Image and Vision Computing 31 (2013) 255–274

Contents lists available at SciVerse ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

A text reading algorithm for natural images ☆

Álvaro González *, Luis Miguel Bergasa

Department of Electronics, University of Alcalá, University Campus, 28871 Alcalá de Henares (Madrid), Spain

ARTICLE INFO

Article history:
Received 5 July 2012
Received in revised form 30 October 2012
Accepted 22 January 2013

Keywords:
Text detection
Text recognition
Character recognition
Character segmentation
Natural images
Scene text detection

ABSTRACT

Reading text in natural images has focused again the attention of many researchers during the last few years due to the increasing availability of cheap image-capturing devices in low-cost products like mobile phones. Therefore, as text can be found on any environment, the applicability of text-reading systems is really extensive. For this purpose, we present in this paper a robust method to read text in natural images. It is composed of two main separated stages. Firstly, text is located in the image using a set of simple and fast-to-compute features highly discriminative between character and non-character objects. They are based on geometric and gradient properties. The second part of the system carries out the recognition of the previously detected text. It uses gradient features to recognize single characters and Dynamic Programming (DP) to correct misspelled words. Experimental results obtained with different challenging datasets show that the proposed system exceeds state-of-the-art performance, both in terms of localization and recognition.

1. Introduction

Automatic text recognition has traditionally focused on analyzing scanned documents. However, during the last years digital cameras have started to be embedded in low-cost consumer products such as mobile phones and Tablet PCs, so user applications related to digital image processing have become very popular. Nevertheless, automatic text recognition in natural images still remains one of the most challenging problems in computer vision. In addition, as textual information can be found on any environment, both indoors and outdoors, the range of applications of automatic text reading systems can be wide, from support to visually impaired people to automatic geocoding of businesses, including support to robotic navigation in indoor and outdoor environments, image spam filtering, driver assistance or translation services for tourists, among others.

Up to now, most works have focused on concrete subsets of the problem, such as extracting text in CD cover images [1] or segmenting text in web images [2]. This is due to the wide variety of text appearance because of different fonts, thicknesses, colors, sizes, textures, as well as the presence of geometrical distortions and partial occlusions in the images, different lighting conditions and image resolutions, different languages, etc. In this paper, we propose a system to read text in any kind of scenario, both indoors and outdoors, in natural images. We simply constrain to machine-printed text and English language. To benchmark the performance of the proposed system, results have been obtained with several datasets that include images in different scenarios and situations. These datasets were released for the robust reading competitions held in the frame of the ICDAR (International Conference on Document Analysis and Recognition) 2003, 2005 and 2011 conferences. Most of researchers in this field use these datasets as a benchmark.

There are a number of contributions in this paper. First, a segmentation method based on a combination of Maximally Stable Extremal Regions (MSER) [3] and a locally adaptive thresholding method has been proposed. Second, a thorough study on different features to describe text has been carried out and the main results are shown here. A set of fast-to-compute features to discriminate between character and non-character has been proposed. It has also been proposed to use a restoration stage based on position and size features in order to bring back characters erroneously rejected. The results section will show the importance of this stage. On the other hand, a new feature is proposed for recognizing single characters. Finally, misspelled words are corrected using DP with substitution costs based on the confusion matrix of the character recognizer.

The remainder of the paper is organized as follows. In Section 2 we overview the main state-of-the-art methods. Section 3 presents a short general description of the proposed method. Section 4 describes the study that has been carried out in order to obtain a set of features that allow us to distinguish characters from non-characters. Section 5

☆ This paper has been recommended for acceptance by Enrique Dunn.

* Corresponding author. Tel.: +34 918856726; fax: +34 918856591.
E-mail addresses: alvaro.g.arroyo@depeca.uah.es (Á. González), bergasa@depeca.uah.es (L.M. Bergasa).

0262-8856/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.imavis.2013.01.003

describes the text location algorithm, while Section 6 explains the recognition approach. Experimental results are widely described in Section 7. Section 8 ends the paper highlighting the main conclusions and future works.

2. Related work

Automatic text location and recognition has been one of the main challenges in computer vision ever. Most of the works in this field are based on locating the text areas, so it is difficult to make an overview of all the implemented methods, since there has been thorough research on text location. In this section we focus on similar works from the last decade.

Yao et al. [4] use locally adaptive thresholding to segment an image. Then, certain geometric features are extracted from connected components and used to discard most non-character connected components (CCs) by a cascade of classifiers. Non-discarded CCs are fed into a SVM in order to be classified into characters or non-characters and the CCs verified by the classifier are merged into candidate text regions according to certain neighboring properties such as proximity and height.

Neumann and Matas [5] assume characters to be MSERs [3] in certain scalar image projections. The resulting connected components of the MSER detection stage are classified into character and non-character based on certain basic features such as aspect ratio or color consistency. Text line candidates are formed based on geometric character attributes and character recognition is applied together with a typographic model to correct inconsistencies.

Epshtein et al. in [6] propose the Stroke Width Transform (SWT), a local image operator which computes per pixel the width of the most likely stroke containing the pixel. The idea under the SWT is to look for matched pairs of pixels in a small region with corresponding opposing gradients. Pixels with similar stroke width are merged into connected components and letter candidates are found from certain geometric basic rules concerning aspect ratio or variance of the stroke width. Letters are grouped into text lines if they have similar features such as stroke width, height or average color.

Chen et al. in [7] follow the same idea of [6]. However, they only apply the SWT on the connected components resulting after a MSER detection stage. In addition, they propose an alternative way of computing the SWT from a binary image.

Pan et al. in [8] use a text region detector to estimate probabilities of the text position and scale information. This detector is based on Histogram of Oriented Gradients (HOG) [9] and a WaldBoost cascade classifier [10] on image pyramids. The information extracted from each scale is merged into a single text confidence map and text scale map. Secondly, the gray-level image is segmented using the Niblack's local binarization algorithm [11] and a connected component analysis is carried out with a conditional random field (CRF) model [12] to assign candidate components as text and non-text by considering both unary component properties, such as width or height, and binary contextual component relationships, such as spatial distance or overlap ratio.

Chen and Yuille [13] use a set of informative features based on the intensity, gradient direction and intensity gradient. Weak classifiers, using joint probabilities for feature responses on and off text, are used as input to an AdaBoost cascade classifier. Regions selected by the classifier are clustered into groups according to their location and size. An adaptive binarization algorithm is applied and connected components are extracted. The connected components are grouped into lines followed by an extension algorithm to find missing boundary letters.

Hanif and Prevost [14] use a small set of heterogeneous features (Mean Difference Feature (MDF), standard deviation (SD) and HOG) which are spatially combined to build a large set of features. A neural-network-based localizer learns the localization rules automatically.

Most of the state-of-the-art methods fail in images with a complex background or when the background illumination is not homogeneous, that is, when it is brighter in one part of the image and darker in others, or the image is blurred. We propose a segmentation method based on a combination of a global and a local thresholding methods that works fine in images with complex background and when the illumination of the image is heterogeneous or it is blurred. Unlike other methods, we have also carried out an analysis on different features in order to find which are the best to describe text.

Most of the work in the area of automatic text recognition is based locating text areas and applying commercial OCRs where text areas have been found in the image. However, due to the huge variability of font styles, thickness, colors, texture, resolution or illumination, among other factors, the scenarios where commercial OCRs work well are very limited. Some attempts to develop robust character recognition techniques for natural images have been carried out.

A comparison of six different types of local features (Shape Context [15], Geometric Blur [16], Scale Invariant Feature Transform [17], Spin image [18], Maximum Response of filters [19] and Patch descriptor [20]) is presented in [21]. Using a bag-of-visual-words representation, the authors achieve a performance superior to commercial OCR systems for images of street scenes containing English and Kannada characters.

In [22], text recognition is carried out using contour-based features and a multi-class SVM classifier, which is trained using synthetic data. The method takes into account multiple hypotheses in text localization and recognition stages and selects the best one in the final stage using a unigram language model. Grayscale features and Convolutional Neural Networks (CNNs) are proposed in [23] to achieve character recognition. Grayscale features are also used in [24], but character images are divided into 8-by-8 pixel sub-patches. A simple binary classifier is used to decide if each of these sub-patches corresponds to text or not.

A new feature based on local symmetry and orientation is proposed in [25] for character recognition, while [26] proposes to use the Fisher Component Analysis (FCA).

We propose a new feature based on simple gradient features to improve text reading in natural images. We also use a unigram language model to correct misspelled words. The experimental results obtained show the competitiveness of the proposed method.

3. General overview of the system

Fig. 1 shows the flowchart of the proposed framework, which is made up of two main blocks. The text location block aims at locating the text in the image precisely and to discard those parts of the image that do not contain text. It is thoroughly explained in Section 5. On the other hand, the recognition block treats to recognize the text detected in the previous stage. It is detailed in Section 6.

4. Text features analysis

In order to obtain a set of distinctive features capable of distinguishing character objects from non-character objects, we have made an analysis of certain text features under the ICDAR 2003 Reading Competition dataset. This dataset contains a total of 509 realistic images with complex background, captured in a wide variety of situations, with different cameras, at different resolutions and under different lighting conditions. The dataset is divided into two sections: a training set that contains 258 images and a test set which has 251 images. We have worked with the training set for the analysis that we present below.

