. StreetViewText-Perspective dataset

Most of the standard datasets for scene text recognition, e.g., [1, 5, 13, 17, 32], are limited to frontal texts. For example, the annotators of the Street View Text (SVT) dataset were instructed to "minimize skew" when choosing the angles of texts [1]. Recently, there are more challenging datasets: NEOCR [33] and MSRA-TD500 [10], which include texts of arbitrary orientations and perspective texts. However, these two datasets are still not ideal for evaluating perspective text recognition for two reasons. First, because they are not specifically designed for perspective texts, many of the words in these datasets are still frontal. Second, in practice, we often have the context information, e.g., the GPS location and the address, of the image, which can be used to improve text recognition. However, this information has been discarded in these datasets.

Hence, although we do include MSRA-TD500 in our experiments, we also introduce a new dataset called StreetViewText-Perspective (*SVT-Perspective*)[1], which is specifically designed for evaluating perspective text recognition. It also preserves the address, i.e., the context information, of the input images. Our dataset is built based on the original SVT dataset [1] for two reasons. (SVT is a public dataset that contains images taken from Google Street View with frontal texts of shop names, street names, etc.)

First, we would like to reuse the lexicons in SVT, which were collected by Amazon Mechanical Turk workers. As a consequence, our dataset contains images taken at the same addresses on Google Street View. However, instead of choosing the frontal texts, we intentionally picked side-view angles such that texts are still readable to humans. The lexicon of each image was taken to be the same as that of the corresponding SVT image. Second, as our images were taken at the same locations, they allow for a meaningful analysis of the degradation in performance between frontal and perspective texts.

For each image in our dataset, the words were manually annotated using quadrilaterals. Similar to SVT, we only annotated the words that were present in the image-specific lexicons. Figure 6 shows a comparison of an image from SVT and an image from SVT-Perspective.

Our dataset contains 238 images, which correspond to the images in the SVT test set. The number of cropped words is 639. The words are of a variety of viewpoints and orientations. Their heights vary from 9 to 330 pixels.

6. Experimental results

We performed experiments on perspective texts, texts of arbitrary orientations and frontal texts. For the first class of texts, we used our own dataset for reasons explained in the previous section. For the second class of texts, we picked MSRA-TD500 [10] because it is a very recent dataset that is specifically designed for texts of arbitrary orientations. (Note that in terms of size, NEOCR [33] is larger than MSRA-TD500. However, it also contains more languages. The

[1] Available at http://www.comp.nus.edu.sg/~phanquyt/

Lexicon: **PICKLES**, **PUB**, HOTEL, INN, …
(a) SVT image

Lexicon: **PICKLES**, **PUB**, HOTEL, INN, …
(b) SVT-Perspective image

Figure 6: An image from SVT and the corresponding image from SVT-Perspective. Both images are taken at the same address, and thus have the same lexicon. In (b), the bounding quadrilaterals are shown in black for "PICKLES" and "PUB".

English subsets of these two datasets, which are our focus in this paper, are comparable in size [10, 33].) For the third and final class of texts, among the various datasets that have been used in the literature [13, 17, 32], we chose ICDAR 2003 [5] and SVT [1] because they are the most widely used datasets with many reported results.

MSRA-TD500 only contains annotations at the text line level. Thus, to evaluate word recognition, we manually added word-level annotations[2] for the English words in this dataset (denoted as *MSRA-TD500-Word*).

For ICDAR 2003, we used the benchmarks for character recognition (*ICDAR-Char*) and word recognition (*ICDAR-Word*). For SVT, we used both the original word-level annotations (*SVT-Word*) and the character-level annotations provided in [2] (*SVT-Char*).

Due to the large number of visual words used (Section 3.2), we need to collect enough data to train the character classifier. We used samples from ICDAR-Char (only the training subset) and two other public datasets for frontal texts: Weinman's dataset [13] and Chars74k [32] (only the English subset). In total, we had 19,800 training samples. This training set was used for all the experiments in this section.

6.1. Experiment on perspective texts

In this experiment, we used our SVT-Perspective dataset for evaluation. We obtained the source codes of [1] and [15] from the authors' websites. We also re-implemented [2] following the descriptions in the paper, and included ABBYY FineReader 9.0 [34], a commercial OCR engine, in the comparative study. We used the same experimental settings as [1, 2]. In particular, words with less than 3 characters or containing non-alphanumeric characters were ignored.

The second column of Table 1 shows that our method significantly outperformed the other methods. The increase in accuracy from 45.7% (of [2]) to 62.3% (of our method) represents a relative improvement of 36%. In Figure 10 (on the last page), our method recognized the words correctly despite the blurring, occlusion and large variation in text appearance. (Due to space constraint, additional results are given in Figure 11 in supplementary material.)

