Scale Based Region Growing For Scene Text Detection

Junhua Mao [†], Houqiang Li [†], Wengang Zhou [‡], Shuicheng Yan [♯], Qi Tian [‡]
[†] Dept. of EEIS, University of Science and Technology of China, Hefei 230027, P. R. China
[‡] Dept. of Computer Science, University of Texas at San Antonio, Texas, TX 78249
[♯] Dept. of ECE, National University of Singapore, Singapore 117583
{mjhustc@mail., lihq@}ustc.edu.cn; zhwgeeis@gmail.com; eleyans@nus.edu.sg; qitian@cs.utsa.edu

ABSTRACT

Scene text is widely observed in our daily life and has many important multimedia applications. Unlike document text, scene text usually exhibits large variations in font and language, and suffers from low resolution, occlusions and complex background. In this paper, we present a novel scale-based region growing algorithm for scene text detection. We first distinguish SIFT features in text regions from those in background by exploring the inter- and intra-statistics of SIFT features. Then scene text regions in images are identified by scale-based region growing, which explores the geometric context of SIFT keypoints in local regions. Our algorithm is very effective to detect multilingual text in various fonts, sizes, and with complex background. In addition, it offers insights on efficiently deploying local features in numerous applications, such as visual search. We evaluate our algorithm on three datasets and achieve the state-of-the-art performance.

Categories and Subject Descriptors

I.4.8 [**Computing Methodologies**]: Image Processing and Computer Vision—*scene analysis*; I.4.9 [**Computing Methodologies**]: Image Processing and Computer Vision—*Applications*

General Terms

Algorithms, Experimentation, Performance.

Keywords

Scene Text, SIFT, Region Growing, Neural Network.

1. INTRODUCTION

With the increasing popularity of high-performance and low price digital camera devices and smart phones, explosive growth of photos/images are taken in daily life. In those images, it is widely observed that scene text frequently appears in the form of ads, graffiti, and signs. Lots of efforts have been made to understand the content of these images in multimedia community. As one of the critical yet challenging tasks, scene text detection serves as the basis for numerous multimedia applications, such as license plate detection [35], landmark recognition [26], and automatic navigation [28]. In augmented reality, signs and texts of foreign languages can be translated for visitors by their smart phones [12]. All these applications require a scene text detection algorithm with robustness and effectiveness.

MM'13, October 21–25, 2013, Barcelona, Spain.

ACM 978-1-4503-2404-5/13/10 ...$15.00.
http://dx.doi.org/10.1145/2502081.2502108.

Figure 1: Examples of the detected text regions by our algorithm (highlighted by the red bounding box) in natural scene images from three databases. (a) The sample images from the ICDAR database [1]. (b) The sample images from the SWT database [2]. (c) The sample images from the MSRG database.

Moreover, image search based on visual features is becoming popular and promising [27, 36]. However, the existing techniques often ignore the information of embedded text in images. If scene text can be robustly detected and recognized, the accuracy of search is expected to be further boosted.

The ignorance of scene texts in current search engine is mainly due to the dramatic decrease of accuracy of scene text detection algorithm compared to that of document texts. There are two main reasons. Firstly, the image condition varies extensively in natural scene image. Many tough conditions, such as blur, occlusion, intensive light, and low resolution, make it very difficult for scene text detection. Secondly, while document and caption text is structured and characterized by homogenous background, scene texts have little prior information of their structure and background. They often appear in different colors, sizes, fonts, spatial configurations and suffer from deformation when embedded on non-planar objects. In addition, current image retrieval systems adopt some popular features, such as SIFT and SURF [25, 34]. But many state-of-the-art text detection algorithms use features particularly designed for text [5, 9, 29], which needs additional computational cost in order to combine them with the image retrieval systems.

In this work, we propose a novel method to robustly locate the text within natural scene images based on the SIFT descriptors [16]. We define SIFT keypoints in the interior region of text patches as text keypoints. These keypoints often have high entropy of their orientation distribution and relatively low variance of their scales.

Based on these properties, we present an efficient algorithm to initially filter obvious non-text keypoints. In addition, extensive experimental study reveals that SIFT features on text keypoints possess very distinctive patterns. With these clues, a neural network classifier with accuracy higher than 80% is trained to select possible text keypoints. We also find an interesting phenomenon that SIFT keypoints from text regions often cluster together geometrically. To further explore the relationship of the geometrically neighboring text keypoints, we propose a simple but effective region growing algorithm. The algorithm can group homogenous keypoints together in a local region with different scales. The few misclassifications of the single keypoint can be rectified by the voting of all the keypoints in a grown region. Finally, regions recognized as text are fused and bounding boxes are drawn to highlight text regions.

Our method differs from previous methods in that it fully explores both the property of a single SIFT keypoint and the relationship of neighboring keypoints in a local region. Some previous text detection and recognition algorithms use isolated SIFT [4] or multiple kinds of local features [8] to detect the structure of a specific character in a certain type of language. Because isolated SIFT features are sensitive to noise, these methods could be only used in the document text analysis or the detection of scene text with clean background. We address this problem by using the information of the geometrically neighboring keypoints. Although our algorithm is particularly effective on language with complex structure (e.g., hieroglyphics), it is not limited to any specific language. The properties we used, such as the high entropy of the orientation distribution, the strong contrast between text pixels and background, and the low variance of color along the stroke, are shared by various kinds of language and handwriting texts. Unlike some algorithms, we do not need the assistance of any language-specific process, such as OCR filtering step [6]. So, our algorithm is not constrained to a specific language and has promising potential for multilingual multimedia tasks.

In contrast to other approaches that use features which are only effective on text, such as low variance of stroke width [9] and special wavelet or Laplacian coefficients [24, 30], we adopt a commonly used feature (SIFT). Therefore, our text detection results can provide important clues for numerous SIFT applications. For example, the algorithm can be combined with visual search systems without further computational load to extract feature descriptors. The SIFT information on text can also assist OCR engines and verify their recognition results.

The experimental results on three scene text databases reveal that, compared to some state of the art algorithms, our method is less sensitive for a specific training set. We evaluate the performance of our algorithm by training and testing in different datasets. When trained on one English and Roman numerals dataset and tested on a multilingual dataset, the f measure drops only by 6% at most. Therefore, some of the properties of SIFT descriptor are shared by different languages.

The main contributions of this paper are as follows:

- We select candidate SIFT keypoints through the unique properties of single text keypoints, such as the high entropy of their orientation distribution within a local region, relatively low variance of their scales, and their distinctive 128-D descriptors. Those properties are shared by multilingual texts.
- We present a novel scale-based region growing algorithm to explore the relationship of geometrically neighboring candidate keypoints.