There are 6185 single characters in the training set, but we have analyzed only 5438 characters, because not all samples are valid due to different reasons, such as too small size or partial occlusions. We have binarized every sample and we have computed several geometric

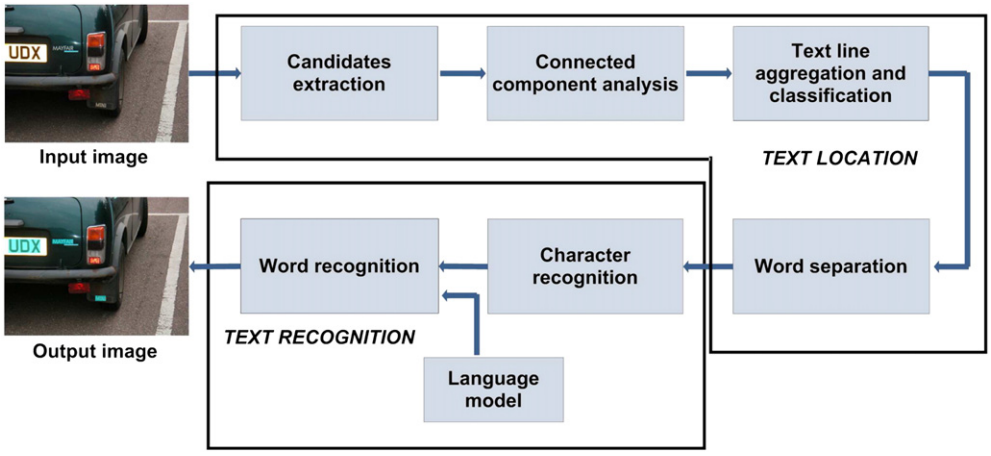

Fig. 1. Flowchart of the proposed framework.

features. In Fig. 2, we show some examples of the binarized characters from which we have computed the features in Eqs. (1)–(8).

$$Occupy\ rate = \frac{area}{height * width} \tag{1}$$

$$Aspect\ ratio = \frac{max(width, height)}{min(width, height)} \tag{2}$$

$$Compactness = \frac{area}{perimeter * perimeter} \tag{3}$$

$$Solidity = \frac{area}{convex\,area} \tag{4}$$

$$Occupy\ rate\ convex\ area = \frac{convex\ area}{height * width} \tag{5}$$

$$Stroke\ width\ size\ ratio = \frac{Stroke\ width}{max(height, width)} \tag{6}$$

$$Max\ stroke\ width\ size\ ratio = \frac{Max\ stroke\ width}{max(height, width)} \tag{7}$$

$$Stroke\ width\ variance\ ratio = \frac{Stroke\ width\ variance}{Stroke\ width} \tag{8}$$

The convex area is the area of the convex hull, which is the smallest convex polygon that contains the region.

If we represent the histogram of each of the features in Eqs. (1)–(8), we see that they follow a Gaussian distribution, but it cannot be drawn the same conclusion for non-character components, as it is shown in Figs. 3 and 4. The values of mean (μ) and standard deviation (σ) for each of these features are shown in Table 1.

Fig. 2. Some training samples.

5. Text location

The flowchart of our text location algorithm is shown in Fig. 5. We find letter candidates with a segmentation method based on MSER and a locally adaptive thresholding method. The resulting candidates are filtered using certain constraints based on the prior features shown in the previous section. Character candidates are grouped into lines and each line is classified into text or non-text using a classifier based on HOG. Finally, words within a text line are separated, giving segmented word areas at the output of the system.

5.1. Candidate extraction

First stage of our text detection method consists of finding letter candidates. We have combined two different region detectors used previously for text detection by other authors. MSER has been proved to be one of the best region detectors as it is robust against changes in scale, viewpoint and lighting conditions. However, it is very sensitive to image blur and MSER cannot detect small letters in blurred images. On the other hand, the image decomposition method proposed by Yao et al. [4] is able to detect most of the characters in an image, even small characters, but it produces too many letter false positives. It is a locally adaptive thresholding method, which computes a threshold over each pixel by a local mean. The output of this method is a three-layer image (white, gray and black) in which objects in white or black layers are candidate character pixels. Objects in the white layer are bright objects on darker regions in the original image, while objects in the black layer are dark objects on brighter regions. We propose to combine the complementary properties of MSER and Yao's methods.

The polarity of text in an image can be bright or dark with respect to its background. Consequently, we need to take into account that both situations can be present in an image. We search for both dark-on-bright and bright-on-dark letter candidates. Forthe subsequent CC analysis, we are assuming that the polarity of text in a line does not change. Firstly, input image is decomposed using the Yao's method, thus pixels are classified in one of these three layers: black, gray and white. Later, dark-on-bright MSER regions are extracted from the input image and are combined with the pixels of the black layer using an AND logical operation. In the meantime, bright-on-dark MSER regions are extracted and combined with the pixels of the white layer similarly. The conjunction of both methods helps to reduce the number of letter candidates given by the Yao's method and consequently it increases the efficiency of the whole method proposed

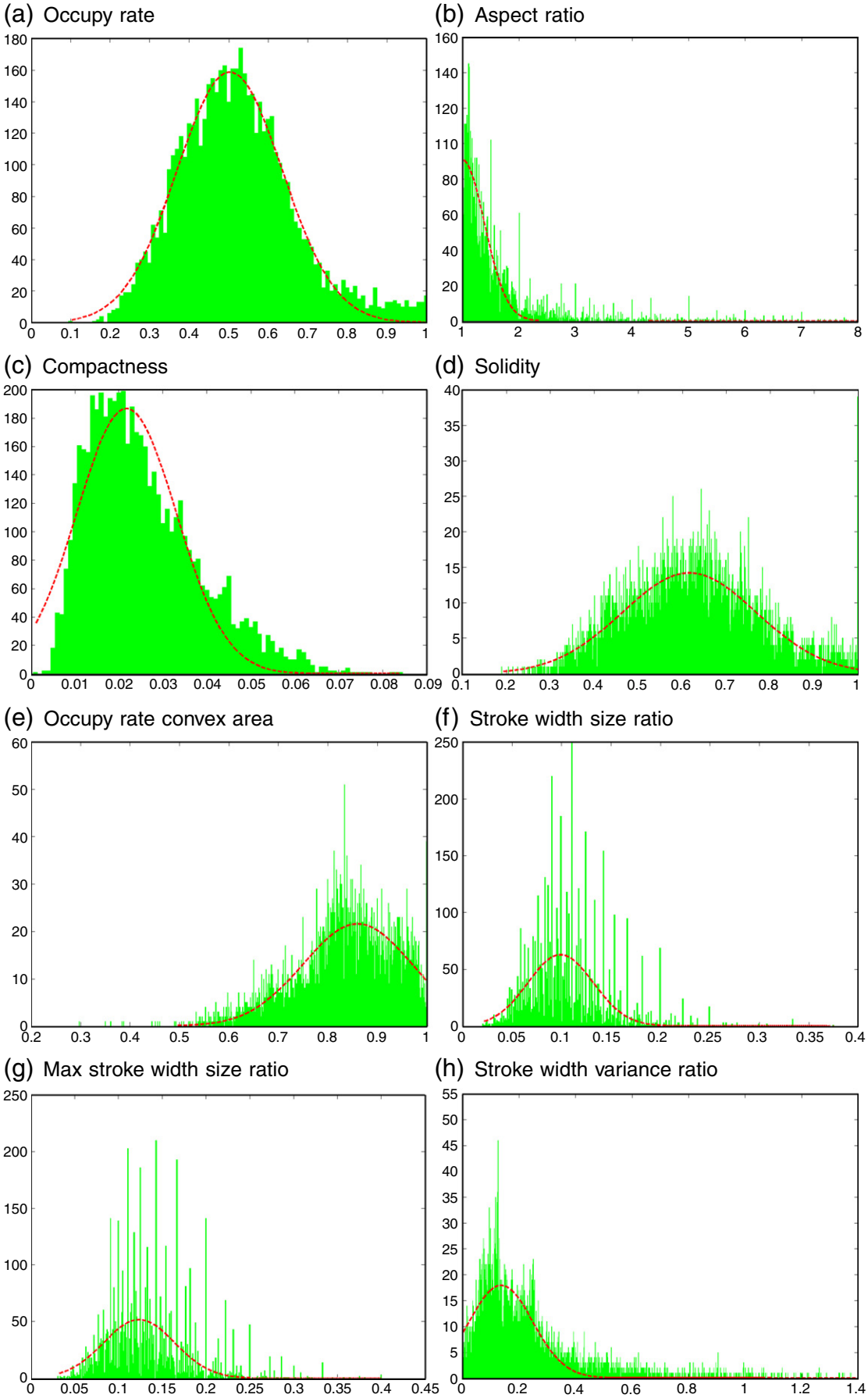

Fig. 3. Histograms of features vs approximated Gaussian functions for character components on ICDAR '03 train set.

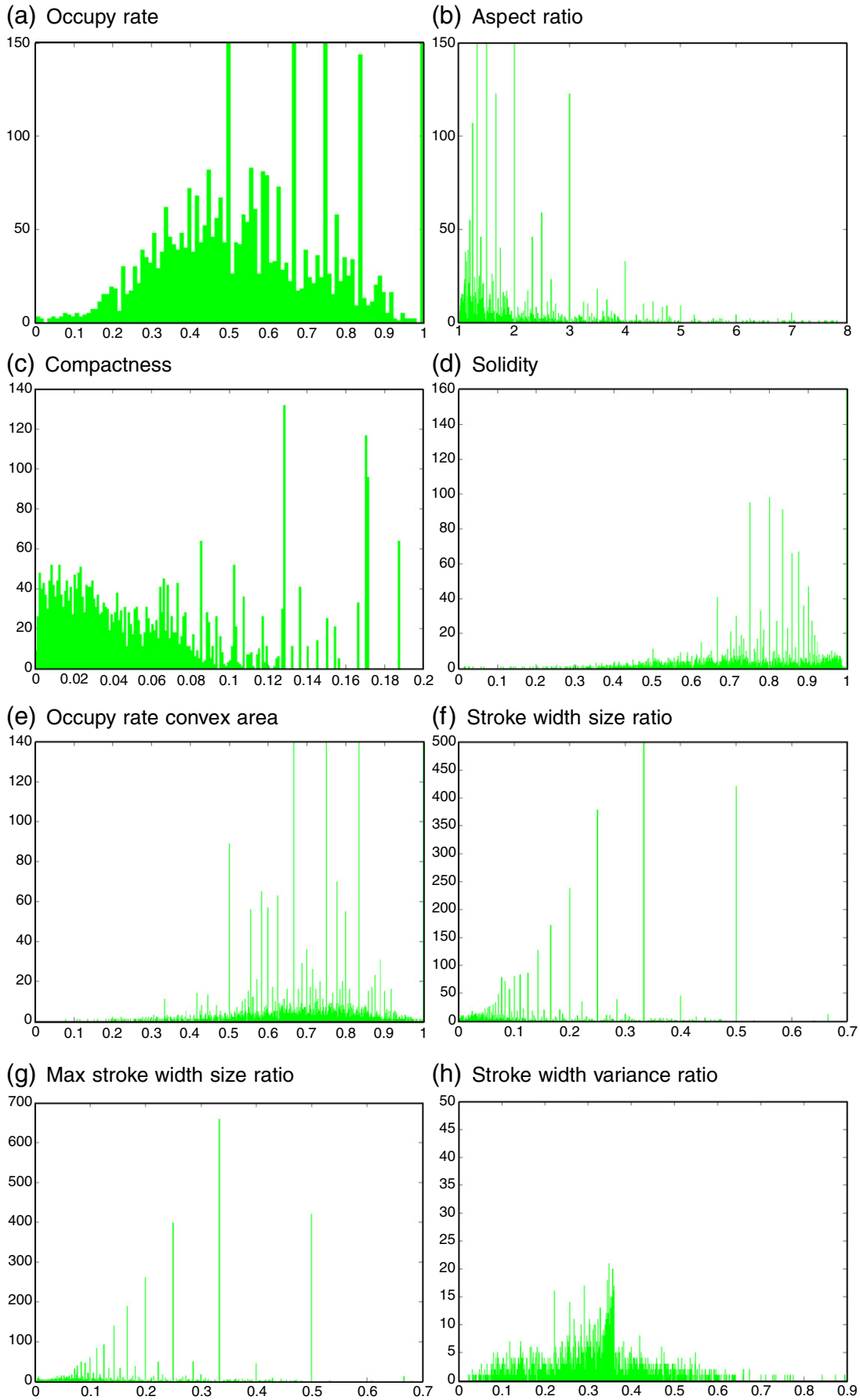

Fig. 4. Histograms of features for non-character components on ICDAR '03 train set.

