Perspective Scene Text Recognition With Feature Compression and Ranking

Yu Zhou[1‡], Shuang Liu[1‡], Yongzheng Zhang[1§], Yipeng Wang[1], and Weiyao Lin[2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China,
[2] School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Abstract. In this paper we propose a novel character representation for scene text recognition. In order to recognize each individual character, we first adopt a bag-of-words approach, in which the rotation-invariant circular Fourier-HOG features are densely extracted from an individual character and compressed into middle level features. Then we train a set of two-class linear Support Vector Machines in a one-vs-all schema to rank the compressed features by their contributions to the classification. Based on the ranking result we further select and keep those top rated features to build a compact and discriminative codebook. By using densely extracted features that are rotation-invariant and efficient, our method is capable of recognizing perspective texts of arbitrary orientations, and can be combined with the existing word recognition methods. Experimental results demonstrates that our method is highly efficient and achieves state-of-the-art performance on several benchmark datasets.

1 Introduction

Nowadays with the widespread availability of low cost devices equipped with cameras, lots of natural scene images that contain text information are generated. Texts in images contain much semantic information, which could be used to build useful applications such as street sign interpretation, content based image retrieval and product recognition.

Unlike traditional document optical character recognition (OCR)[1], which has achieved sufficient accuracy for practical application, recognizing text in uncontrolled environments is still a challenge. This is because texts in uncontrolled environments often suffer from low resolution, blur, non-uniform illumination and cluttered background. Especially, many images captured by handheld devices may suffer from perspective distortion, which is still an open problem in the computer vision community.

Many text detection methods, such as [2–5] already provides character level segmentations. Thus, we are mainly concerned with segmented scene text. Most

[‡] These authors contributed equally
[§] Corresponding author, Email: zhangyongzheng@iie.ac.cn

existing scene text recognition methods are focused on recognizing frontal scene text, while the task of recognizing perspective text of arbitrary orientations is still not well addressed. Some methods use text rectification to deal with perspective distortion with the assumption that the shape of the text can be extracted with high accuracy and the text is in a straight line [6, 7]. This approach however, might not work well because of the aforementioned interference factors. Besides, methods that rely on rigid features, e.g., histogram of oriented gradient(HOG) [8], achieved satisfying performance on frontal text recognition datasets. However it cannot be used for recognizing perspective text directly[9, 10]. To solve this problem, one can train a classifier with character examples of all possible poses. However, this approach is not realistic since it is expensive to collect and label examples. Also, to describe texts in such a way would make a model too complex and computationally inefficient. Therefore, it is important to develop new methods to effectively represent and recognize characters in natural scene images.

Usually, to discriminate among a fairly large number of classes (e.g., 62 character classes), low level features have to be extracted densely to provide enough information. These low level features can be directly used in a bag-of-words approach [11, 12].However non-text regions often introduce strong noise that are not helpful. These features have to be processed equally when searching the visual word codebook, and this can be very costly. Besides, since there are only a few distinctive prototypes for each character class, considering all of these low level features individually is inappropriate. To address the above problem, we observe that characters can be reduced to a few reoccurring shape prototypes. For example, intuitively, from a rotation-invariant point of view "o" is made entirely of curve strokes, and "N" is made of straight strokes and two identical turns. These patterns are obvious and prevalent regardless of languages. Also, it can be expected that the features of non-symmetrical characters such as "R" and "G" will have more variation than those of symmetrical ones. If these patterns can be discovered we can build a more compact and discriminative middle level representation with them. Although, a rotation-invariant representation would cause some confusions in individual character recognition (e.g., between "N" and "Z"), language models can be designed to effectively correct these confusions by incorporating lexicons.

Therefore, in this paper, we propose to extract rotation-invariant features densely and compressing them into compact and discriminative middle level features. These features are automatically learned and independent to language. We first extract circular Fourier-HOG (CHOG) [13] densely from character images, which serves as the underlying low level rotation invariant feature. Due to the nature of scene text, it is the rotation that causes the most signification variance when the viewpoint changes. Although CHOG is only invariant to in-plane rotation, when trained with a multi-scale bag of words scheme on sufficient samples from different viewpoints, it can tolerate perspective distortion to some extent. Next, to compress the CHOG features we use k-means to partition them into a number of prototypes depending on the variation. Then, we rank these prototypes and use the top rated features to build a compact visual word codebook. Finally we retrain a character classifier with the new codebook. In this way the size of our codebook and the computational workload of our method is significantly decreased. To evaluate the effectiveness of our method, we incorporate our method into the classic PLEX word recognition pipeline [14] and compare our method to the state-of-the-art methods on scene character recognition and scene word recognition. Our method outperforms the state-of-the-art methods on perspective text recognition while being orders of magnitude faster, and its performance is competitive on frontal text recognition.