The remainder of this chapter is organized as follows. Section 2 reviews related work on text detection. Section 3 introduces the properties of SIFT feature and the motivation for our algorithm. Section 4 discusses the proposed scene text detection algorithm in details. In Section 5, experiments are conducted and results are compared with the state-of-the-art methods in English, Latin and multilingual databases. Section 6 concludes the work.

2. RELATED WORK

Many text detection methods have been proposed recently. Comprehensive surveys are also made in [13, 15, 19]. In general, existing methods can be categorized into three groups: thresholding-based, texture-based and region-based.

Thresholding-based methods define a global or local threshold to separate text from background. Many thresholding algorithms, such as adaptive binarization techniques [15] and histogram-based thresholding [23] fall in this category. These methods are computationally efficient, but cannot well address complex background or variations of the text color and intensity level. Some methods, such as OTSU algorithm [13], perform well for automatically thresholding in 1-D data and can be used to automatically determine some parameters.

Texture-based methods treat text as a unique object that has distinguishable features from background. Popular features used include special horizontal and vertical frequencies [33], wavelet or Laplacian coefficients [24, 30], and features with low entropy [6, 7]. These methods still suffer from many limitations, including high computational complexity, inapplicability to slanted text, and waste of the information provided by the transformation.

Region-based methods use geometric constraints and information to choose text candidates and form bounding boxes based on region patches in images. Epshtein et al. [9] proposed a transform (SWT) to find regions with small variance of stroke width and generate letter candidates. The candidate letters are grouped into bounding boxes with geometric constraints. Their method claims to achieve the best result on ICDAR database. Chen et al. [5] modified the method using edge-enhanced Maximally Stable External Regions (MSER) to detect interested regions and found stroke width by distance transform. Neumann et al. [22] also presented an end-to-end method for text localization and recognition using MSER. Yao et al. [29] extended the standard SWT methods by adding two sets of features and a two level classification step. Their work is more effective on multi-orientation texts than previous SWT algorithms. In general, this group of methods can detect text in different scales and directions. However, it may be difficult for them to detect text characters with larger variance of stroke width and complex structures, such as hieroglyph.

There are some existing algorithms using SIFT to detect characters. Block et al. [4] extracted local features on some specific portions of a certain character to create their database. The features from a testing image are compared and matched with the images in the database to recognize the character. The algorithm relies only on isolated SIFT keypoints and ignore the valuable information provided by the neighborhood. So, their method is sensitive to background noises. Zheng et al. [32] extended this framework to recognize Chinese, Japanese and Korea characters. Because their task is character recognition instead of text detection, their dataset images are machine-generated and clean without any background, and their geometric verification is strict. It leads to possible false negatives on the task of detecting text with complex background. Some works adopt multiple kinds of local features and bag-of-visual-words representation [8]. But they focused on the recognition of single characters and their method requires manually segmented individual characters in an image.

Our method falls in the region-based category. We use scale-based region growing scheme to group keypoints in a local region. These regions can be robustly classified by the voting of all the keypoints belong to that region. Similar to conventional region-based method, we take advantage of geometric constraints to link cluster candidates into bounding boxes. We also borrow some insights from methods in the thresholding-based and texture-based groups. Our method takes advantage of the discriminative properties of SIFT keypoints on various kinds of characters which distinguish themselves from those of the background. In addition, thresholding-based methods are modified to automatically determine some parameters in our algorithm instead of their initial purpose to separate foreground pixels. Experimental results show that our method is not limited to specific characters and languages. Finally, we address the drawback of some region-based method in that the complexity of the structure and large variance of stroke width are beneficial properties and are well explored in our method.

3. MOTIVATION

Before discussing our approach, we would like to introduce the background of the classic SIFT feature briefly. SIFT feature is designed to capture the appearance of a local image patch centered at an interest point. The interest point can be detected with invariance detectors, such as DoG [16], Hessian Affine [20], etc, and is characterized with scale and orientation parameters. In the standard SIFT feature, around an interest point, the local patch with radius proportional to the characteristic scale is evenly partitioned into sub-patches along the direction of the characteristic orientation. Then, each sub-patch is described with an 8-D orientation histogram. Consequently, a 128-D SIFT descriptor is obtained by concatenating the orientation histograms over all 16 sub-patches. The merit of SIFT feature lies in its invariance in rotation and scale changes and robustness in illumination changes [20]. A standard SIFT feature contains the following components: location, scale, orientation, and 128-D descriptor vector.

Recent works have shown that SIFT descriptors combined with bag-of-visual-words algorithm have very good discriminative property in some specific characters, such as the digits in the license plate [35] and manually segmented English and Kannada characters [8]. We develop their methods. Our experimental study of SIFT feature shows that the keypoints on text regions process special properties in three ways, which are shared with multilingual characters, especially for the ones with complex structure.

Firstly, statistics properties of geometrically neighboring keypoints in a local text region are unique. For example, keypoints in text regions have high entropy of their orientation distribution and relatively low variance of their scales. We take advantage of these properties to select text sensitive keypoints. Experiments in Section 5 show that this selection step increases the accuracy of the trained classifier.

Secondly, the 128-D SIFT descriptors in text regions usually appear in different patterns from those in non-text regions. As illustrated in Figure 2, a trained classifier can successfully classify single text keypoints and non-text keypoints with an acceptable low misclassification rate. It can effectively filter keypoints on random background clutters (such as grasses and leaves), which are big challenges of text detection algorithms [29].

Thirdly, as shown in Figure 2, if we define the local region of a text keypoint as a circle whose radius is the scale of the keypoint, we can often find other text keypoints located in the region. When iteratively grouping the keypoints, a large region can be achieved. Keypoints in these regions tend to belong to the same group (text or non-text). We propose a novel region growing algorithm to utilize this property. The algorithm is easy to implement and can integrate with the single SIFT keypoint classifier to improve the performance.

Table 1: The notations and descriptions of the variables

Notations	Descriptions
$loc(p)$	The location of a keypoint p.
$s(p)$	The scale of a keypoint p.
$ori(p)$	The orientation of a keypoint p.
$R_l(p)$	The local region of a keypoint p.
$N(p)$	The neighborhood of a keypoint p.
$R_c(k)$	The region candidate k.
$Range(k)$	The range of the region candidate k.
F	The free keypoint set.
D_f	The initial free keypoint set.
$l(p)$	The label of a keypoints p.
$label(k)$	The label of a region candidate k.
$R_t(k^{'})$	The text region candidate $k^{'}$.