Table 1
Mean and standard deviation of each feature.

Feature	Mean (μ)	Std deviation (σ)
Occupy rate	0.5227	0.1485
Aspect ratio	1.0	1.1357
Compactness	0.0257	0.0127
Solidity	0.6253	0.1514
Occupy rate convex area	0.8342	0.0993
Stroke width size ratio	0.1081	0.0406
Max stroke width size ratio	0.1356	0.0485
Stroke width variance ratio	0.2427	0.3005

here. It is also capable of separating some regions that MSER is not able to segment on its own. Fig. 6 shows the whole segmentation process.

5.2. Connected component analysis

After the previous stage, two binary images are obtained. The foreground CCs for each of these images are considered character candidates. In order to filter out non-text objects, the prior features obtained in Section 4 are applied. Features in Table 1 are computed for each candidate and those objects that are out of the range $(\mu - 2\cdot\sigma,\ \mu + 2\cdot\sigma)$ for at least one of the features, are rejected. Objects whose number of holes is higher than 2 are also rejected. We have chosen this value because the maximum number of holes that a letter can have is 2, as it is the case of 'B' and '8'. Only those holes with a certain minimum size with respect to the size of the candidate are taken into account, so small holes due to noise generated at the binarization stage are not considered. In addition, components whose size is too small are also filtered out. Experimentally, we have chosen a minimum font height of 10 pixels. By applying all these rules, most of text objects are accepted and a high percentage of non-text objects are pruned out. However, some text candidates can be rejected erroneously, especially those letters which have a high aspect ratio such as 'i' or 'l'. In order to bring back the mistakenly removed characters, we apply a method to restore them.

We are assuming that text is aligned horizontally. For each removed object, we look for the closest non-rejected objects to the left and to the

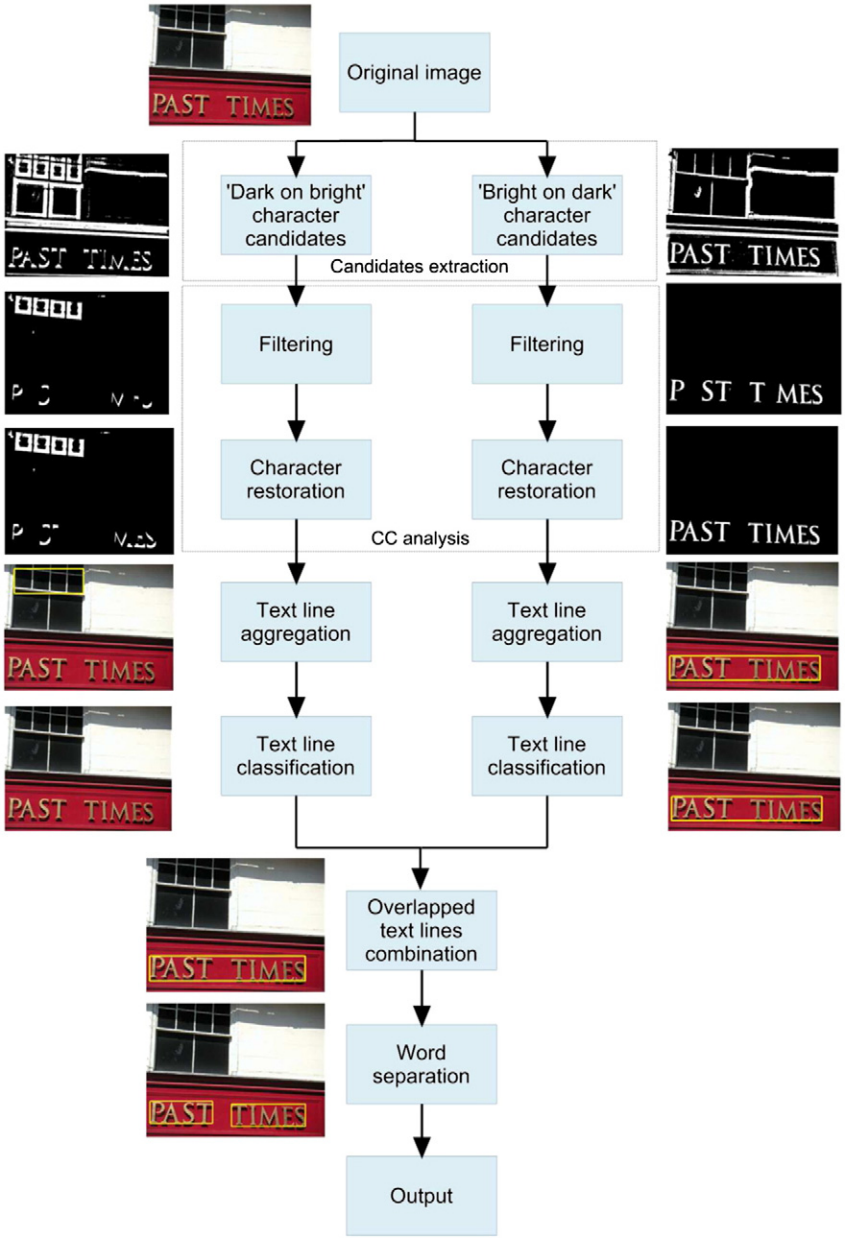

Fig. 5. Flowchart of the text location algorithm.

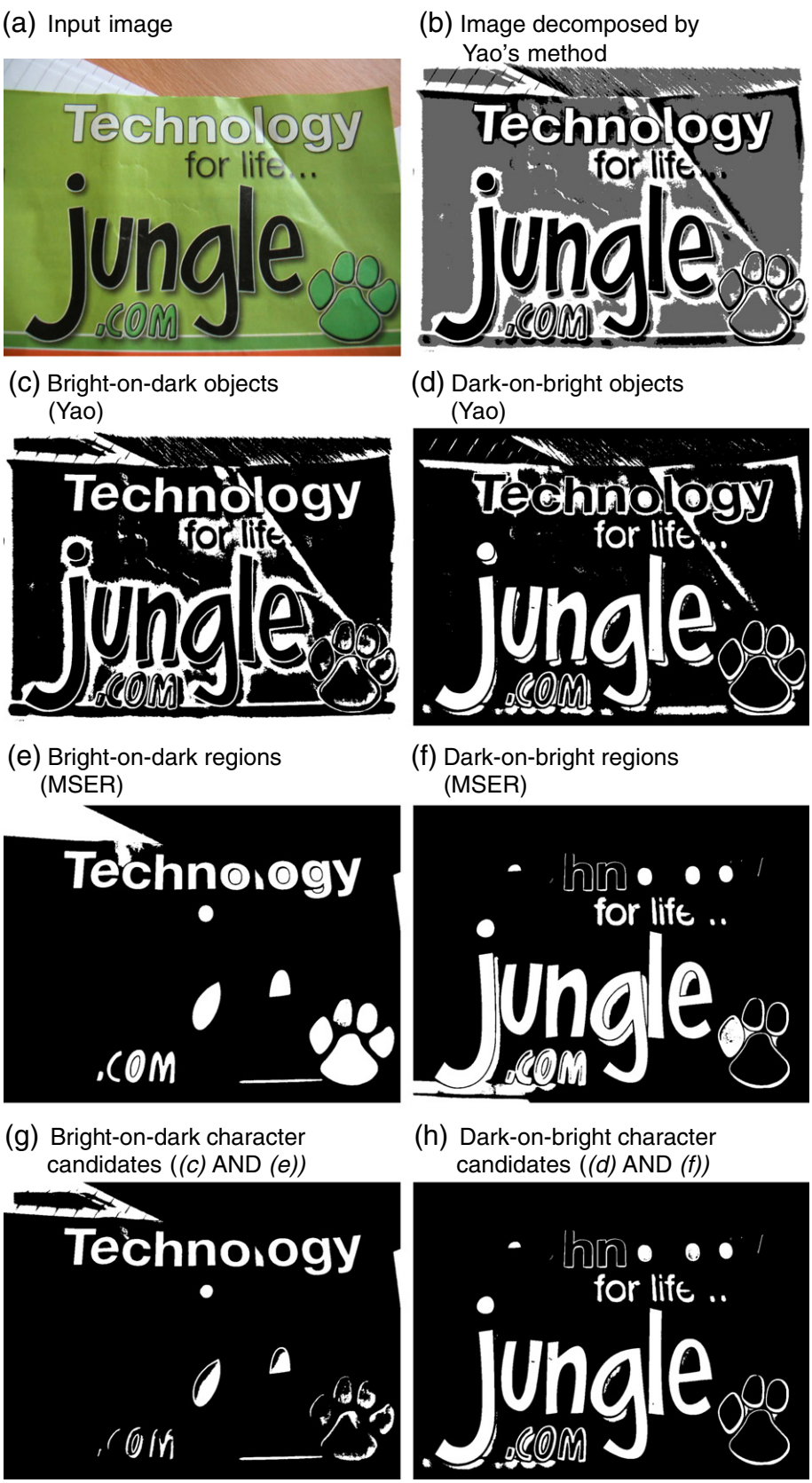

Fig. 6. Image segmentation.