The remaining of this paper is organized as follows: in Section 2 we discuss related work. We describe our method in detail in Section 3, followed by the implementation details and the experimental results of our method. Finally we conclude our paper in Section 6.

2 Related Work

A variety of methods have been developed for scene text detection and recognition. Maximally stable extremal region[6, 1, 15], stroke width transform[3] and HOG[14] have been successfully applied in scene text detection. The outputs of the detection algorithm are usually the bounding boxes of either characters or words. HOG templates have also been used to match character instances in test images with training examples[16]. Shi et al. [17] proposed to use manually designed deformable part-based model to represent characters. Most of these methods are mainly concerned with only frontal text.

Using rotation invariant feature such as scale invariant feature transform (SIFT) to describe characters has been proved successful. Phan [12] proposed to use dense SIFT instead of normal SIFT to describe individual characters. With the original SIFT, the descriptors are only extracted at sparse interest points. Since scene characters suffer from deformations such as blurring and uneven illumination, the number of detected interest points is not sufficient. The dense SIFT defined in the literature was designed for scene classification, which does not require rotation invariance [18]. An extraction scheme that fixes the position and size but allows the orientation of the interest points to vary was devised in their work, which provides rotation invariance. The work of Phan et al. provides helpful insights into perspective scene text recognition, and it also introduces two datasets that are used to benchmark perspective recognition performance.

SIFT aligns a local coordinate system to the dominant gradient direction at each detected interest point, which relies on the assumption that such a dominant gradient orientation is available. SIFT does not work well for arbitrary positions or dense feature computation, and it is a main source of error in dense image alignment[19]. Most recent text recognition approaches skip this step and use the non-invariant dense HOG features with a sliding window classifier.

CHOG, on the other hand, offers a well defined rotation behavior by representing circular HOG on Fourier domain. The window function of CHOG is isotropic, unlike rectangle spatial window, the descriptor rotates with respect to rotation of its underlying data without leading to any discrete binning artifacts in the histogram. Also, there is no need for interpolation. CHOG can be computed both densely and efficiently, while still being highly discriminative like HOG with rectangle spatial window. Therefore, we use CHOG as the low level building block of our method.

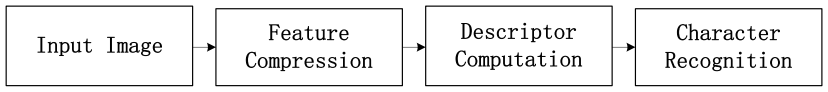

Fig. 1. Flow diagram of our method

3 Proposed Method

An overview of our method is shown in Figure 2. The flow diagram of our method is illustrated in Figure 1. In the following section we describe the procedure for each step.

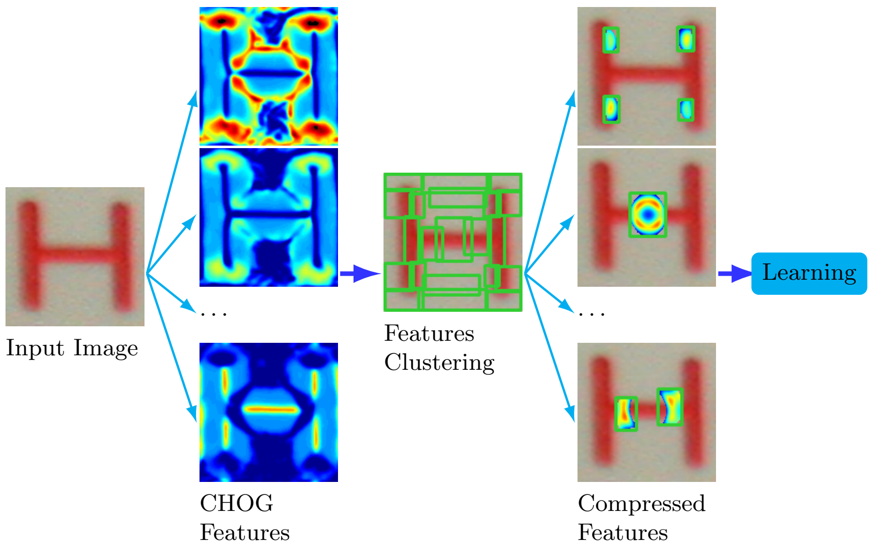

Fig. 2. Overview of our method. For a single character, the densely extracted rotation-invariant features are compressed and ranked by their discriminativeness, which are used to retrain the final character classifier.