4. THE TEXT DETECTION ALGORITHM

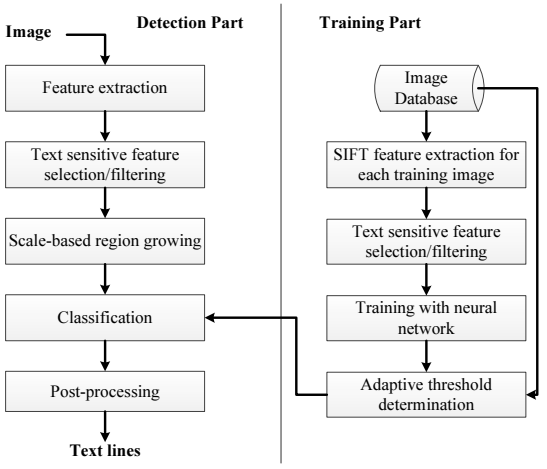

Figure 3: The flowchart of the text detection algorithm

The flowchart of our framework is shown on Figure 3. In Section 4.1, we discuss how to initially filter noisy SIFT features. In Section 4.2, a classifier is trained to label possible text keypoints. After that, in Section 4.3, a region-growing scheme is used to encapsulate those nearby sub-regions of interest and classify the regions according to their labels. Finally, in Section 4.4, a post-processing scheme is applied to determine bounding boxes for scene text in images. The notations and descriptions of the variables for our algorithm are summarized in Table 1.

4.1 Text Sensitive Feature Selection/Filtering

A large proportion of the keypoints actually serve as noise in the text detection process. By extensive experimental study of the statistic properties of keypoints in a local region, it is observed that there are two types of noisy SIFT features. The first type of noisy features is usually detected along straight edges with low entropy of their orientation distribution, while the second type exhibits very large or small scale. Those noises can be identified and removed by a fast filtering algorithm.

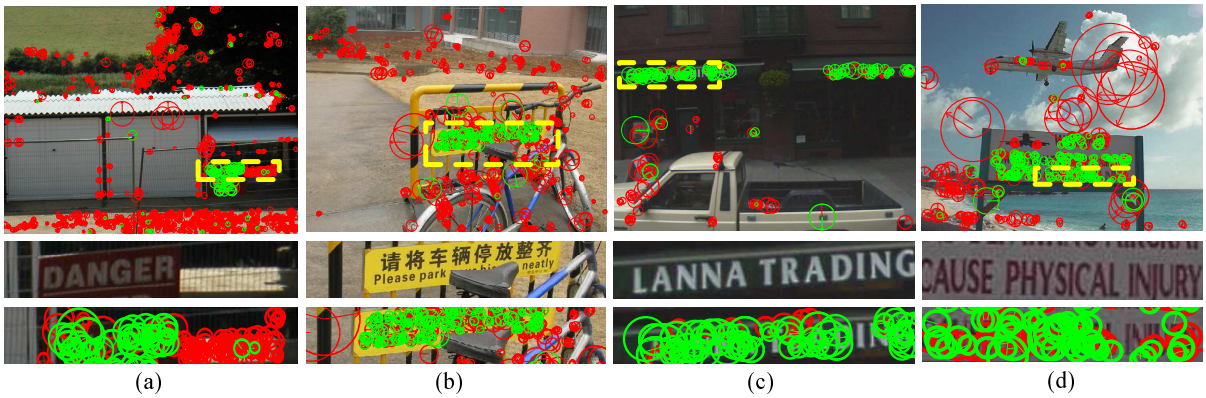

Figure 2: Illustration for the motivation of our algorithm. The images in the first row show all the SIFT keypoints after applying text sensitive feature section as described in Section 4 (Yellow rectangles represent some regions of interest (ROI)). The keypoint is represented as an arrow that starts with its location and points to its orientation with the length equal to its scale. We will adopt this representation format of keypoints in all the figures. A circle whose radius is equal to the scale of a keypoint is also drawn. The green and the red circles represent the keypoints that are classified as text keypoints and non-text keypoints respectively. The third row shows the ROI without the arrows and circles. The fourth row shows the ROI (zoom in). Note that text keypoints are often clustered together. If we carefully define a local region, the few misclassifications can be rectified by the voting of all the keypoints in that region. (Best viewed in color)

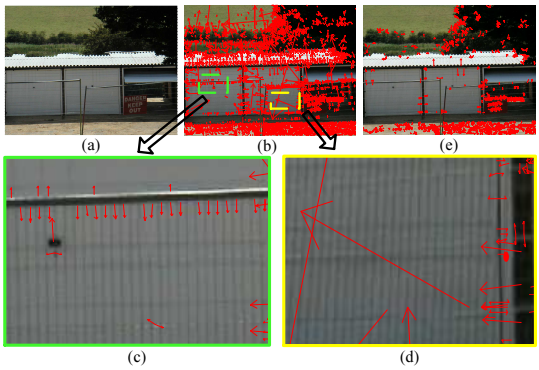

Figure 4: (a) The original sample image with size 1280×960. (b) The initial 5786 SIFT features. (c) The non-text paralleling keypoints along the long straight object (first type of noisy keypoints). (d) Useless keypoints with overly large scale (second type of noisy keypoints). (e) After applying the selection algorithm, 2202 keypoints remain. (Best viewed in color)

As shown in the green square of Figure 4(c), some non-text keypoints possess very distinctive property from text keypoints. They and their neighborhood have same or opposite directions to each other, thus possess low entropy of their orientation distribution. Such kind of SIFT keypoints is often observed along the long straight object. In contrast, high entropy often occurs for the orientation distribution of the neighborhood of a text keypoint. Experiment shows that, for text keypoints extracted from the three databases (described in Section 5.1), the entropy of the orientation distribution of their neighborhood is 0.9007 on average. For non-text keypoints, the entropy is 0.6489 on average, which is much lower than that of the text keypoints. Based on this observation, we filter the first kind of noisy keypoints using the following steps.

We denote the location of a keypoint p as $loc(p)$ and its scale as $s(p)$.We then define the local region $R_l(p)$ for a keypoint p as a circular region whose center is at $loc(p)$ and radius equals to $s(p)$.

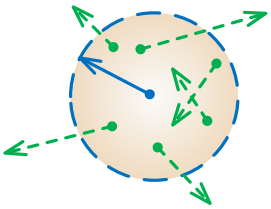

Figure 5: The sketch map for the local region of a keypoint. The solid line arrow represents a keypoint p. The dashed line arrows represent its neighborhood.

Figure 5 illustrates the local region of a keypoint. We also define the neighborhood keypoints of a keypoint p as all the keypoints locate in $loc(p)$, denoted as $N(p)$. For each keypoint p, we calculate the orientation entropy of all the keypoints in $N(p)$. If the entropy is larger than a threshold τ_1, the keypoint p will be pruned as the first kind of noisy keypoints.

It is intractable to calculate the entropy of the orientation distribution if we treat $ori(p)$ as a continuous random variable. Therefore, we quantize $ori(p)$ into m levels and treat $ori(p)$ as a discrete variable. In practice, a good performance can be achieved if m equals to 8 or larger. The threshold τ_1 could be learned easily in the training dataset by using OTSU algorithm [23]. This algorithm can automatically choose a threshold that maximizes the between-class variance. We will adopt the same algorithm to automatically determine the thresholds in this paper.