right and apply some constraints between the non-removed and the removed objects. These rules relate to position, size, alignment and stroke width and are shown in Eqs. (9)–16). If all of these constraints are fulfilled, the removed component is accepted as a valid text object. The threshold values have been obtained using genetic algorithms, trying to maximize for the training set the function $G = \frac{Sensitivity + Specificity}{2}$,

where $Sensitivity = \frac{TP}{TP+FN}$ and $Specificity = \frac{TN}{TN+FP}$. *TP* stands for the number of true positives (characters correctly restored), *FP* is the number of false positives (non-characters erroneously restored), *TN* is the number of true negatives (non-characters correctly discarded) and *FN* stands for the number of false negatives (characters erroneously discarded).

$$\frac{Width_{removed}}{Height_{removed}} \leq 3 \tag{9}$$

$$Height_{removed} \geq 10 \tag{10}$$

$$Nholes_{removed} \leq 2 \tag{11}$$

$$\frac{Max(Height_{removed}, Height_{non\ removed})}{Min(Height_{removed}, Height_{non\ removed})} \leq 2 \tag{12}$$

$$\frac{Max(Stroke\ width_{removed}, Stroke\ width_{non\ removed})}{Min(Stroke\ width_{removed}, Stroke\ width_{non\ removed})} \leq 1.5 \tag{13}$$

$$\theta_1 < 40^\circ \text{ or } \theta_2 < 40^\circ \text{ or } \theta_3 < 40^\circ \tag{14}$$

$$Area_{overlap} > 0 \tag{15}$$

$$\frac{Max(Area_{removed}, Area_{non\ removed})}{Min(Area_{removed}, Area_{non\ removed})} \leq 8 \tag{16}$$

θ_1, θ_2 and θ_3 refer to the angles shown in Fig. 7(a). On the other hand, condition (15) means that the removed and the non-removed objects have to be overlapped in the magnified regions shown in Fig. 7(b). Conditions (14)–(16) were previously proposed by Ashida [27].

This method is done recursively for each removed component, until no more objects are restored.

The CC analysis explained in this section is carried out for each binary image separately because we are assuming that the polarity of text along a text line does not change. Fig. 8 shows an example of how the method detailed in this section works.

5.3. Text line aggregation

The accepted components in the previous section are considered letter candidates. Letters usually do not appear isolated in images. They form words and groups of letters. Moreover, characters on a line are expected to have some similar attributes, such as stroke width, height, alignment, adjacency and constant inter-letter and inter-word spacing. Initially, letter candidates are grouped into pairs if they fulfill the conditions (12)–(16). Then, two pairs are merged together if they share one of their initial or ending elements. This process of merging chains of candidates is repeated until no more groups can be merged. We are assuming that letters do not appear alone in an image, so we reject those groups which have less than 3 elements. The accepted groups are considered to be text lines. Fig. 9 shows an example.

Finding these groups of letters means that the proposed method cannot detect isolated characters and text lines with less than 3 objects, but it is an effective way of filtering out objects that were erroneously classified as letters in the previous stage of the algorithm.

5.4. Text line classification

Repeating structures such as windows, bricks or fences, commonly seen in urban scenes, can lead to mistakenly accepted text lines in the previous step of the algorithm. In order to reject false positives, a text line classifier has been implemented. It has been trained with the training set of the ICDAR dataset. We have manually labeled a set of positive and negative samples, in a proportion of 1:2 approximately. The classifier is based on SVM with linear kernel.

Three different types of features are extracted for each line: Mean Difference Feature (MDF), standard deviation (SD) and HOG. To compute these features, each line is resized to 64-by-256 pixels and a grid of 16-by-64 pixels is placed on the window, thus making a total of 16 blocks per window, as shown in Fig. 10(a). The mean value for each block is computed, giving a vector of 16 components. In order to make it independent to brightness, blocks are weighted by convoluting them with the masks shown in Fig. 10(b) and the absolute value is taken. This is the Mean Difference Feature (MDF). The standard deviation (SD) of each block is also computed, obtaining a 16 dimensional vector. Finally, HOG is computed for each block, resulting in a total of 16 histograms of 8 dimensions. Therefore, we get a feature set of 39 features (7 MDF, 16 SD and 16 HOG). All features are normalized by their euclidean norm in order to make the classifier independent to contrast.

5.5. Word separation

Finally, text lines are split into words. We compute the distance between each pair of adjacent letters and classify them into one of these two classes: intra-word separation or inter-word separation. Character separation inside a word tends to be constant, while separation of two consecutive characters that belong to two different words is consistently higher than the distance between two characters of the same word. The proposed method to separate words combines two ways of computing the gap between two adjacent letters.

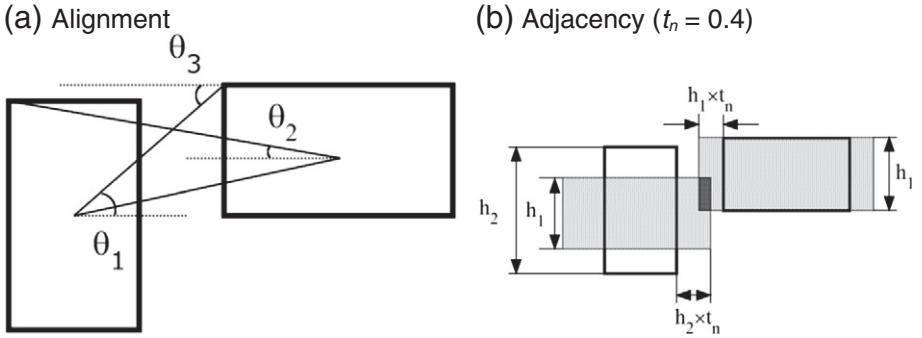

Fig. 7. Restoring conditions.

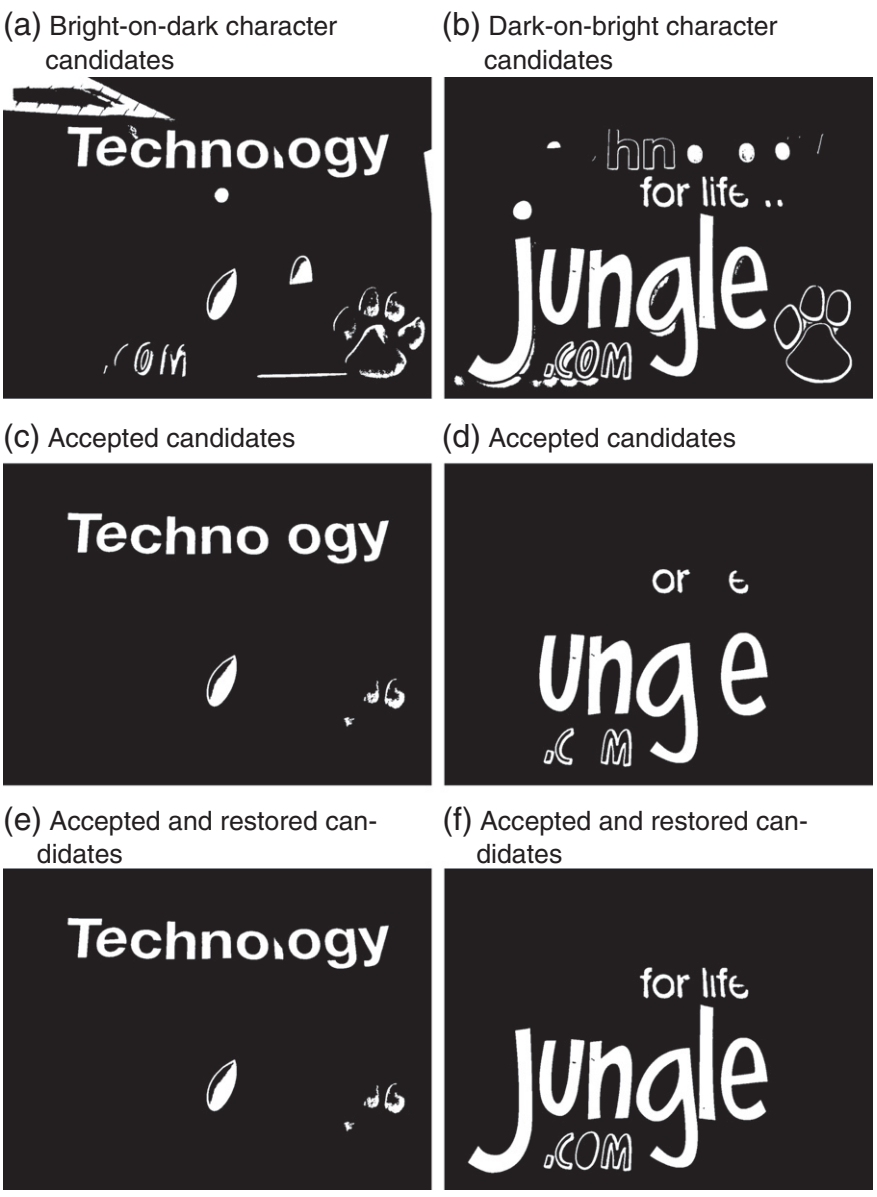

Fig. 8. Connected component analysis.

The first one consists of computing the euclidean distance between the intersection points of the convex hull of each character and the line that joins the centroids of both letters (see Fig. 11(a)). The second one consists of computing the horizontal distance between the bounding boxes of both letters (Fig. 11(b)). This method minimizes the disadvantages of using both metrics separately.

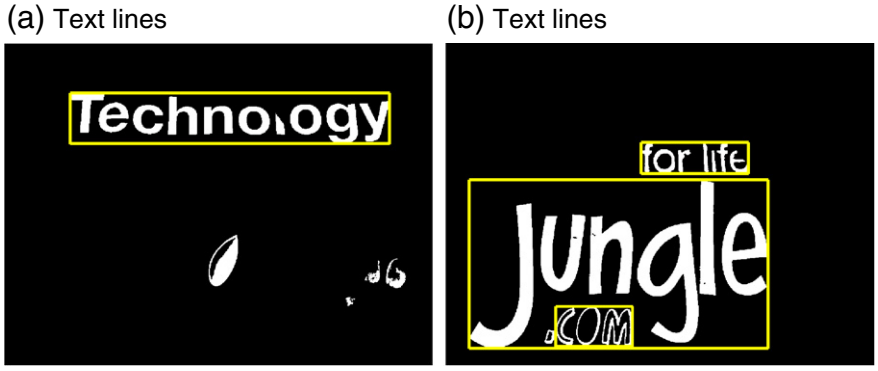

Fig. 9. Text line aggregation.