3.1 Character Representation

In order to sufficiently describe character images, local descriptor of the images has to be computed densely. First, for both training and testing we resize all character images to 48×48. Next, to capture the multi-scale structure in a pixel's surrounding, Gaussian window functions of different sizes are used. The radial profile of the circles is nested and Gaussian smoothed ($e^{\frac{-(r-d)^2}{2\sigma^2}}, \sigma \in \mathbb{R}$), to ensure that the corresponding CHOG descriptors are neither suffering from discretization effects nor from small deformations. A local CHOG at position $x \in \mathbb{R}^2$ is computed by collecting all magnitudes of gradients within the window function w contributing to orientation $n \in \mathbb{R}^2, \|n\| = 1$ according to the continuous distribution function

$$\mathrm{CHOG}\{f\}_w(\mathbf{x}, \mathbf{n}) = \int_{\mathbf{r} \in \mathbb{R}^2} \|\mathbf{g}(\mathbf{r})\| \delta_{\mathbf{n}}(\hat{\mathbf{g}}(\mathbf{r})) w(\mathbf{x} - \mathbf{r}) d\mathbf{r} \tag{1}$$

where $\mathbf{g} : \mathbb{R}^2 \to \mathbb{R}^2, g = \nabla f$ is the gradient field of the image f, $S1$ denotes the unit-circle. $\hat{\mathbf{g}} = \mathbf{g}/\|\mathbf{g}\|, \hat{\mathbf{g}} := \mathbb{R}^2 \to S_1$ the gradient orientation field and $\mathbf{n} \in S_1$ is the current direction histogram entry taken into account. $\delta_{\mathbf{n}} : S_1 \to \mathbb{R}$ denotes the Dirac delta function on the circle that selects those gradients out of $\mathbf{g}$ with orientation $\mathbf{n}$. Next, the CHOG features are represented in terms of the orthogonal (periodic) circular Fourier basis functions. Thanks to the rotation preserving characteristic, CHOG features rotates smoothly with respect to the underlying image data. We refer the interested readers to [13].

In our experiments specifically, we use more nested circles for larger scale structure, and less nested circles for smaller scale structure. Using more nested circles will result in longer and more detailed descriptors and vice versa. The rationale behind this is that structures of different scales are supposed to carry different amount of information, thus it would make sense to use descriptors of different detail to describe structures of different scale. From our understanding of the description in [12], the standard SIFT implementation was used. The length of the SIFT feature vector is 128 for all key points of different sizes. Using long feature vector to describe small scale image will inevitable introduce more noise and unnecessarily slow down the recognition. Therefore, the features we extracted are supposed to have the advantage of reduced complexity and higher quality [20].

3.2 Feature Compression

The final descriptor of a single character is a histogram computed by assigning its visual words to their nearest visual codes. The variation of the low level features of any particular character is different but limited. Most of the densely extracted low level features vary from their neighbors slightly, except for those around sharp gradient adjustment. Non-text regions and smooth regions often do not contribute to the classification. Yet to find the most similar visual word in the codebook the distance has to be computed for all these features nonetheless, which is a huge waste of computation time. The image resolution of the character segmented from the scene image is generally very small. Therefore, as suggested in [12], keypoint detection methods fail to produce enough high quality keypoints for meaningful classification. Moreover, although the cost of clustering is non-trivial, we can compress the densely extracted features to a compact subset, in which all the features are distinctive from one another. We can both benefit from the discriminative power of densely extracted features and compute the descriptor (histogram) faster. As a result, the combined computational workload is still significantly smaller than directly using densely extracted feature.

Of course, if somehow we could use only the discriminative and distinctive features to build the histogram, we could avoid the unnecessary computation. We propose to densely extract CHOG features on every pixel, then use k-means to compress these features while explaining a sufficient percentage of variation necessary for effective recognition. Here we denote the “percentage explained” as the ratio of the sum of the standard deviation of each clusters to the standard deviation of the whole image. Although the sum of the variation of the features is readily available, we cannot directly determine how many clusters to use. Through empirical analysis on the training samples, we learn the boundary of the number of clusters that is needed to explain a required percentage. We also learn the relationship between the number of clusters and the percentage of variation explained with linear regression. At runtime, we start with the number of clusters predicted by the linear regression model and run k-means for a few iterations, then increase the number of clusters and restart until the required percentage of variation is explained or the number of clusters reaches the upper boundary. The resulting centroids of the clusters are compressed, stable and more importantly distinctive from one another. The computation overhead of k-means clustering is easily compensated because the computation workload using the compressed features instead of the original densely computed raw features is significantly reduced. The outline is outlined in Procedure 1. Finally, to exploit the spatial characteristic for each compressed feature vector, we add another dimension by appending the distance between the centroid corresponding to the compressed feature and the center of the character to the end. This is before the global vocabulary construction. In a rotation invariant setting it is difficult to differentiate between characters like “U”,“C”. Doing so has virtually no cost, we can both keep the rotation invariance and build a more discriminative vocabulary, since in general “U” are more slender than “C”.