As shown in Figure 4(d), the keypoints in the yellow square are too large to be useful for the scene text detection. This kind of keypoints often represent the interaction between foreground and background objects. Since background changes in different images, these keypoints are not a trustworthy clue. So, we define the second type of noisy keypoints based on the proportion of the neighboring keypoints within a given scale interval. For each keypoint p, we calculate the number of keypoints in $N(p)$ whose scales are within the interval: $[r_{min} \cdot s(p) \;\; r_{max} \cdot s(p)]$. We denote the number as n_1. We also calculate the total number of keypoints in

$N(p)$, denoted as n_2. If $\frac{n_1}{n_2} < \tau_2$, the keypoint p will be pruned as the second type of noisy keypoints.

By testing and learning from training samples, the optimal numerical values of the variables are empirically selected as: $r_{min} = 0.67$, $r_{max} = 1.5$.

The experimental results show that about 70% of the SIFT keypoints on average can be pruned. Most of them are the noisy keypoints on non-text object. Some of the keypoints on the text regions are also eliminated. However, these keypoints are either the ones with overly large scale or the small scale ones among the long straight strokes of some characters. Both kinds of keypoints will disrupt the classifying process and are regarded as noise. Figure 4(e) shows the result of the noise filtering process.

4.2 Labeling of Single Feature

After coarsely filtering the two kinds of noisy keypoints, some non-text keypoints still remain. One method is to use the 128-D descriptor vector to discriminate such keypoints. We train a neural network to classify the single SIFT keypoint. The training process is described as follows.

The text keypoints and the non-text keypoints in the training database are treated as positive or negative training samples respectively. These keypoints are divided into 67% training samples and 33% testing samples.

A neural network with 128 dimensions input layers and 40 dimensions hidden layers are trained on the training dataset. The number of hidden layers is chosen to obtain the best training accuracy while avoiding huge computational cost. The trained neural network will label a testing keypoint as 1 (predicted as text keypoint) or -1 (predicted as non-text keypoint). We denote the label of a keypoints p as $l(p)$. Note that other machine learning algorithms, such as SVM, can also be applied.

4.3 Scale-based Region Growing

Due to the huge intra-class variations of characters, especially for text in multiple languages, it is difficult to distinguish each keypoint as text and non-text precisely because of potential noise in the background that is similar to text. Therefore, we propose a novel region growing algorithm to explore the context within the local region of a keypoint. After applying this algorithm, keypoints are grouped together to form several region candidates. The region candidates can be classified according to the labels of keypoints in the region.

4.3.1 Region Growing

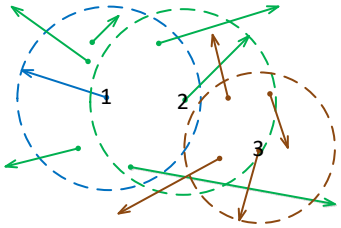

Figure 6: Region growing process. We start with keypoint 1 (the blue keypoint) as the seed point. Then, the keypoints in the local region of keypoint 1 (all the green keypoints) are included into the region. For each green keypoint (e.g., keypoint 2), all the keypoints within its local region are included (all the brown keypoints). We repeat such procedure for each newly included keypoint until no more new keypoints can be involved. (Best viewed in color)

Algorithm 1 Pseudo-code of the SIFT based region growing

Initialization:
  $k \leftarrow 0$; $F \leftarrow D_f$;
Iteration:
  while $F \neq \varnothing$ **do**
    $k \leftarrow k + 1$; $R_c(k) \leftarrow \varnothing$;
    $seed \leftarrow SelectSeed(F)$;
    $F \leftarrow F - seed$;
    AddKeypointToRegion($seed$,$R_c(k)$);
    while $\exists p \ : \ p \in F \ \wedge \ p \in Range(k)$ **do**
      AddKeypointToRegion(p,$R_c(k)$);
      $F \leftarrow F - p$;
    end while
  end while

It is easily observed that in a text region the keypoints are often densely clustered as shown in Figure 2. Inspired by the idea of region growing in image segmentation, an efficient scale-based region growing algorithm is developed to determine regions with such characteristics.

The proposed region growing process begins with randomly chosen keypoints as the seeds. We define the free keypoint set F as all the keypoints that have not been assigned to any region candidate. Initially, F is equal to the set of keypoints after the text sensitive filtering (denoted as D_f). The initial region candidates k (denoted as $R_c(k)$) only include the seed keypoints. We define the range of $R_c(k)$, denoted by $Range(k)$, as the sum of the local region of all the keypoints belong to that region candidate. The regions then gradually grow larger by including all the keypoints in F that fall in the range of the initial regions. This step is repeated until all the regions are stable. Finally, region candidates initialized by seeds of text keypoint will grow neatly to cover text regions while non-text seeds will grow to cover non-text regions. In other words, very few misconnections between non-text regions and text regions exist in this process as the experiment shows in Section 5.4. With the candidate regions, high accuracy can be achieved by rectify the incorrect classifications with correct ones in the same region. Some examples are shown in Figure 7.

The pseudo-code for the scale-based region growing process is shown in algorithm 1. The time complexity of this algorithm is $O(n)$ using the disjoint-set data structure, where n is the number of keypoints left after the text sensitive selection. The keypoint growing sequence as well as the initialization of the seeds has little influence on the final results.

4.3.2 Classification of the Region Candidates

After the above process, region candidates are obtained. Some of the region candidates are on the background, such as the leaves of plants. The next step is to classify them into text region candidates $R_t(k^{'})$ and non-text ones. We use the label of the region $label(k)$ calculated below to classify the region candidates.

$$label(k) = \frac{1}{N}\sum_{k=1}^{N} l(p), p \in R_c(k) \quad (1)$$

$$R_t = \bigcup_k \{R_c(k) | label(k) > \tau_3\} \quad (2)$$

N is the number of keypoints in $R_c(k)$. τ_3 is learned from the training dataset by OTSU algorithm. The region candidates whose labels are larger than τ_3 will be treated as text region candidates. As shown in our experiments (Section 5.4), the major part of most