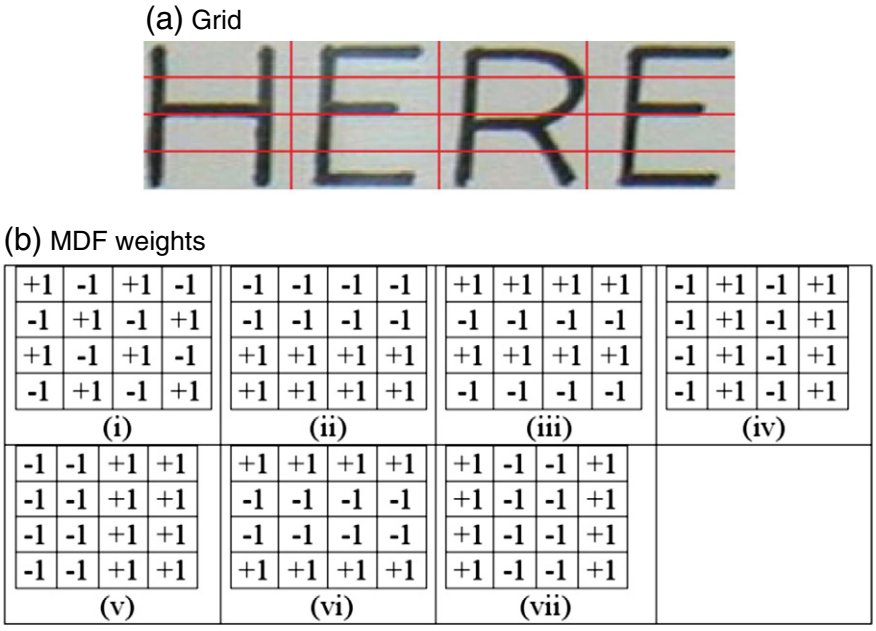

Fig. 10. MDF classifier.

6. Text recognition

The flowchart of the text recognition algorithm is shown in Fig. 12. Single characters are recognized using a classification approach based on K-Nearest Neighbors (KNN) and gradient direction features. Later, a unigram language model is applied in order to correct misspelled words.

6.1. Character recognition

The character recognizer takes each binarized character as input, it computes its feature vector and the object is classified into a class using a KNN approach. We propose a new feature vector based on simple and fast-to-compute features. We have baptized it as Direction Histogram (DH). We detect the edge pixels of the binarized objects and then we compute the direction of the gradient for each edge pixel. As it is a binarized image, there is only gradient on the edge pixels, so it is faster to compute. Later we quantize the direction of the gradients in the edge pixels into 8 bins: {−135°, −90°, −45°, 0°, 45°, 90°, 135°, 180°}, and we compute the histogram for each bin. The image is divided into 16 blocks in order to have spatial information, and the histograms for each block are concatenated into a 128-dimensional vector. As this method is based exclusively on the direction of the edge pixels, it is not affected by color neither intensity. Section 7 will show the robustness of the proposed feature compared to other widely used features. An overview can be seen in Fig. 13.

The classification is based on a KNN approach. The training dataset is composed of 5482 character samples extracted from the train set of the ICDAR 2003 Robust Reading Competition dataset, which has a wide diversity of fonts. Instead of giving only one solution, we propose to give different solutions with output probabilities. Firstly, the nearest K neighbors in the training dataset of the character to be classified are extracted. Each neighbor belongs to a class, i.e. each neighbor votes for a certain letter $S=\{s_1,s_2,\ldots,s_K\}$, where $s_i \in$ {'A', 'B', …, 'Z', 'a', 'b', …, 'z', '0', …, '9'} (62 classes). The set of distances from the object to each neighbor is $D=\{d_1, d_2, \ldots, d_K$. Fig. 14 shows all these definitions. We define the ratio between each distance to the minimum one as in Eq. (17).

$$R = \{r_1, r_2, \ldots, r_K\} = \left\{1, \frac{d_1}{d_2}, \ldots, \frac{d_1}{d_K}\right\} \tag{17}$$

We define p as the output probability of the nearest neighbor. We assume that the output probabilities of the following *K-1* nearest

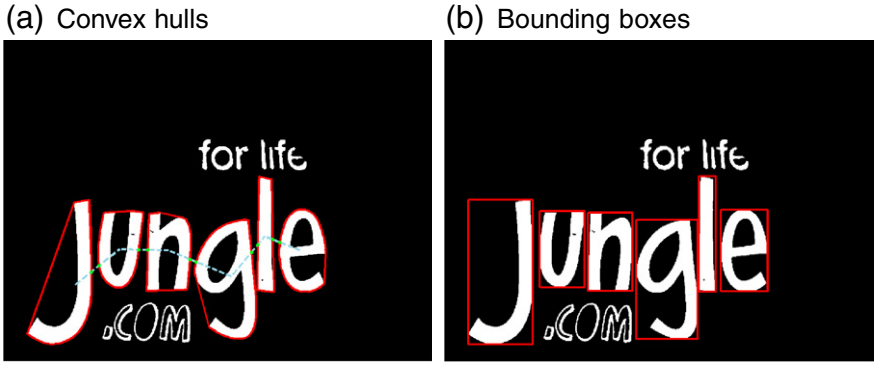

Fig. 11. Word separation.

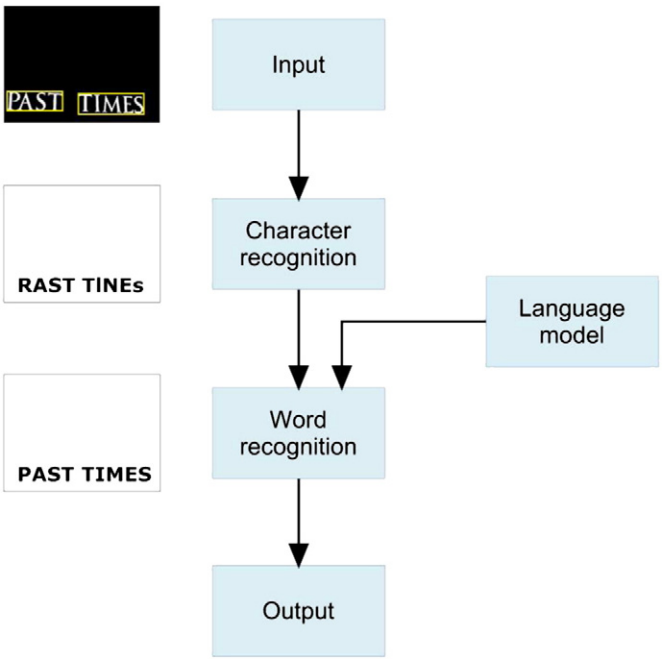

Fig. 12. Flowchart of the text recognition algorithm.

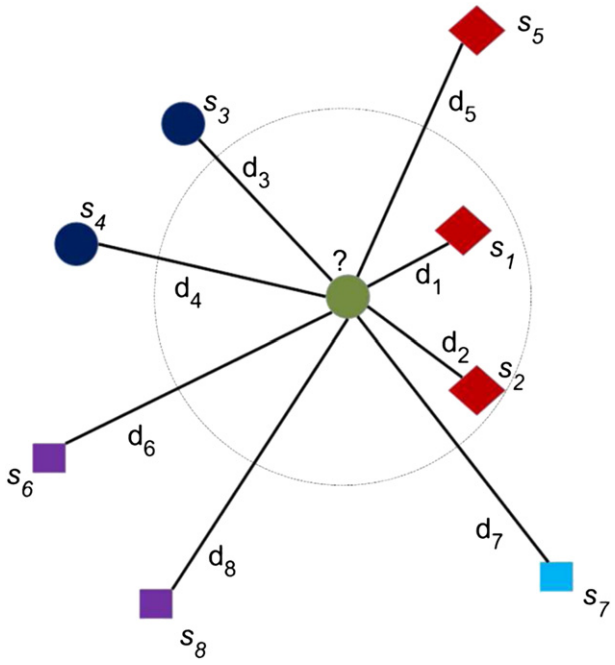

Fig. 14. Classification.

neighbors are related to p by the distance ratios defined in Eq. (17). Therefore, (Eq. (18)) must be fulfilled.

$$\sum_{i=1}^{K} r_i \cdot p = p + \frac{d_1}{d_2} \cdot p + \ldots + \frac{d_1}{d_K} \cdot p = 1 \tag{18}$$

The value of p can be easily computed from Eq. (18). The output probabilities of the object for every class can be computed using Eq. (19). Eq. (19) means that the probability of the object of belonging to class'A' is computed only from the neighbors that correspond to this class. The same is done for class 'B', 'C' and so on.

$$\begin{aligned} p_A &= \sum_{j=1}^{K} r_j \cdot p \quad \forall j \; s.t. \; s_j = A \\ p_B &= \sum_{j=1}^{K} r_j \cdot p \quad \forall j \; s.t. \; s_j = B \\ &\vdots \\ p_9 &= \sum_{j=1}^{K} r_j \cdot p \quad \forall j \; s.t. \; s_j = 9 \end{aligned} \tag{19}$$

With this method, when the object to be recognized is clearly a certain letter, there are many minima that vote for the same class, thus it will have a high output probability for that class. When it is not a clear case, the highest output probability tends to be low. The worst case would be when each neighbor is at a similar distance and votes for a different class, thus there would be K outputs with comparable probability. Therefore, it must be found a compromise in the value of K. A low value for K could be insufficient to have reliable output probabilities, while a high value could lead to errors, as the solutions with highest output probabilities would tend to those classes with a bigger number of samples. In our case, in which the training dataset is asymmetric, i.e. there are classes with a number of elements much higher than other classes, the number of nearest neighbors K has been set empirically to 25.