3.3 Feature Selection

To build a compact and discriminative codebook, we need to rank and filter out those visual words that contribute weakly to the classification. First we perform k-means clustering on the compressed features of all the training samples to generate an initial codebook. We then use this codebook to compute the initial histograms of each training samples. These histograms are used to train a set of two-class linear support vector machines (SVMs) in a one-vs-all schema, whose weights are used to rank the visual words by their relevance. We keep

Procedure 1 Feature Compression

```
Input: Raw features, required percentage, variation predictor
Output: Compressed features, percentage explained
1: Variation Computation: Compute the standard deviation of the raw features.
2: Cluster Initialization: Predict the number of clusters needed to explain the
   required percentage of variation.
3: Feature Clustering: Start clustering and compute percentage of variation ex-
   plained when it converges.
4: Reiteration: Increase the number of clusters and restart until required percentage
   is explained or the number of clusters exceeds the upper boundary, then repeat step
   3.
```

the top rated ones of each character class and build a new codebook. We then compute new histograms while discarding features that are not found in the new codebook. Finally we train a multi-class radial basis function (RBF) kernel SVM with the new histograms as our character classifier. Multi-class SVM has a well defined probabilistic output, which can be useful for further natural language processing. Ranking the feature by their relevance helps us to focus on discriminative features and discard unnecessary features.

The one-vs-all schema is used to rank the features. SVMs classify data via finding a separating hyperplane with the maximal margin between two classes. Given a set of the visual word histograms $\mathbf{x}_i \in \mathbb{R}, i = 1, \ldots, l$ and character classes $y_i \in \{1, \ldots, 62\}, i = 1, \ldots, l$, we use a one-vs-all schema for each individual character class to train a set linear SVMs that solve the following unconstrained optimization problem:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{1}^{l}\xi(\mathbf{w}, b; \mathbf{x}_i, y_i) \tag{2}$$

where $\xi(\mathbf{w}, b; \mathbf{x}_i, yi)$ is a loss function, and $C \geq 0$ is a penalty parameter on the training error. In this paper specifically we use L2-loss linear SVMs. For testing instance $\mathbf{x}$, the decision function is

$$f(\mathbf{x}) = sgn(\mathbf{w}^T\phi(\mathbf{x}) + b) \tag{3}$$

where the mapping function is $\phi(x) = x$. After the set of linear SVMs are obtained for each character class, the weights $\mathbf{w} \in \mathbb{R}$ in eq.2 can be used to decide the relevance of each feature [21]. Because the absolute value of the weights $\mathbf{w}$ in linear SVMs indicates the importance of a particular feature in the decision function Equation3. The features can be ranked by sorting the corresponding weights [22]. We only keep the top **K** ranked features. This procedure is outlined in Procedure 2. Some of the top ranked features sorted by character class is shown in Figure 3, note that the nested window functions are not displayed because they are hard to visualize, thus the bounding box contains merely the center positions of the CHOG features, which capture the underlying surrounding structures.