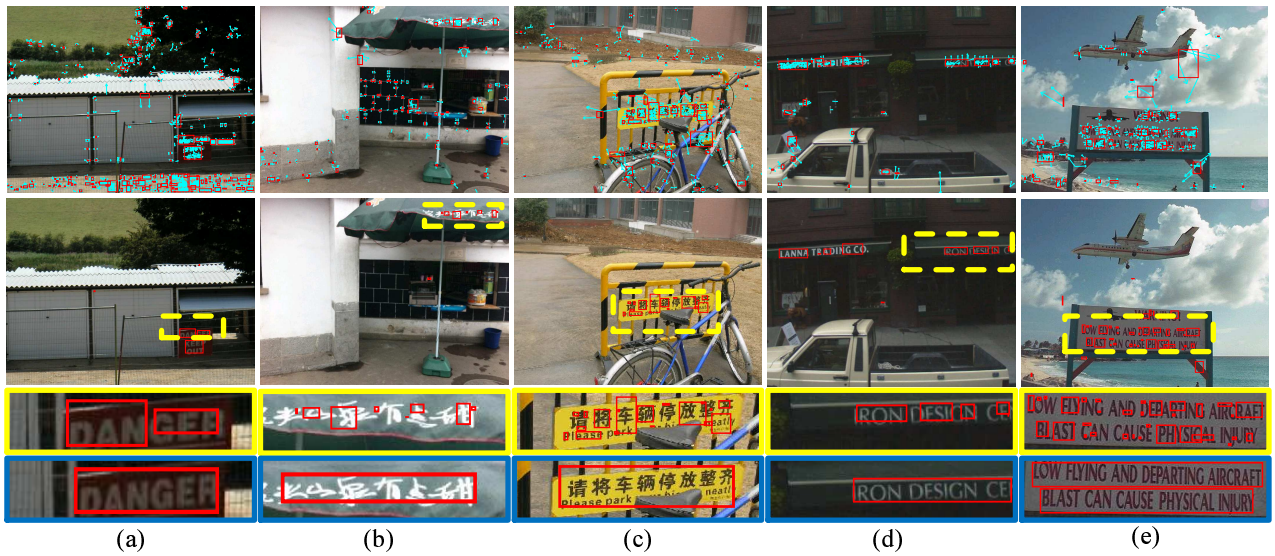

Figure 7: Some examples of scale-based region growing and classification. The first row shows the results of the region growing. Note that the major part of the regions are either on texts or other objects. The second row shows the results of the classification (yellow rectangles represent some regions of interest (ROI)). The third row shows the ROI (zoom in). The fourth row shows the final results after applying post-possessing.

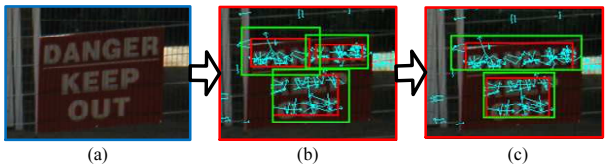

Figure 8: (a) Original image region. (b) The outer green rectangle indicates the impact rectangle of a text region candidate which will be used in the two-step text blocks formation. (c) The detected text blocks. (Best viewed in color)

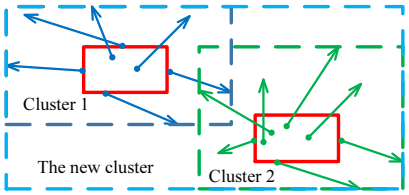

Figure 9: Merge region candidates with overlapped impact rectangles. Impact rectangle is the minimum rectangle that covers all the arrows of SIFT keypoints in the region candidate. (Best viewed in color)

grown regions are either within or outside the text regions. Therefore, the misclassification of a single keypoint by the neural network will be rectified by the voting of other keypoints in the same region candidate.

4.4 Post-processing

After obtaining text region candidates, the next process is to link them to form text blocks. The blocks which contain several text lines will be separated and adjusted to yield the final results.

4.4.1 Composition of Text Blocks

We form the bounding boxes embracing scene text by a two-step algorithm. Some scattered noisy cluster will also be eliminated by this step.

In step one, we define the impact rectangle of a region candidate as the minimum rectangle that covers all the arrows of SIFT keypoints as shown in Figure 8. The arrow of a SIFT keypoint starts with the location of the keypoints and points to its orientation with the length equal to its scale. Regions that have overlapped impact rectangles will be merged to form new text region candidates.

In step two, we group text region candidates to form the text block based on the following criteria:

- The grouped text region candidates appear in a linear form.
- The Possible Extending Distance (PDE) of a text region candidate is set as the width of the region.
- The distance for the centers of grouped neighboring region candidates must not exceed their minimum PDE.
- The grouping regions must have similar color histogram for their impact rectangles.

We use a scheme [14] which is originally developed for color quantization. This scheme extracts a 44-D color feature in HSV color space. It is adopted due to its simple procedure and good performance. We normalize the 44-D feature by their second order norm. The similarity of the color features for two cluster candidates are measured by their inner product.

In addition, we prune the scattered noisy keypoints by this step. A region candidate will be removed if it is too small (The area of the cluster is less than 100 pixels) and cannot be merged with other region candidates. Since small and isolated regions rarely appear in image as text, this step removes scatter noises effectively.

4.4.2 Division of the Text Blocks

As shown in Figure 8(c), for many natural scenes, several text lines are geometrically close to each other and the algorithm might group them as a single text block. Some background pixel lines are

Algorithm 2 Pseudo-code of text block division

Initialization:
T=GetTextblock();
Process:
$axis$=HoughTransform(T); S=Project(T, $axis$); v=Var(S);
if $v < \tau_v$ **then**
$T_1 = T$
else
τ_s=OTSU(S);
[B_1,B_2]=BandDivision(S,T,τ_s);
[B_f,B_b]=JudgeForeground(B_1,B_2);
T_1=Getnoise(B_f,B_b);
end if

also included in the text block. For example, the lower text block in Figure 8(c) contains two text lines and some background pixels. All these factors decrease the precision of the OCR algorithm or other text information applications. To solve this problem, we propose a fast text block division algorithm. The pseudo-code is shown in algorithm 2.

The vertical axis is determined by Hough transform for each text block. Grey level pixels are projected on the vertical axis to generate projection array S. To avoid the situation that the text block is actually a single text line, we use a threshold τ_v of the variance of S to determine whether a text block should be divided. As illustrated in Figure 10, the BandDivision() function separates the text block into two groups according to the adaptive threshold τ_s. Then we calculate the text SIFT keypoints in each group. The group with more text SIFT keypoints will be marked as the foreground group. In addition, the bands with very small height (fewer than 3 pixels) are also marked as foreground. We prune the background bands to form the initial bounding boxes.

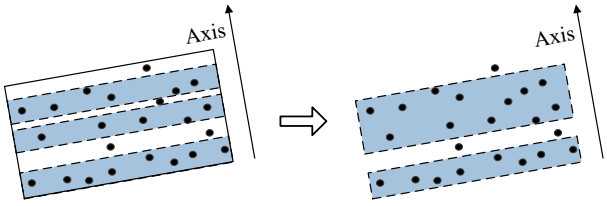

Figure 10: The sketch map of text block division. The points represent text SIFT keypoints in the text block. (a) Two groups that is represented by colored and blank bands respectively. (b) The result of text block division. The bands with very small height are marked as foreground.