[2] Available at http://www.comp.nus.edu.sg/~phanquyt/

Table 1. Recognition accuracy on perspective words (in %)

Method	SVT-Perspective-Word	SVT-Perspective-Word (Full)
FineReader 9.0 [34]	16.9	9.7
K. Wang et al. [1]	40.5	26.1
Mishra et al. [2]	45.7	24.7
T. Wang et al. [15]	40.2	32.4
Our method	**62.3**	**42.2**

Table 2. Accuracy on words of arbitrary orientations (in %)

Method	MSRA-TD500-Word (Full)
FineReader 9.0 [34]	23.2
K. Wang et al. [1]	44.5
Mishra et al. [2]	27.8
T. Wang et al. [15]	20.8
Our method	**58.4**

AZONA EXPO HYDRANT TARGET

Figure 7: Sample recognition results of our method for words of arbitrary orientations.

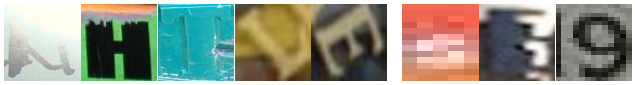
These results demonstrate the advantage of using dense SIFT to recognize perspective texts of arbitrary orientations. Our optimized alignment algorithm also contributes to the handling of characters that are occluded or hard to read. Moreover, we have shown that using **only frontal characters** for training is a sensible and realistic approach because it avoids the cost of collecting perspective character samples.

We would like to emphasize that our training data did **not** contain any samples from SVT. The data came from other datasets, as aforementioned. Thus, the successful recognition of the perspective words is due to the generalization power of dense SIFT and SVM (and **not** because of the similarity between SVT and SVT-Perspective).

We also analyzed how the recognition accuracy varied with the lexicon size. Intuitively, a larger lexicon makes it more difficult to recognize a word, especially if there are several similar words in the lexicon. In addition to the original lexicon size (of around 50 words per image), we used another lexicon size denoted as *Full*. This lexicon contained 377 words (an increase of 7.5 times in size) and was constructed by putting all the ground truth words in the test set into a list (following the procedure in [1]). The third column of Table 1 shows that even when a larger lexicon was used, our method still achieved the best accuracy.

6.2. Experiment on texts of arbitrary orientations

In this experiment, we ran the same set of methods on MSRA-TD500-Word. Since this dataset does not have lexicons, we constructed a *Full* lexicon of 395 words in a similar way as in the previous section. Table 2 shows that our method also significantly outperformed the other methods on this dataset. The increase in accuracy from 44.5% (of [1]) to 58.4% (of our method) represents a relative improvement of 31%. Figure 7 shows sample results of our method.



(a) Success cases (b) Failure cases

Figure 8: Sample character recognition results of our method. In (a), the characters were correctly recognized despite the strong highlight, small occlusion, similar text and background colors, and rotation. In (b), the characters were wrongly recognized due to low resolution, strong shadow and rotation invariance. The last character was recognized as '6'.

Figure 9: Sample results of our method for frontal words. It was able to recognize the words under challenging scenarios: transparent text, occlusion, fancy font, similar text and background colors and strong highlight.

We are the first to report the recognition accuracy on MSRA-TD500, a very recent public dataset for text of arbitrary orientations. The fact that our method performed well on both SVT-Perspective and MSRA-TD500 demonstrates its advantage over existing methods.

6.3. Experiment on frontal texts

In the previous sections, we do not evaluate cropped character recognition because SVT-Perspective and MSRA-TD500 do not have character-level annotations. In contrast, ICDAR and SVT do have annotations at the character level. Thus, we compared the character recognition accuracy (using 62 classes) on cropped character images. We have also included the results reported by other recent works.

Table 3 shows that on SVT-Char, our method achieved the state-of-the-art accuracy of 67.0%. The previous best known result on this dataset was 61.9% [2].

On ICDAR-Char, our method outperformed [1]. Although its accuracy was lower than those of [14, 15], these results should be interpreted with the following consideration: when rotation-invariant features are used, it is difficult to distinguish pairs of characters such as 'u' and 'n', and '6' and '9', especially because no context information is available at the character level. Therefore, because our method uses rotation-invariant features, it is at a slight disadvantage compared to the other methods, including [14, 15]. Figure 8 shows sample results of our method.