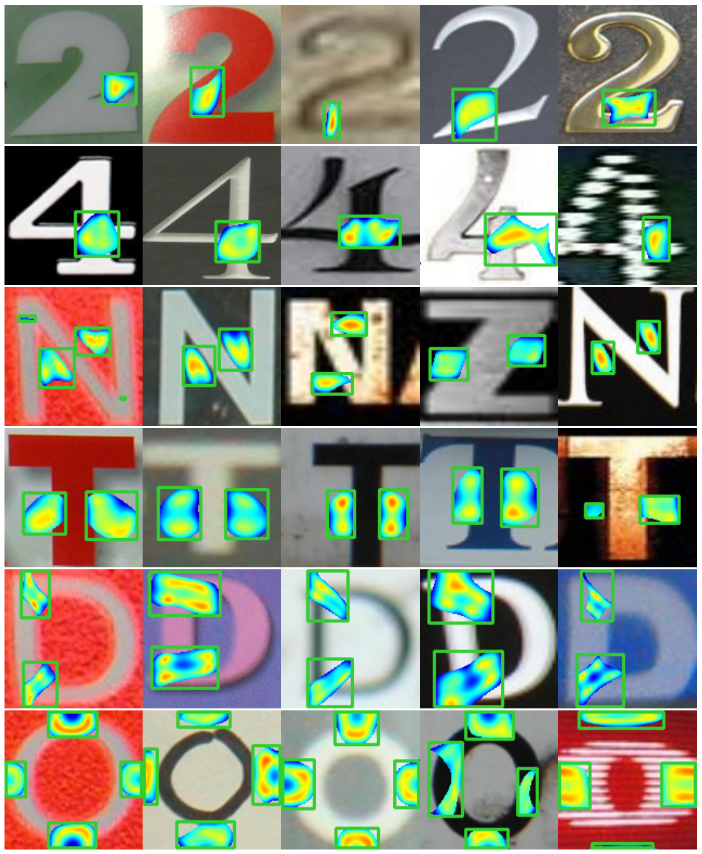

Fig. 3. Top rated compressed feature examples from several character classes

4 Implementation Detail

In our implementation we densely extract CHOG features of three different scales on every pixels. By nesting window functions of different sizes we extracted feature vectors of different lengths, which describe the surrounding area with different radius and detail. The window functions are given by $w1 := \{d = 0, \sigma = 4\}$, $w1 := \{d = 0, \sigma = 4\}; w2 := \{d = 4, \sigma = 6\}$ and $w1 := \{d = 0, \sigma = 4\}; w2 := \{d = 4, \sigma = 6\}; w3 := \{d = 10, \sigma = 12\}$, where σ indicates the outer radius and d indicates the inner radius. To compare the computational cost, we used the average dimension of the feature vectors. The average length of the extracted CHOG feature vectors is 60. On average the extraction procedure for each image takes 200ms.

In terms of feature compression we obtain the best result in experiments when roughly 95% of the variation of individual character image is explained. A linear regression model is trained to describe the relationship between the percentage of variation explained and the number of clusters used in k-means. At runtime this

Procedure 2 Feature Ranking

```
Input: Training samples, visual word codebook
Output: Top ranked features and retrained classifier
1: Intial Train: Train a set of two-class L2-loss linear SVMs for each individual
   character class using one-vs-all schema, using grid search to find best penalty pa-
   rameters $C$ on the visual word codebook.
2: Feature Ranking: Rank the visual words according to the absolute values of
   weights in the set of SVMs for each individual character class.
3: Feature Selection: Truncate the visual word codebook, keeping only the top K
   ones in each individual character class.
4: Model Retrain: Retrain a multi-class SVM to obtain the final classifier.
```

model is used to determine the initial number of clusters we use on the testing images. After a few initial iterations we increase the number of clusters and restart until the number of clusters reaches the upper boundary or the required percentage of variation is explained. In our implementation on average, this procedure needs 10 iterations with 20 clusters on 48×48 feature vectors whose average length is 60. Supposing on a visual word codebook of size $\mathbf{N}$ the workload for finding the most similar $\mathbf{K}$ prototype with $\mathbf{M}$ dimension is $\mathbf{N} \times \mathbf{M} \times \mathbf{K}$, this procedure is equivalent to a workload of $10 \times 20 \times 60 \times (48 \times 48) = 6912000$, which takes 100ms on average.

For feature ranking, the relationship between the number of features and the accuracy of the classification is illustrated in Figure 4. Take the visual word codebook used for SVT-Perspective for example. As illustrated in Figure 4, the best performance was achieved when the size of the vocabulary learned from compressed CHOG features is 250. The computation workload defined above equals to $1 \times 250 \times 60 \times 20 = 300000$. The combined computation workload of our method is only 13.0% of the state-of-the-art perspective character recognition method described in [12], which is equivalent to $1 \times 3000 \times 128 \times (12 \times 12) = 55296000$. Consequently, on SVT-Perspective (with the original lexicons), the average processing time of our method is 3.5 seconds while the method in [12] is 38.6 seconds.

5 Experimental Results

Proposed method has been evaluated on several benchmark datasets. Results on these datasets are compared to the state-of-the-art methods. For character recognition our method was evaluated and compared to other methods on ICDAR-Char and SVT-Char. ICDAR-Word and SVT-Word are used for frontal word recognition, while SVT-Perspective-Word and MSRA-TD-500 are used for the recognition of perspective word.