As the feature proposed is a distribution represented by histograms, it is natural to use the χ^2 test statistic. Therefore, the distances in the classification are computed using Eq. (20), where $h_i(k)$ and $h_j(k)$ denote the N-bin normalized histogram for objects i and j respectively.

$$D_{ij} = \frac{1}{2} \sum_{k=1}^{N} \frac{\left[h_i(k) - h_j(k)\right]^2}{h_i(k) + h_j(k)} \tag{20}$$

We use these output probabilities to split those characters that were not possible to separate in the segmentation step. Fig. 15 shows an example where the objects U and T are not separated in the segmentation step because they are 8-connected in the binary image. Initially only 4 objects (R, O, UT, E) are detected. The first candidates and the output probabilities of the first candidate for each object are, respectively, 'R' with $p = 0.97$, 'O' with $p = 1.0$, 'M' with $p = 0.42$ and 'E' with $p = 0.74$. It can be clearly seen that the

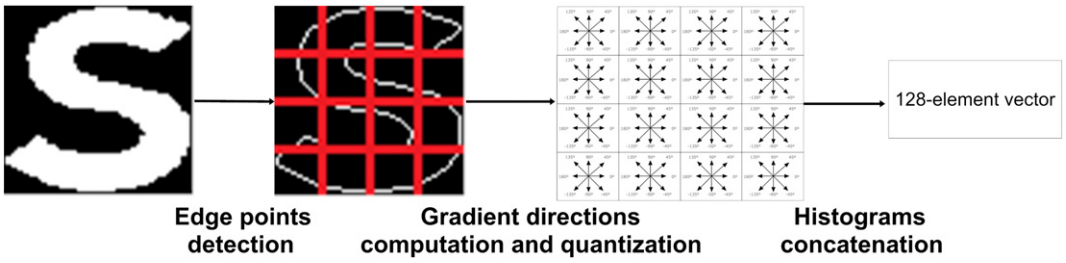

Fig. 13. Feature computation.

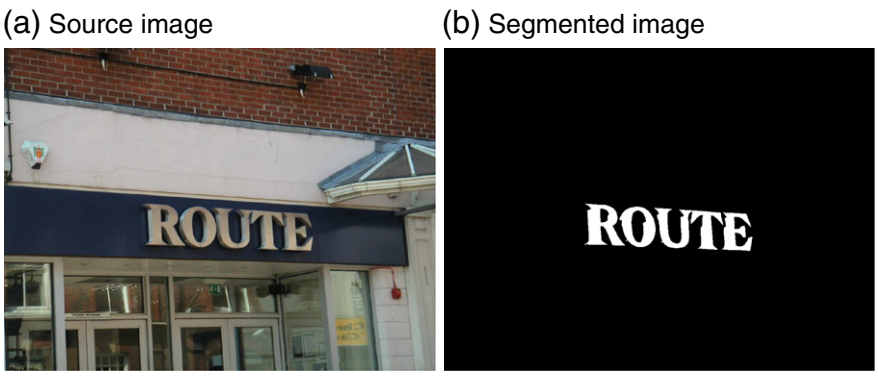

Fig. 15. Segmented image.

third object, which has not been correctly segmentated, has a lower probability with respect to the others. It suggests that something is wrong with it. We have developed an algorithm that uses this evidence together with the width of the object with respect to the others of the same word, in order to deal with this kind of situations by separating the objects that are highly likely to be wrongly split up. If the output probability of the first candidate is clearly lower than the output probabilities of the first candidates of the rest of the characters in the same word, and the width of the object is clearly higher than the rest of the characters in the word (which are not thin letters such as 'I', 'i' or 'l'), a separation is done where there is a minimum in the projection of the binarized object on the horizontal axis. After applying this method, we are able to solve situations like the one shown in Fig. 15, where the first candidate for each object would be: 'R' with $p=0.97$, 'O' with $p=1.0$, 'U' with $p=0.61$, 'T' with $p=1.0$ and 'E' with $p=0.74$, respectively. However, the problem of having erroneously split characters is also dealt in the word recognition step, where hypotheses of having wrongly separated objects are made and a set of candidates for each word is input into the word recognizer, as it will be explained later.

6.2. Word recognition

The OCR can erroneously classify some objects, thus leading to words that do not exist or are unlikely to be found in certain circumstances. In order to correct these errors and constrain the output of the OCR to a set of meaningful words, a language model is applied. We use a unigram language model because the benchmark with which we have tested our algorithm is given as independent words. This model is based on the British National Corpus (BNC) [28], which is a 100 million word collection of samples of written (90%) and spoken (10%) language from a wide range of sources. The corpus includes the number of times that each word occurs, which is essential to compute the prior probability for each word and to build our language model.

Let denote z as a noisy detection of some unknown word w, which $w \in W$, where W is the set of all possible terms (the BNC). The most likely word w_{MAP} that could have generated z is the one that maximizes the a posteriori probability $p(w|z)$, as shown in Eq. (21).

$$w_{MAP} = arg \max_{w \in W} p(w|z) \tag{21}$$

Applying the Bayes rule, Eq. (21) can be expressed as in Eq. (22).

$$w_{MAP} = arg \max_{w \in W} \frac{p(z|w)p(w)}{p(z)} \tag{22}$$

In Eq. (22), $p(z)$ is identical for all words w in the set W. Therefore, the denominator can be dropped as we are trying to find the maximum, thus Eq. (22) reduces to Eq. (23).

$$w_{MAP} = arg \max_{w \in W} p(z|w)p(w) \tag{23}$$

$p(w)$ is the prior probability of the word w occuring in a scene. We use word frequencies obtained from the BNC. $((z|w)$ is the likelihood of the OCR returning a sequence z when the underlying word is w. This probability is computed using DP, especifically the Viterbi algorithm. The idea behind DP consists of transforming one sequence into another one using edit operations that replace, insert or remove an element of the input sequence. Each operation has an associated cost. The goal is to find the optimum sequence, that is, the one with the lowest associated total cost. We work with probabilities instead of using costs, thus the optimum sequence would be the one with the highest associated probability.

If we assume that a misspelled word z has a set of letters $z=z_1, z_2, \ldots, z_m$ and the correct word w has a set of letters $w=w_1, w_2, \ldots, w_n$, DP aims at computing the probability $p(z|w)$ of recognizing the sequence z given w. The way of modeling it is by using a confusion matrix, which stores, for any given pair of letters, how likely a particular edit is to happen. Therefore, there are three confusion matrices: the

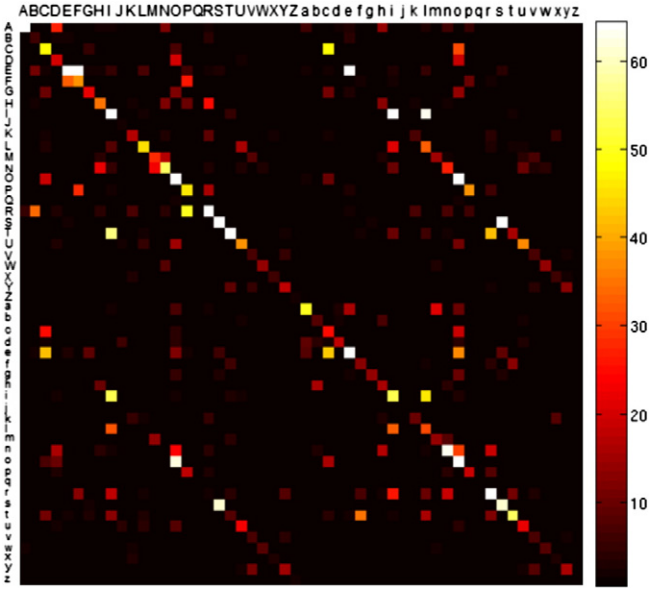

Fig. 16. Confusion matrix of the character recognizer.

Table 2
Text location ICDAR '03 dataset.

Algorithm	Precision	Recall	f
Pan et al. [8]	0.67	0.70	0.69
Our system	**0.81**	**0.57**	**0.67**
Epshtein et al. [6]	0.73	0.60	0.66
Chen et al. [7]	0.73	0.60	0.66
Lee et al. [34]	0.69	0.60	0.64
1st ICDAR '05 [30]	0.62	0.67	0.62
Yao et al. [4]	0.64	0.60	0.61
Alex Chen [30]	0.60	0.60	0.58
Zhang and Kasturi [35]	0.67	0.46	0.55
1st ICDAR '03 [27]	0.55	0.46	0.50

Table 3
Text location on ICDAR '11 Chall. 1 (%).

Algorithm	Precision	Recall	H. mean
Our system	**89.23**	**70.08**	**78.51**
Textorter	85.83	69.62	76.88
TH-TextLoc	80.51	73.08	76.62
TDM IACAS	84.64	69.16	76.12
OTCYMIST	64.05	75.91	69.48
SASA	67.82	65.62	66.70
Text Hunter	75.52	57.76	65.46

substitution matrix,the insertion matrix and the deletion matrix. For instance, for the pair of letters x and y, the substitution matrix $sub[x,y]$ keeps the count of how often x is recognized as y. The insertion matrix $ins[x,y]$ keeps the count of how often y is inserted after x, while the deletion matrix $del[x,y]$ keeps the count of how often y is removed after x. Therefore, insertion and deletion are conditioned on the previous character. In our case, we propose a different approach. As we do not have enough reliable data to compute the insertion and deletion matrices, we do not use them, but only the substitution matrix. We deal with deletions and insertions in a different way. As it was stated in the previous subsection, sometimes characters are erroneously separated, as a character can be wrongly split up into two parts, or two characters can be 8-connected and thus cannot be separated. In order to deal with thefirst case (objects that are erroneously broken into two parts), we make the hypothesis that each character could have been wrongly separated, and we join each pair of adjacent binarized objects and apply the character recognizer. This is equivalent to making a deletion in the sequence. To deal with the second case (characters wrongly detected as one), we separate each object into two parts and apply the character recognizer to each part. The point of separation is given by the minimum of the projection of the

Fig. 17. Text location results on some images from the ICDAR 2003/2005 dataset.

Table 4
Text location on ICDAR '11 Chall. 2 (%).

Algorithm	Precision	Recall	H. mean
Kim's method	82.98	62.47	71.28
Our system	**72.67**	**56.00**	**63.25**
Yi's method	67.22	58.09	62.32
TH-TextLoc	66.97	57.68	61.98
Neumann's method	68.93	52.54	59.63
TDM IACS	63.52	53.52	58.09
LIP6-Retin	62.97	50.07	55.78
KAIST AIPR system	59.67	44.57	51.03
ECNU-CCG method	35.01	38.32	36.59
Text Hunter	50.05	25.96	34.19

binarized object over the horizontal axis. This separation is equivalent to making an insertion in the sequence. As a result, we generate a set of candidates $Z = z^1, z^2, \ldots, z^L$ for each sequence z. Then, for each candidate z^i, we apply the Viterbi algorithm to compute the most likely word that could have it generated. In other words, each candidate z^i has its own most likely solution w^i with an output probability p^i. Finally, the output of the word recognizer w for a certain sequence z is the word w^i with higher output probability p^i.