4.4.3 Adjustment of the Initial Bounding Boxes

As shown in Figure 11, we need two steps to achieve the final result. Firstly, we adjust the size of the bounding boxes to the minimum rectangle that contains all the text keypoints in the initial bounding boxes. Secondly, we need to merge overly divided text blocks whose endpoints share the same horizontal or vertical coordinate. The final result is shown in Figure 11(c).

5. EXPERIMENTS

5.1 Experiment Databases

To compare with the state-of-the-art text detection algorithms, we evaluate our algorithm on two publicly available datasets and

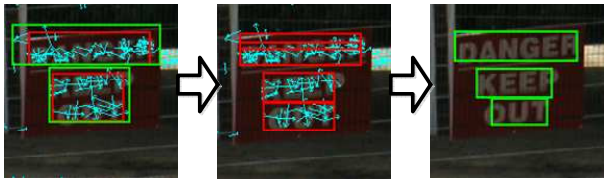

Figure 11: (a) Text blocks that contain more than one text line. (b) The result of the dividing algorithm. Note that unnecessary dividing occurs and the size of the text line square is not correct. (c) The result of the adjustment. (Best viewed in color)

our new benchmark dataset of mixture language, such as English, Latin, Roman numerals and hieroglyphs. The description of these three datasets is presented below. Some sample images are shown in Figure 12.

ICDAR Database (ICDAR) This dataset [1] has been used in ICDAR 2003 and ICDAR 2005, which are the most popular text detection competitions. It contains 258 images for training and 251 images for testing. The images cover various text conditions, such as blur, curving, strong highlights, transparence, different viewing angles, non-planar, and so on. Text in the database is relatively large compared to the size of the image. Many images are taken from book covers, road signs, and trademarks with simple background and a few disruptive objects. Word separation is needed in this dataset.

SWT Dataset (SWT) This database [2] contains 307 color images with sizes ranging from 1024×368 to 1024×768. The scene in this database is more complex and the texts are smaller and more intensive compared to the database of ICDAR. Various plants and undistinguishable repeating patterns, such as windows and fences, are common in images of the dataset. Therefore this database is much more difficult and challenging than ICDAR.

Multilingual Database (MSRG) The two databases above contain text only in Latin and English. However, many characters in diversified languages have different properties from Latin and English character. For example, hieroglyphs possess com-

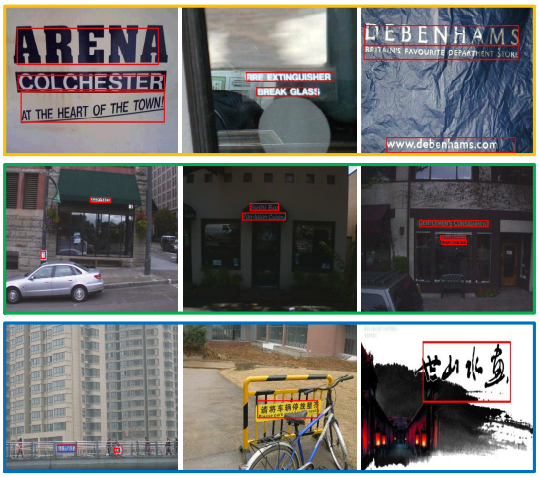

Figure 12: Examples of sample images and the detected text regions of our algorithm on the three databases. The images in the first, second, and third row are from the ICDAR, the SWT, and the MSRG database respectively.

plex structures and have dramatic variance of stroke width of some fonts. We construct the MSRG dataset, which contains a mixture of hieroglyphs, Latin, English and Roman numerals text in natural scene images. It includes 278 images of streetscapes, road signs, horizontal scrolls of painting or calligraphy, and so on, with various image conditions. The resolution of the images ranges from 922×692 to 1296×968. It is also more challenging than ICDAR database considering its complex background.

5.2 Evaluation Metric

To compare our work with the state-of-the-art algorithms, we choose the same evaluation metric of a benchmark reading competition, ICDAR 2003 [18]. In this system, the match m_p between two rectangles is defined as the area of intersection divided by the area of the minimum bounding box containing both rectangles. Therefore, m_p has the value one for identical rectangles and zero for rectangles that have no intersection. For each rectangle in the set of estimates (E), the closest match is found in the ground-truth set of targets (T), and vice versa. Hence, the best match $m(r, R)$ for a rectangle r in a set of Rectangle R is defined as:

$$m(r, R) = max\{m_p(r; r_0)|r_0 \in R\} \tag{3}$$

Then precision, recall and f measure are defined as:

$$precision = \frac{\sum_{r_e \in E} m(r_e, T)}{|E|} \tag{4}$$

$$recall = \frac{\sum_{r_t \in T} m(r_t, E)}{|T|} \tag{5}$$

$$f = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{6}$$

5.3 Training of Neural Network

5.3.1 Training and Testing Samples

Feature selection and noise filtering described in Section 4.1 are performed both on the training and testing samples. We label each keypoint as 1 (for possible text keypoints) and -1 (for possible non-text keypoints) depending on whether it is within the bounding box of the ground truth. Of the labeled keypoints, 67% are randomly sampled for training while the remaining samples are for testing. The accuracy is shown in Table 2.

Table 2: Accuracy for the three databases

	Num. of SIFTs	Non-text	Text	Accuracy
ICDAR	135,235	62.60%	37.40%	81.00%
SWT	193,198	82.70%	17.30%	87.90%
MSRG	138,912	80.20%	19.80%	86.46%

5.3.2 Hidden Layer Number

In order to obtain the optimal training accuracy, we test the hidden layer number n of the neural network. The results are shown in Figure 13 and Table 3, respectively. It can be observed that the training accuracy becomes stable when n increases to 40 and beyond, while the time cost keeps increasing. In our experiment, we choose 40 to be the number of hidden layers.

5.3.3 Effectiveness of the Feature Selection

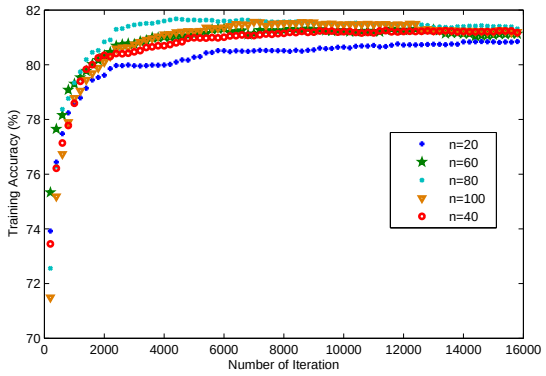

Figure 13: The performance of different hidden layer number

Table 3: Comparison of time per iteration

Hidden layer number	20	40	60	80	100
Time per iteration (seconds)	8	10.3	12.9	20.5	33

In order to show the efficiency and necessity of our feature selection step, we train a neural network whose training samples do not undergo this step on SWT database. For this database, there are 1,481,816 initial SIFT keypoints in total. 90.8% of them are non-text keypoints. After applying the two-step pre-processing, 694,556 SIFT keypoints are kept while 88.2% of them are removed as non-text keypoints. We then carry out four tests which are described in Table 4. The results are shown in Table 5.