The PLEX framework[14] requires a lexicon for each word image, which is not provided in ICDAR-Word by default. For fairness we used the same lexicons and lexicon construction method in [14]. Following previous work [17, 23, 10, 12] on this dataset we skipped the words with less than 3 characters, and those with

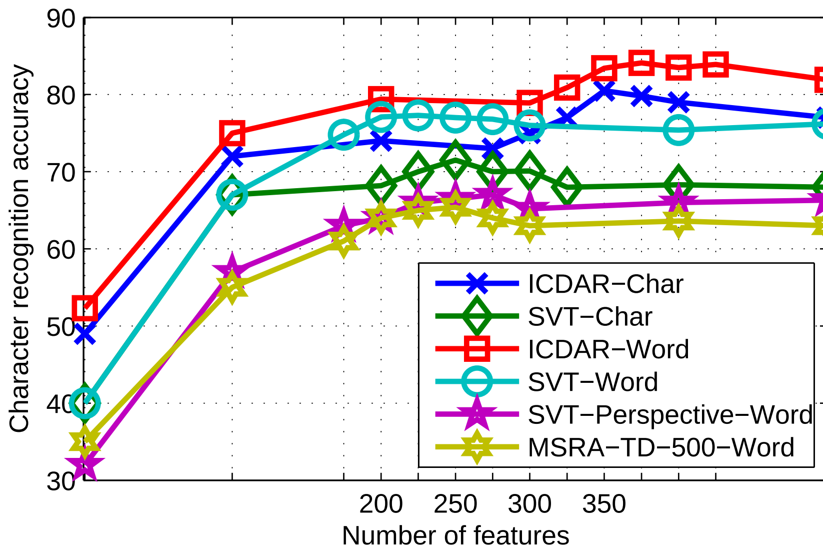

Fig. 4. Recognition accuracy of our method with different vocabulary size

non-alphanumeric characters, and then constructed additional lexicons denoted as *Full* which contain all the combined ground truth words.

5.1 Classifier Training

Our classifier was trained only on frontal examples. To train the character classifier we harvested labeled character images from ICDAR-Char, SVT-Char, IIIT 5K-Word and Chars74K. ICDAR-Char is a character level subset of the ICDAR 2003 Robust Word Recognition Competition [24] dataset. Similar to ICDAR, the SVT dataset [14, 25] also contains both word level annotations on full images and character level annotations. IIIT 5K-Word [23] contains word level and character level annotation of cropped word images only. Chars74K [26] contains only cropped characters. Our character detector in the PLEX framework was also trained on these datasets. Leave-one-out cross validation was performed on these four datasets.

Theoretically, the performance of our method on perspective scene text datasets would not degrade as much as those methods that only focused on frontal text recognition, because CHOG is inherently more suited for dense computation compared with SIFT and our representation is more robust. The experiments in Section 5.4 demonstrated the effectiveness of our method.

5.2 Character Recognition

We evaluated the character recognition performance of our method on both ICDAR-Char and SVT-Char. Table 1 lists the performance on these two datasets.

Some characters in SVT suffer from perspective distortion, which makes SVT different from ICDAR in that the characters in ICDAR are mainly frontal. This difference is reflected by the recognition accuracy difference between two datasets. Our method achieved the best result on SVT-Char. Without context, it is hard to tell the difference between some characters such as "W" and "M", "q" and "b", etc. Nevertheless, the performance of our method on ICDAR-Char is better than that of [12] and is only slightly worse than that of [27]. Although the performance gap can be partially explained by the aforementioned reason, it suggests that neural network based methods are still better suited for frontal character recognition tasks accuracy-wise. These were the two main causes for the performance gap between our methods and [27] on frontal word recognition.

Table 1. Character recognition accuracy (in%). Only our method and [12] aims to solve perspective character recognition, others focus only on frontal character recognition.

Method	ICDAR-Char	SVT-Char
Proposed	80.5	**71.5**
FineReader 9.0 [28]	21.0	11.7
K. Wang (PLEX) [14]	64.0	N.A
Mishra [9]	N.A	61.9
Coates [29]	81.7	N.A
T. Wang [27]	**83.9**	N.A
Yi [30]	76.0	N.A
Phan [12]	75.6	67.0

5.3 Frontal Word Recognition

Although we focused on perspective scene text recognition, for a comprehensive comparison we still evaluated the word recognition performance of our method on frontal text datasets. The word level ICDAR-Word subset of ICDAR was used to evaluate a variety of scene text recognition methods in cropped images. Most of the word images in ICDAR-Word are frontal, whereas the SVT-Word dataset contains more perspectively distorted words. The word level SVT-Word subset of SVT was also used in the evaluation. Table 2 lists the word recognition results. Some recognition examples are shown in Figure 5. We achieved the state-of-the-art performance on SVT-Word and outperformed several other methods on ICDAR-Word. This result accords with our analysis in Section 5.2. ,

5.4 Perspective Word Recognition

We used the words with English characters and digits in SVT-Perspective-Word and MSRA-500 to evaluate our method on perspective word recognition. The recognition accuracy is listed in Table 3. The performance degradation between

Table 2. Frontal word recognition accuracy (in%).