The next experiment is on word recognition. Each image in SVT-Word comes with a lexicon (of around 50 words). On the other hand, ICDAR-Word does not have lexicons. For fair comparison, we used the lexicons provided in [1]. Table 4 shows that on SVT-Word, our method achieved the best recognition accuracy. Our accuracy of 73.7% is slightly higher than the previous best known result of 73.6% [17]. On ICDAR-Word, our method achieved the third best accuracy. Sample results of our method are shown in Figure 9.

Furthermore, the fact that the images in SVT and SVT-Perspective were taken at the same addresses on Google Street View allows for an analysis of the performance degradation between frontal and perspective texts. The drop in accuracy of our method (-15.5%) was significantly lower than those of the other methods (Table 5). This shows that our method is more robust against rotation and perspective distortion, which is important for practical applications.

Table 3. Cropped character recognition accuracy (in %)

Method	ICDAR-Char	SVT-Char
FineReader 9.0 [34]	21.0	11.7
K. Wang [1]	64.0	N.A
Mishra [2]	N.A	61.9
Coates [14]	81.7	N.A
T. Wang [15]	**83.9**	N.A
Our method	75.6	**67.0**

Table 4. Recognition accuracy on frontal words (in %)

Method	ICDAR-Word	SVT-Word
FineReader 9.0 [34]	56.0	35.0
K. Wang [1]	76.0	57.0
T. Wang [15]	**90.0**	70.0
Novikova [4]	82.8[3]	72.9
Mishra [2]	81.8	73.3
Mishra [17]	80.3	73.6
Our method	82.2	**73.7**

Table 5. Degradation in performance between frontal texts and perspective texts (in %)

Method	SVT-Word	SVT-Perspective-Word	% change
FineReader [34]	35.0	16.9	-51.7
K. Wang [1]	57.0	40.5	-28.9
Mishra [2]	73.3[4]	45.7[5]	-37.7
T. Wang [15]	70.0	40.2	-42.6
Our method	**73.7**	**62.3**	**-15.5**

6.4. Additional experiments

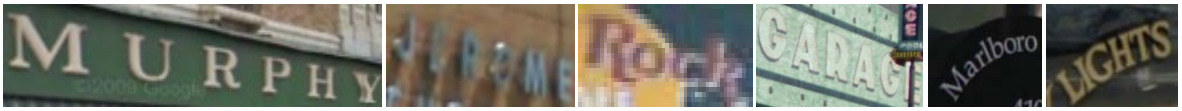
On SVT-Perspective (with the original lexicons), the average processing time of our unoptimized Matlab code was 38.6 seconds. This was measured on a machine with Intel Core i5 processor (quad-core, 3.2 GHz) and 4 GB RAM. Our code can be optimized by e.g., using a trie structure for the lexicon to avoid redundant computation [1].

We also analyzed how our accuracy varied with respect to the number of visual words (Section 3.2). This experiment is included in the supplementary material.

7. Conclusion

We have described a method for recognizing perspective scene texts of arbitrary orientations. Our work serves as a step towards practical applications (of scene text extraction)

[3] Achieved using a slightly larger lexicon for ICDAR-Word.
[4] Taken from [2].
[5] Obtained from our re-implementation of [2], which follows the paper closely. Its accuracy on SVT-Word was 69.1%, which is close to the 73.3% accuracy reported in [2] for the same dataset.



K. Wang [1]	COFFEES	PACIFIC	THIRTEEN	FIRST	CENTER	ADLER
Mishra [2]	SQUARE	TOURS	CREAM	SAKE	BAR	SAN
T. Wang [15]	INC	NEW	THE	FRY	PMS	SAN
Our method	**MURPHY**	**JEROME**	**ROCK**	**GARAGE**	**MARL-BORO**	**LIGHTS**

Figure 10: Sample recognition results for texts of arbitrary orientations and with perspective distortion.

in two aspects. First, most existing works make the simplistic assumption that text is horizontal and frontal parallel to the image plane. However, in many real-world scenarios, this assumption does not hold. Thus, by handling perspective texts, this work has attempted to address an important research gap. Second, an attractive feature of our method is that it is trained on only frontal character samples, and thus does not require collecting samples of perspective characters. This drastically reduces the cost of data collection.

The second aspect is achieved by the use of dense SIFT in a bag-of-keypoints framework, which is robust to rotation and viewpoint change. Our optimized alignment algorithm is also designed to handle the challenges of perspective texts, e.g., one or more characters may be hard to read or occluded.

Another contribution is the SVT-Perspective dataset, which we propose to evaluate perspective text recognition. On this dataset, our method compares favorably to the state-of-the-art, with a 36% improvement in recognition accuracy. Similarly, on a public dataset for texts of arbitrary orientations, our method achieves a 31% improvement in recognition accuracy over the compared methods. Therefore, our results and dataset serve as a baseline for future studies on perspective texts of arbitrary orientations.