Here, t_p represents the number of true positives, t_n represents the true negative, f_p represents the false positive, and f_n represents the false negative. $precision^{'} = \frac{t_p}{t_p+f_p}$. $recall^{'} = \frac{t_p}{t_p+f_n}$. The standard f measure follows the same rule as Equation 6.

As shown in the four cases, the standard f measure reaches the highest when training samples and test samples both undergone the text sensitive filtering step. Therefore, the filtering step is effective to enhance the capability of neural network.

5.4 Statistics of Region Growing

To justify the effectiveness of our scale-based region growing algorithm, we count the Text Region Overlapping Ratio (TROR)

Table 4: Illustration of The Four Tests

	Training samples filtering	Testing samples filtering
T1	Yes	Yes
T2	Yes	No
T3	No	Yes
T4	No	No

Table 5: Performance comparison of the four tests

	T1	T2	T3	T4
Num. of SIFTs	694,556	1,481,816	694,556	1,481,816
t_p	5.40%	3.40%	7.70%	5.20%
f_p	2.60%	2.10%	12.40%	9.60%
t_n	85.60%	88.80%	75.80%	81.20%
f_n	6.30%	5.80%	4.10%	4.00%
$precision^{'}$	67.50%	61.80%	38.30%	35.10%
$recall^{'}$	46.20%	37.00%	65.30%	56.50%
f **measure**	0.548	0.463	0.483	0.433

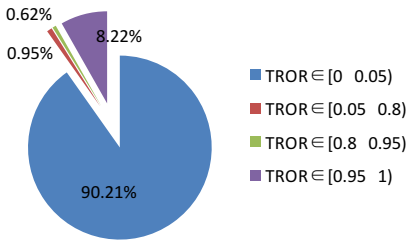

Figure 14: The pie chart of the TROR for all the grown regions generated from the three databases. Different colors represent grown regions with different value intervals of TROR. (Best viewed in color)

for every grown regions on the three databases mentioned above. The TROR for a grown region $R_c(k)$ is defined as the overlapped area of $R_c(k)$ with the ground truth of the text region divided by the total area of $R_c(k)$. Ideally, the region growing algorithm should only generate regions with the TROR equal to 0 or 1 (there is no misconnection between text region and non-text region).

The results are shown in Figure 14. 99.05% grown regions are the suitable regions we need. Among them, 90.22% regions have TROR less than 0.05 (they are grown on the non-text part of the testing images) and 8.22% ones have TROR larger than 0.95 (they are grown on the text part). Only 0.95% regions have TROR between 0.05 and 0.80. They are mostly regions grown on the interval between two nearby text line and can be handled by the post-processing steps.

Based on this experiment, the major part (> 95% area) of most grown regions are either within or outside the ground truth bounding boxes. This property is critical to enhance the performance of the text classifier using Equation 2.

5.5 Performance Comparison

5.5.1 *Comparison on ICDAR Database*

Though the algorithm in general does not include word separation part, to meet the demand of this database, we divide bounding boxes into isolated words using a simple algorithm similar to the division of text blocks. We adaptively use canny edge detection and the direct grey level projection depending on whether the background is simple or not (The background is treated as simple if its density of interest points is lower than a threshold). The results on this database from published paper are shown in Table 6. The lower half shows the algorithms proposed in ICDAR2003 and ICDAR2005. Becker [17] achieves the best performance in standard f measure of 0.62, but with a relatively large time cost for 14.4 seconds per image. The upper half shows the algorithms proposed after the competitions. Our algorithm gets the better recall rate than Epshtein [9] and Chen [5]. Because of our relatively simple word separation algorithm and the special evaluation metric (A bounding box containing two words can only achieve a precision and recall of 50%), our standard f measure is not very high. This problem can be solved by the text recognition step since the text lines can be separated into words more precisely in this step.

Our algorithm takes 1.65 seconds per image while Epshtein's algorithm takes 0.94 seconds per image [9]. After extracting the SIFT keypoints, our algorithm needs only 0.15 seconds on average. So the complexity of our overall detection algorithm is mainly driven by the SIFT extraction stage. Recently, there are some substitutions of SIFT, such as CARD [3], which is comparable to SIFT in discriminative property, but much more efficient and can be applied to real-time tasks. Such features may substitute SIFT feature used in our approach. Other fast features can also be adopted with a little modification of our algorithm. We will investigate this issue in our future work.

Table 6: Performance comparison in the database of ICDAR

Algorithm	Precision	Recall	f
Epshtein [9]	**0.73**	0.6	**0.66**
Chen [5]	**0.73**	0.6	**0.66**
Fabrizio [10]	0.46	0.39	0.43
Minetto [21]	0.63	0.61	0.61
Our algorithm	0.66	0.62	0.64
Chen [17]	0.6	0.6	0.6
Becker [17]	0.62	**0.67**	0.62

5.5.2 *Comparison on SWT Database*

As described in Section 5.1, the database is much more challenging than ICDAR. It requires to mark the whole text line without word separation. The results are shown in Table 7. Our algorithm outperforms other the state of the art algorithms on this database.

Table 7: Performance comparison in the database of SWT

Algorithm	Precision	Recall	f
Epshtein [9]	0.54	**0.42**	0.47
Chen [5]	0.45	0.39	0.42
Our algorithm	**0.58**	0.41	**0.48**

5.5.3 *Comparison on MSRG Database*

The above two subsections show the effectiveness of our algorithm in Latin, English and Roman numerals. These kinds of texts share some common properties, such as low variance of stroke width and a few letter categories. However, some kinds of language, such as hieroglyph characters, can probably exhibit large stroke width variances, complex structures and thousands of characters. To demonstrate the robustness of our algorithm, we test it on our multilingual database which contains texts of Latin, English, Roman numeral and hieroglyphs. Because there are no published results in this new dataset, we implement some of the state-of-the-art algorithms. As shown in Table 8, our algorithm achieves the best performance on this database.

All of the three databases contain mainly vertical and horizontal text, or tiled text within a certain degree. Our algorithm has the potential to handle seriously angled texts because both the SIFT features and the scale-based region growing algorithm are invariant to rotations. We will explore it in our future work.