Method	ICDAR-Word	SVT-Word
Proposed	84.1	**77.3**
FineReader 9.0 [28]	56.0	36.0
K. Wang et al. [14]	76.0	57.0
Mishra et al. [9]	81.8	73.3
Mishra et al. [23]	80.3	73.6
T. Wang et al. [27]	**90.0**	70.0
Phan et al. [12]	82.2	73.3

frontal and perspective text is also measured in Table 4. The smaller gap between the performance on SVT-Word and SVT-Perspective-Word indicates that our method is more robust against rotation and perspective distortion.

In addition to leave-one-out cross validation, we also excluded SVT-Char from the training set when SVT-Perspective-Word is being tested. In this way we eliminated the possibility of rigged result because the samples in SVT are very similar to those in SVT-Perspective. The fact that our method suffered the least from the perspective distortion revealed that even it was only trained on frontal examples it generalizes well on perspective texts of arbitrary orientations. Some example recognition results are shown in Figure 6.

Table 3. Recognition accuracy on perspective text (in%).

Method	SVT-Perspective Word	SVT-Perspective Word (Full)	MSRA-TD500-Word (Full)
Proposed	**67.0**	**45.7**	**65.4**
FineReader 9.0 [28]	16.9	9.7	23.2
K. Wang et al. [14]	40.5	26.1	44.5
Mishra et al. [9]	45.7	25.7	27.8
T. Wang et al. [27]	40.2	32.4	20.8
Phan et al. [12]	62.3	42.2	58.4

6 Conclusion

In this paper, we adopt a bag-of-words approach to recognize scene character. We propose to use CHOG as the building block, which has desirable properties such as rotation invariance and efficient dense computation. We compress the densely extracted features to summarize the over-complete raw features to essential subsets, which greatly reduced the computation workload. Finally, we rank the compressed features to build a compact and discriminative visual word codebook. Our method achieves the state-of-the-art performance on perspective scene text recognition, and its performance on frontal scene text recognition is competitive with many state-of-the-art methods. Moreover, for perspective scene

Table 4. Degradation in performance between frontal texts and perspective texts taken from [12] (in%).

Method	SVT-Word	SVT-Perspective Word	% Change
Proposed	**77.3**	**67.0**	**-13.3%**
FineReader 9.0 [28]	35.0	16.9	-51.7
K. Wang et al. [14]	57.0	40.5	-28.9
Mishra et al. [9]	73.3	45.7	-37.7
T. Wang et al. [27]	70.0	40.2	-42.6
Phan et al. [12]	73.7	62.3	-15.5

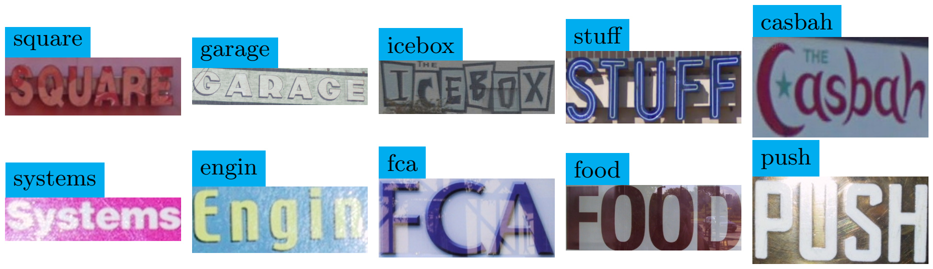

Fig. 5. Example recognition results on frontal text

text recognition our method is many times faster than [12], which previously was state-of-the-art method.

7 Acknowledgment

This paper is partially supported by National Natural Science Foundation of China under Contract nos. 61303170, 61402472 and 61471235, and also supported by the National High Technology Research and Development Program of China (863 programs)under Contract nos. 2013AA014703 and 2012AA012803.