Acknowledgments

We thank Quang Loc Le, Bolan Su and Hung Huu Hoang for their helpful comments on the manuscript. We would also like to thank Qianpan Jiang, Thanh Phu Tran and Quan Liu for their help in annotating the SVT-Perspective dataset. This research is supported in part by A*STAR grant 0921010051.

References

[1] K. Wang, B. Babenko, and S. Belongie. End-to-End Scene Text Recognition. In ICCV 2011.

[2] A. Mishra, K. Alahari, and C. V. Jawahar. Top-Down and Bottom-up Cues for Scene Text Recognition. In CVPR 2012.

[3] D. L. Smith, J. Field, and E. Learned-Miller. Enforcing Similarity Constraints with Integer Programming for Better Scene Text Recognition. In CVPR 2011.

[4] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky. Large-Lexicon Attribute-Consistent Text Recognition in Natural Images. In ECCV 2012.

[5] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 Robust Reading Competitions. In ICDAR 2003.

[6] L. Ballan, M. Bertini, A. Del Bimbo, and G. Serra. Semantic Annotation of Soccer Videos by Visual Instance Clustering and Spatial/temporal Reasoning in Ontologies. *Multimedia Tools and Applications*, 2010.

[7] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE TPAMI*, 2009.

[8] J. Liang, D. Doermann, and H. Li. Camera-based Analysis of Text and Documents: A Survey. *IJDAR*, 2005.

[9] B. Epshtein, E. Ofek, and Y. Wexler. Detecting Text in Natural Scenes with Stroke Width Transform. In CVPR 2010.

[10] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting Texts of Arbitrary Orientations in Natural Images. In CVPR 2012.

[11] L. Neumann and J. Matas. A Method for Text Localization and Recognition in Real-world Images. In ACCV 2010.

[12] L. Neumann and J. Matas. Real-Time Scene Text Localization and Recognition. In CVPR 2012.

[13] J. J. Weinman, E. Learned-Miller, and A. R. Hanson. Scene Text Recognition Using Similarity and a Lexicon with Sparse Belief Propagation. *IEEE TPAMI*, 2009.

[14] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng. Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning. In ICDAR 2011.

[15] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-End Text Recognition with Convolutional Neural Networks. In ICPR 2012.

[16] K. Wang and S. Belongie. Word Spotting in the Wild. In ECCV 2010.

[17] A. Mishra, K. Alahari, and C. V. Jawahar. Scene Text Recognition using Higher Order Language Priors. In BMVC 2012.

[18] C. R. Dance. Perspective Estimation for Document Images. In Document Recognition and Retrieval 2001.

[19] G. K. Myers, R. C. Bolles, Q.-T. Luong, J. A. Herson, and H. B. Aradhye. Rectification and Recognition of Text in 3-D Scenes. *IJDAR*, 2005.

[20] T. Gandhi, R. Kasturi, and S. Antani. Application of Planar Motion Segmentation for Scene Text Extraction. In ICPR 2000.

[21] L. Li and C. L. Tan. Recognizing Planar Symbols with Severe Perspective Deformation. *IEEE TPAMI*, 2010.

[22] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In BMVC 2002.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *IJCV*, 2005.

[24] Q. Zheng, K. Chen, Y. Zhou, C. Gu, and H. Guan. Text Localization and Recognition in Complex Scenes Using Local Features. In ACCV 2010.

[25] M. Iwamura, T. Kobayashi, and K. Kise. Recognition of Multiple Characters in a Scene Image Using Arrangement of Local Features. In ICDAR 2011.

[26] I. Z. Yalniz and R. Manmatha. An Efficient Framework for Searching Text in Noisy Document Images. In DAS 2012.

[27] M. Rusinol, D. Aldavert, R. Toledo, and J. Llados, Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method. In ICDAR 2011.

[28] A. Bosch, A. Zisserman, and X. Munoz. Scene Classification Via pLSA. In ECCV 2006.

[29] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. In ECCV 2004.

[30] LIBSVM. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[31] S. Maji, A. C. Berg, and J. Malik. Efficient Classification for Additive Kernel SVMs. *IEEE TPAMI*, 2013.

[32] T. E. de Campos, B. R. Babu, and M. Varma. Character Recognition in Natural Images. In VISAPP 2009.

[33] R. Nagy, A. Dicker, and K. Meyer-Wegener. NEOCR: A Configurable Dataset for Natural Image Text Recognition. In CBDAR 2011.

[34] ABBYY FineReader 9.0. http://www.abbyy.com/.