The substitution matrix is computed from the OCR confusion matrix $C[x,y]$, which is shown in Fig. 16. This matrix keeps the count of the number of times that character x is recognized as y. In principle, the substitution matrix $sub[x,y]$, which keeps the probability that a certain character x is recognized as y, would be obtained by normalizing each row in $C[x,y]$. However, many elements in $C[x,y]$ are equal to zero, and we do not want to assign non-zero probabilities, because a non-zero probability for a certain substitution would mean that the substitution is not possible to happen. In order to avoid this, the substitution matrix is computed by applying Add-1 smoothing to the confusion matrix, as shown in Eq. (24), where V is the number of classes, i.e. the number of characters ($V = 52$).

$$subs[x,y] = \frac{C[x,y] + 1}{\sum_{y=1}^{V} C[x,y] + V} \tag{24}$$

Fig. 18. Text location results on some images from the ICDAR 2011 Challenge 1 dataset.

7. Experimental results

We evaluate the proposed method by running it on several public datasets and comparing to the state of the art. The chosen datasets have been used as a benchmark for most of researchers working in the field of text location and recognition in the last decade. A Robust Reading Competition was organized at ICDAR 2003 [29,27]. The competition was divided into three sub-problems: text location, character recognition and word recognition. Here, we will show our results for the three problems. The first one received five entries, while there were no participants in the character recognition and word recognition problems. A new competition was held at ICDAR 2005 [30] using the same dataset. Again, the participants only took part in the text locating problem. The dataset released for these competitions was divided into two sections: a train set that contains 258 images and 1157 words and a test set with 251 images and 1111 words. The images were obtained in different outdoor and indoor scenarios with a wide variety of text appearance in terms of font, thickness, color, size, texture, lighting conditions, points of view and occlusions.

The metrics to evaluate the performance of the text location are defined in [29]. They are the precision and recall together with the f metric, which is used to combine both precision and recall into one single measure.

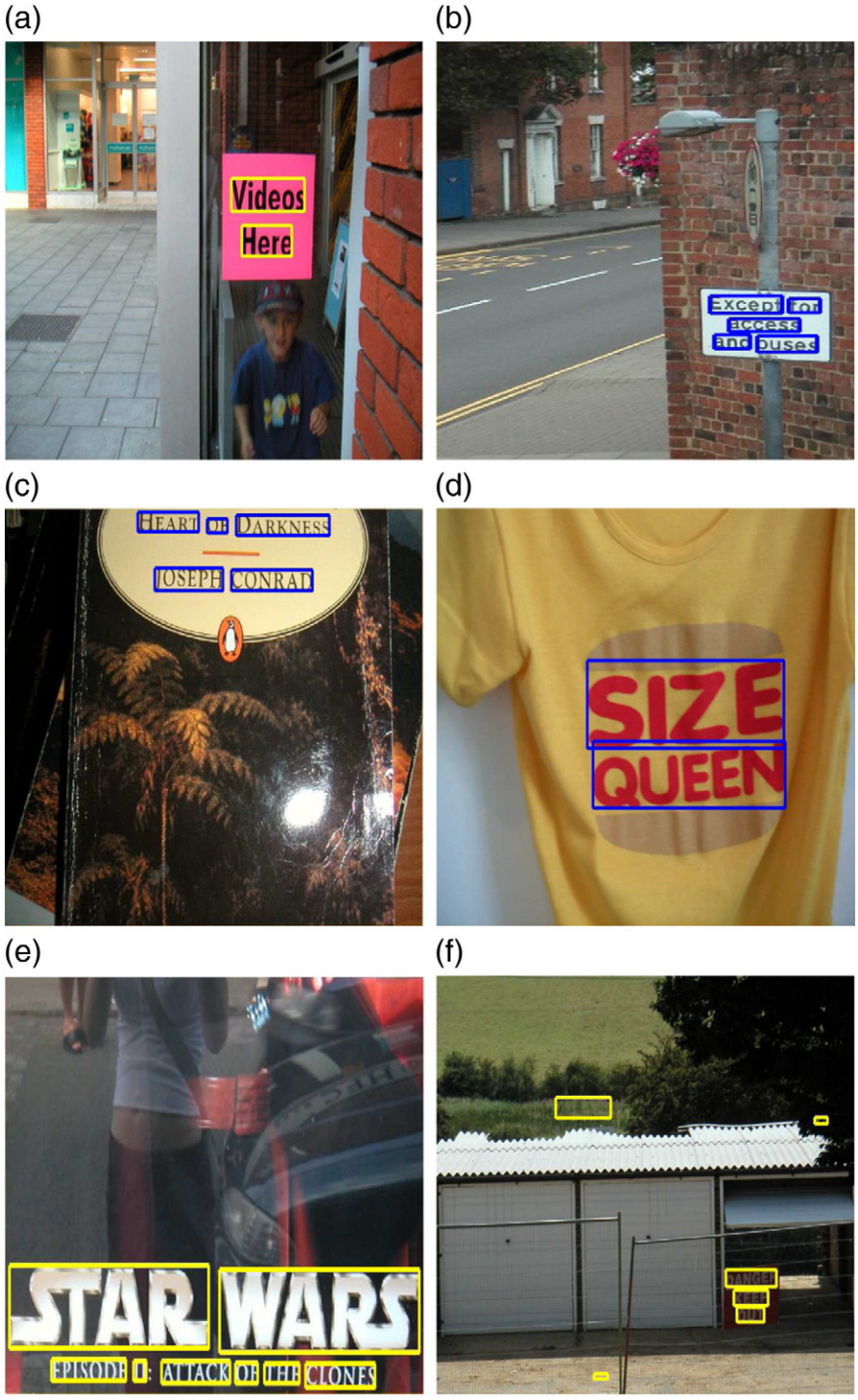

Fig. 19. Text location results on some images from the ICDAR 2011 Challenge 2 dataset.

Table 5
Comparison of segmentation algorithms.

Algorithm	Precision	Recall	f
MSER [3]	0.85	0.53	0.65
Yao et al. [4]	0.67	0.57	0.62
MSER and (Yao et al.)	0.81	0.57	0.67

On the other hand, to evaluate the word recognition accuracy, the percentage of correctly recognized words and the edit distance with equal cost of deletions, insertions and substitutions are used. The edit distance is normalized by the number of characters of the ground truth word and sum over the dataset, as shown in Eq. (25), where $d(\tilde{w}, w)$ represents the edit distance between the output word $\tilde{w}$ and the ground truth word w, and L is the number of words in the dataset.

$$ED = \sum_{i=1}^{L} \frac{d(\tilde{w}, w)}{length(d)} \tag{25}$$

A new competition has been recently held at ICDAR 2011. In this case, the competition was composed of two different challenges. The first challenge deals with born-digital images (web, email), while the second one deals with natural images. The first challenge [31] is organized over three tasks: text location, text segmentation and word recognition. In this paper, we will show our results only for the first and third tasks. The train set for this challenge is composed of 420 images containing 3583 words, while the test set has 102 images containing 918 words. On the other hand, the second challenge [32] consists of two tasks: text location and word recognition. We will show our results for both tasks. The trainset is composed of 229 images and 848 words, whereas the test set has 255 images and 716 words. Most of the images for this challenge have been taken for the ICDAR 2003 Robust Reading competition.

The text location task is evaluated using the methodology proposed by Wolf and Jolion [33], who have developed a software to evaluate object detection algorithms.[1] This method improves the evaluation approach used in ICDAR 2003 competition with the precision and recall measures, which simply compute the overlap area of the bounding box of the detected object and the bounding box of the ground truth object, thus having several drawbacks and can be unfair when evaluating under and over segmentation.

Again, the word recognition task is evaluated using the percentage of correctly recognized words and the normalized edit distance.

7.1. Results on text location

Table 2 shows the performance of our algorithm on the ICDAR 2003/2005 dataset, together with the performance of the winners of the competitions at ICDAR 2003 and ICDAR 2005 and some other methods that have used this dataset as a benchmark in the last decade. It can be seen that we score second in the global ranking, although we outperform the results obtained in the frame of ICDAR 2003 and 2005 competitions. Actually, our approach scores first in terms of precision. It means that the number of false positives that our algorithm produces is the smallest one. Some output samples are shown in Fig. 17.

On the other hand, Tables 3 and 4 show the comparison of our proposed method with the participants in Challenge 1 and Challenge 2 at ICDAR 2011. Some output samples of our method for both challenges are shown in Figs. 18 and 19, respectively. It can be seen that our method scores first in Challenge 1 and second in Challenge 2.

As it was explained in Section 5, the segmentation method is based on a combination of MSER and Yao's algorithm. Table 5 shows the improvement of using both methods instead of using only MSER and only Yao's method. These results have been obtained on the ICDAR 2003 test set.

We have used the Vedaldi's implementation for the MSER algorithm [36]. The MSER algorithm has several parameters that can be

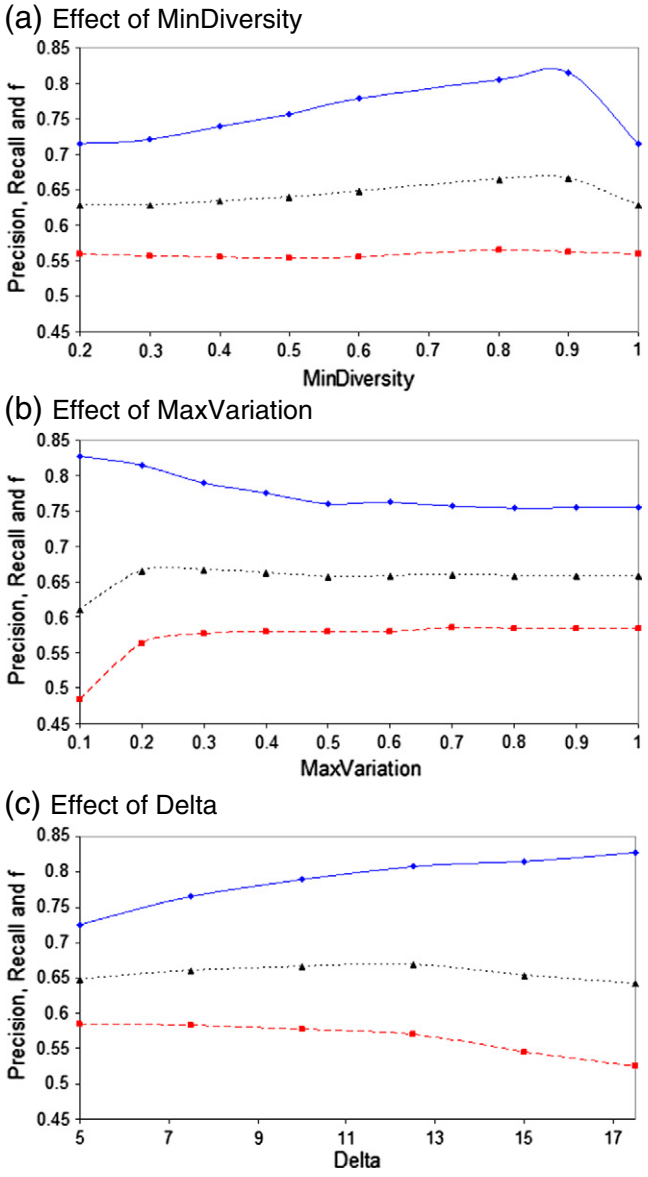

Fig. 20. Effect of varying the MSER parameters on p (continuous), r (dashed) and f (dotted).