References

1. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: CVPR. (2012)
2. Neumann, L., Jiri, M.: Real-time scene text localization and recognition. In: CVPR. (2012)
3. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: CVPR. (2010)
4. Chen, H., Tsai, S.S., Schroth, G., Chen, D.M., Grzeszczuk, R., Girod, B.: Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: ICIP. (2011)

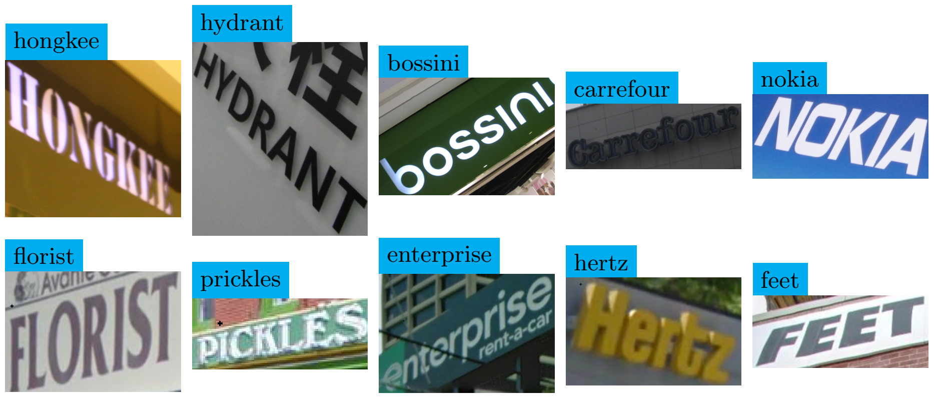

Fig. 6. Example recognition results on perspective text

5. Huang, W., Lin, Z., Yang, J., Wang, J.: Text localization in natural images using stroke feature transform and text covariance descriptors. In: ICCV. (2013)
6. Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: ACCV. (2010)
7. Dance, C.R.: Perspective estimation for document images. In: Electronic Imaging. (2002)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
9. Mishra, A., Alahari, K., Jawahar, C.: Top-down and bottom-up cues for scene text recognition. In: CVPR. (2012)
10. Novikova, T., Barinova, O., Kohli, P., Lempitsky, V.: Large-lexicon attribute-consistent text recognition in natural images. In: ECCV. (2012)
11. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV. (2004)
12. Phan, T.Q., Shivakumara, P., Tian, S., Tan, C.L.: Recognizing text with perspective distortion in natural scenes. In: ICCV. (2013)
13. Skibbe, H., Reisert, M.: Circular fourier-hog features for rotation invariant object detection in biomedical images. In: ISBI. (2012)
14. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: ICCV. (2011)
15. Chen, H., Tsai, S.S., Schroth, G., Chen, D.M., Grzeszczuk, R., Girod, B.: Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: ICIP. (2011)
16. Weinman, J.J., Learned-Miller, E., Hanson, A.R.: Scene text recognition using similarity and a lexicon with sparse belief propagation. IEEE TPAMI (2009)
17. Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S., Zhang, Z.: Scene text recognition using part-based tree-structured character detection. In: CVPR. (2013)
18. Bosch, A., Zisserman, A., Munoz, X.: Scene classification via plsa. In: ECCV. (2006)
19. Lin, W.Y., Liu, L., Matsushita, Y., Low, K.L., Liu, S.: Aligning images in the wild. In: CVPR. (2012)
20. Napoleon, D., Pavalakodi, S.: A new method for dimensionality reduction using k-means clustering algorithm for high dimensional data set. International Journal of Computer Applications (2011)

21. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning (2002)
22. Chang, Y.W., Lin, C.J.: Feature ranking using linear svm. JMLR (2008)
23. Mishra, A., Alahari, K., Jawahar, C.: Scene text recognition using higher order language priors. In: BMVC. (2012)
24. Sosa, L.P., Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: ICDAR 2003 robust reading competitions. In: ICDAR. (2003)
25. Wang, K., Belongie, S.: Word spotting in the wild. ECCV (2010)
26. de Campos, T., Babu, B.R., Varma, M.: Character recognition in natural images. In: VISAPP. (2009)
27. Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: ICPR. (2012)
28. ABBYY FineReader Professional 9.0: http://www.abbyy.com/. (2008)
29. Coates, A., Carpenter, B., Satheesh, S., Suresh, B., Wang, T., Wu, D.J., Ng, A.Y.: Text detection and character recognition in scene images with unsupervised feature learning. In: ICDAR. (2011)
30. Yi, C., Yang, X., Tian, Y.: Feature representations for scene text character recognition: A comparative study. In: ICDAR. (2013)