Table 8: Performance comparison in the database of MSRG

Algorithm	Precision	Recall	f
Epshtein [9]	0.41	0.39	0.4
Chen [5]	0.4	0.37	0.38
Our algorithm	**0.54**	**0.55**	**0.54**

5.5.4 *Training in Different Datasets*

To further evaluate the robustness of our algorithm, we perform the cross-database experiment. We test the performance of our algorithm by the neural network trained on other datasets. When trained on ICDAR, the f measure drops 0.11 and 0.06 on SWT and

MSRG respectively. When trained on SWT, the f measure drops 0.04 and 0.04 on ICDAR and MSRG respectively. When trained on MSRG, the f measure drops 0.04 and 0.06 on ICDAR and SWT respectively.

The background of ICDAR is different from that of SWT and MSRG, so the performance drops most when trained on this dataset and tested on other ones. If the training set is MSRG or SWT, the performance drops at most by 6% when tested on other datasets. We consistently adopt the same parameter learned from experimental results. The result indicates that there are some common properties of the text keypoint in different databases, or even in different languages. These properties are well explored by our algorithm.

6. CONCLUSIONS

We present a novel approach for multilingual text detection from scene image in the wild. In this algorithm, we explore three types of unique properties of SIFT feature, including the often ignored statistical properties in a local region. Based on the properties, text regions in images are identified by the scale-based region growing algorithm. Without character recognition step as used in some previous work, our algorithm works well in Latin, English, Roman numeral and hieroglyph in scene images. The experimental results on three databases demonstrate the effectiveness and robustness of the proposed approach to detect various kinds of text in complex background.

The most time-consuming part of our algorithm is SIFT extraction. To address this issue, we will resort to other features (e.g., CARD [3]), faster extraction algorithms (e.g., a SIFT extractor with paralleling technology [31]), or fast approximated SIFT extractor [11] in our future work.

There are several possible extensions of this work. The discovered scene text information can be efficiently combined with the extracted SIFT features to provide more reliable context clue for visual search. In addition, SIFT features on text can be used as a valuable clue to improve the performance of OCR.

7. ACKNOWLEDGMENT

This work was supported in part to Prof. Houqiang Li by NSFC 61272316, in part to Dr. Qi Tian by ARO grant W911NF-12-1-0057, NSF IIS 1052851, Faculty Research Awards by Google, FXPAL, and NEC Laboratories of America, and 2012 UTSA START-R Research Award respectively, and in part by NSFC 61128007.

8. REFERENCES

[1] http://algoval.essex.ac.uk/icdar/.

[2] http://research.microsoft.com/enus/um/people/eyalofek/text_detection_database.zip.

[3] M. Ambai and Y. Yoshida. Card: Compact and real-time descriptors. In *Proc. ICCV*, pages 97–104, 2011.

[4] M. Block, M. R. Ortegón, A. Seibert, J. Kretzschmar, and R. Rojas. *SITT: A Simple Robust Scaleinvariant Text Feature Detector for Document Mosaicing*. Freie Univ., Fachbereich Mathematik und Informatik, 2007.

[5] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B. Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *Proc. ICIP*, pages 2609–2612, 2011.

[6] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *Proc. CVPR*, volume 2, pages II–366, 2004.

[7] X. Chen and A. L. Yuille. A time-efficient cascade for real-time object detection: With applications for the visually impaired. In *Proc. CVPR*, pages 28–28, 2005.

[8] T. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proc. VISAPP*, 2009.

[9] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Proc. CVPR*, pages 2963–2970, 2010.

[10] J. Fabrizio, M. Cord, and B. Marcotegui. Text extraction from street level images. *CMRT*, 3, 2009.

[11] M. Grabner, H. Grabner, and H. Bischof. Fast approximated sift. In *Proc. ACCV*, pages 918–927. 2006.

[12] I. Haritaoglu. Scene text extraction and translation for handheld devices. In *Proc. CVPR*, volume 2, pages II–408, 2001.

[13] K. Jung, K. In Kim, and A. K Jain. Text information extraction in images and video: a survey. *Pattern recognition*, 37(5):977–997, 2004.

[14] M. Li. Texture moment for content-based image retrieval. In *Proc. ICME*, pages 508–511, 2007.

[15] J. Liang, D. Doermann, and H. Li. Camera-based analysis of text and documents: a survey. *IJDAR*, 7(2-3):84–104, 2005.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[17] S. M. Lucas. Icdar 2005 text locating competition results. In *Proc. ICDAR*, pages 80–84, 2005.

[18] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *Proc. ICDAR*, volume 2, pages 682–687, 2003.

[19] C. Mancas-Thillou and B. Gosselin. *Natural scene text understanding*. PhD thesis, PhD thesis, Faculté Polytechnique de Mons, Belgium, 2006.

[20] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *TPAMI*, 27(10):1615–1630, 2005.

[21] R. Minetto, N. Thome, M. Cord, J. Fabrizio, and B. Marcotegui. Snoopertext: A multiresolution system for text detection in complex visual scenes. In *Proc. ICIP*, pages 3861–3864, 2010.

[22] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Proc. ACCV*, pages 770–783. 2010.

[23] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[24] P. Shivakumara, T. Q. Phan, and C. L. Tan. A laplacian approach to multi-oriented text detection in video. *TPAMI*, 33(2):412–419, 2011.

[25] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proc. CVPR*, pages 1470–1477, 2003.

[26] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proc. MIR*, pages 427–434, 2008.

[27] Q. Tian, S. Zhang, W. Zhou, R. Ji, B. Ni, and N. Sebe. Building descriptive and discriminative visual codebook for large-scale image applications. *Multimedia Tools and Applications*, 51(2):441–477, 2011.

[28] W. Wu, X. Chen, and J. Yang. Incremental detection of text on road signs from video with application to a driving assistant system. In *Proc. ACM MM*, pages 852–859, 2004.

[29] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Proc. CVPR*, pages 1083–1090, 2012.

[30] Q. Ye, Q. Huang, W. Gao, and D. Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23(6):565–576, 2005.

[31] Q. Zhang, Y. Chen, Y. Zhang, and Y. Xu. Sift implementation and optimization for multi-core systems. In *Proc. IPDPS*, pages 1–8, 2008.

[32] Q. Zheng, K. Chen, Y. Zhou, C. Gu, and H. Guan. Text localization and recognition in complex scenes using local features. In *Proc. ACCV*, pages 121–132. 2011.

[33] Y. Zhong, H. Zhang, and A. K. Jain. Automatic caption localization in compressed video. *TPAMI*, 22(4):385–392, 2000.

[34] W. Zhou, H. Li, Y. Lu, and Q. Tian. Large scale image search with geometric coding. In *Proc. ACM MM*, pages 1349–1352, 2011.

[35] W. Zhou, H. Li, Y. Lu, and Q. Tian. Principal visual word discovery for automatic license plate detection. *TIP*, 21(9):4269–4279, 2012.

[36] W. Zhou, Y. Lu, H. Li, and Q. Tian. Scalar quantization for large scale image search. In *Proc. ACM MM*, pages 169–178. ACM, 2012.