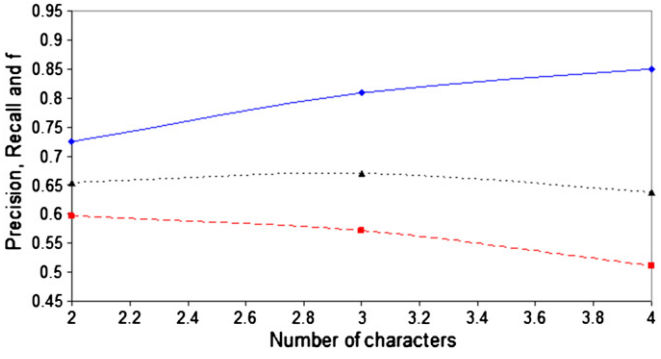

Fig. 21. Effect of varying the minimum number of characters per text line on p (continuous), r (dashed) and f (dotted).

[1] http://liris.cnrs.fr/christian.wolf/software/deteval/index.html.

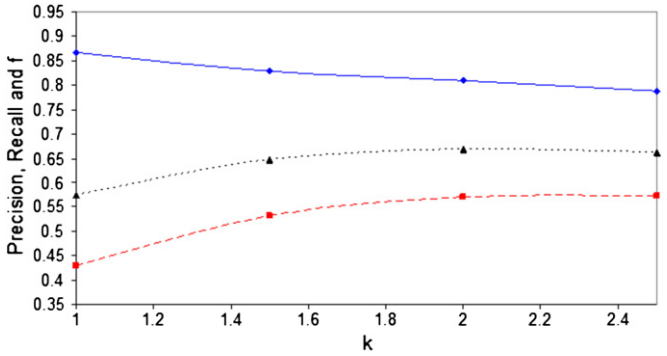

Fig. 22. Effect of varying the ranges of the features $(\mu-k\cdot\sigma, \mu+k\cdot\sigma)$ on p (continuous), r (dashed) and f (dotted).

modified. The most significant are: Delta (it defines the stability of a region, which is the relative variation of the region area when the intensity is changed ±Delta/2), MinDiversity (when the relative area variation of two nested regions is below this threshold, only the most stable one is selected) and MaxVariation (a threshold that sets the maximum variation of the regions). Fig. 20 shows how varying these parameters affects the results. The optimum results are achieved with the values Delta = 12.5, MinDiversity = 0.9 and MaxVariation = 0.3.

In Section 5, it was stated that those lines with less than 3 characters are rejected. This threshold has been found to be optimum, as it is shown in Fig. 21. It can be seen that the less this threshold is, the higher the recall is, but the precision and the f-measure are optimum for 3 characters.

It was also explained in Section 5 that the text features can be approached to Gaussian distributions and that we are using the range $(\mu-2\cdot\sigma, \mu+2\cdot\sigma)$ to discard between text and non-text objects. Fig. 22 shows how varying this range affects to the results. It is seen that we get the optimum results for $k=2$ in $(\mu-k\cdot\sigma, \mu+k\cdot\sigma)$.

The use of the range $(\mu-2\cdot\sigma, \mu+2\cdot\sigma)$ for the text features is optimum, thus a high number of true letters is accepted by keeping low the number of false positives. However, some letter candidates can be erroneously rejected, thus leading to a partial text detection. In order to bring back these mistakenly removed characters, a method to restore them is applied. Table 6 shows its importance.

Finally, Table 7 shows the advantage of using a text line classifier against not using it. If a classifier is not used, the recall is slightly higher but the precision is much lower compared to the case when a classifier is used. It means that the classifier discards many false positives and a small number of true positives. The f measure turns out to be higher when a classifier is used.

7.2. Results on text recognition

7.2.1. Character recognition

In order to evaluate the best features to achieve an optimum character recognizer, the main local features of the state of the art are evaluated:

- Shape Context (SC) [15].
- Geometric Blur (GB) [16].
- Scale Invariant Feature Transform (SIFT) [17].
- Speeded Up Robust Feature (SURF) [37] and Gauge-SURF (G-SURF) [38].

Table 6
Effect of using the character restoration.

Algorithm	Precision	Recall	f
No restoration	0.69	0.33	0.45
Restoration	0.81	0.57	0.67

Table 7
Comparison of text line classifiers.

Classifier	Precision	Recall	f
Without classifier	0.71	0.59	0.65
With classifier	0.81	0.57	0.67

Table 8
Individual character recognition on ICDAR 2003 dataset.

Features	Hit rate 1st candidate	Hit rate 1st/2nd candidate	Hit rate 1st/2nd/3rd candidate
DH	**76.3%**	**91.4%**	**95.6%**
LBP	67.5%	82.7%	90.0%
SC	59.6%	77.0%	83.4%
SIFT	58.9%	66.8%	68.4%
GB	56.1%	70.1%	75.4%
G-SURF	53.1%	64.5%	70.4%
SURF	52.2%	64.0%	70.2%
HOG	48.8%	66.8%	75.4%

- Histogram of Oriented Gradients (HOG) [9].
- Local Binary Patterns (LBP) [39].

Table 8 shows the character recognition rate using each kind of feature. Three cases have been analyzed. The first one only takes into account the hit rate for the output class with highest probability. The second analysis computes the hit rate for those cases in which the recognition succeeds for either the first or the second solution. Similarly it is done for the first, second and third candidates. It can be clearly seen that DH is the best feature and this method successfully recognizes more than 90% of characters as first or second solution. On the other hand, Fig. 23 shows the character recognition rate as a function of the training dataset size. It can be seen that the hit rate for DH feature tends to an asymptote for a training dataset size of 2000 samples, while the asymptote for other features is reached for a major number of samples.

The proposed method has been evaluated on the ICDAR 2003 test dataset, which contains more than 5000 letters in 250 pictures. We compare our approach to the Neumann and Matas' method [5], which was tested with the same dataset. Their method is based on a chain-code bitmap that codes the orientation of the boundary pixels of each binarized object. Table 9 shows the comparison of our method to Neumann's technique. Since Table 8 does not take into account

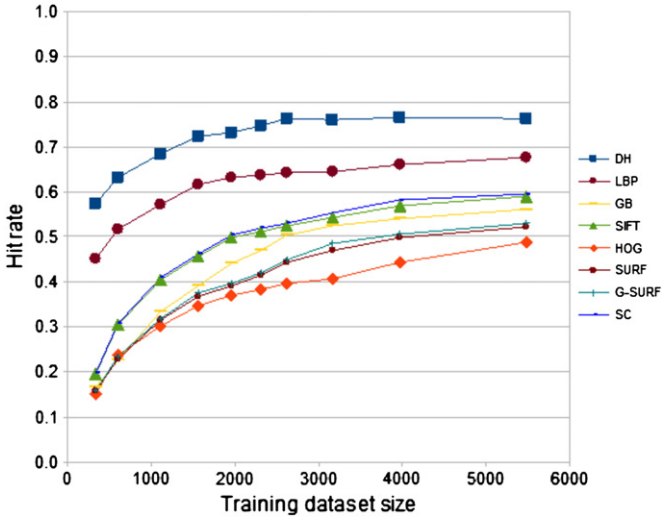

Fig. 23. Recognition rate vs training dataset size (1st candidate).

Table 9
Individual character recognition on ICDAR 2003 dataset.

Algorithm	Matched	Mismatched	Not found
Neumann and Matas [5]	67.0%	12.9%	20.1%
Our method (1st candidate)	68.2%	21.2%	10.6%
Our method (1st/2nd candidate)	81.7%	7.7%	10.6%
Our method (1st/2nd/3rd candidate)	85.4%	4.0%	10.6%

Table 10
Individual character recognition on ICDAR 2003 dataset, taking indistinguishable pairs of letters as one class for each pair.

Algorithm	Matched	Mismatched	Not found
Our method (1st candidate)	80.4%	9.0%	10.6%
Our method (1st/2nd candidate)	84.1%	5.3%	10.6%
Our method (1st/2nd/3rd candidate)	85.8%	3.6%	10.6%

the number of non-detected objects, we have incorporated the non-detection rate in Table 9 in order to make a fair comparison. It can be seen that we get a similar performance to the Neumann's method, even slightly better in terms of hit rate, but we get a really good performance if we take into account the second candidate for this analysis. The mismatched rate for the first two candidates is reduced almost to one third of the mismatched rate with only one candidate and it is much lower than the Neumann's mismatched percentage. Actually, it has been observed that there is a set of pairs and threes of letters that cannot be differentiated between upper-case and lower-case: {'Cc', 'Iil', 'Jj', 'Oo', 'Pp', 'Ss', 'Uu', 'Vv', 'Ww', 'Xx', 'Zz'}. The only way to distinguish these letters in their upper-case and lower-case variants is to use as reference the height of the other unambiguous letters in the same line. In principle, we are just interested in character recognition in a raw way, but if we compute the character recognition rate joining both classes of the undistinguishable letters as only one class for each pair, we get the results shown in Table 10. It can be clearly noticed that the hit rate for the first candidate greatly increases, as it achieves a matched rate higher than 80% and the mismatched rate reduces to 9%.

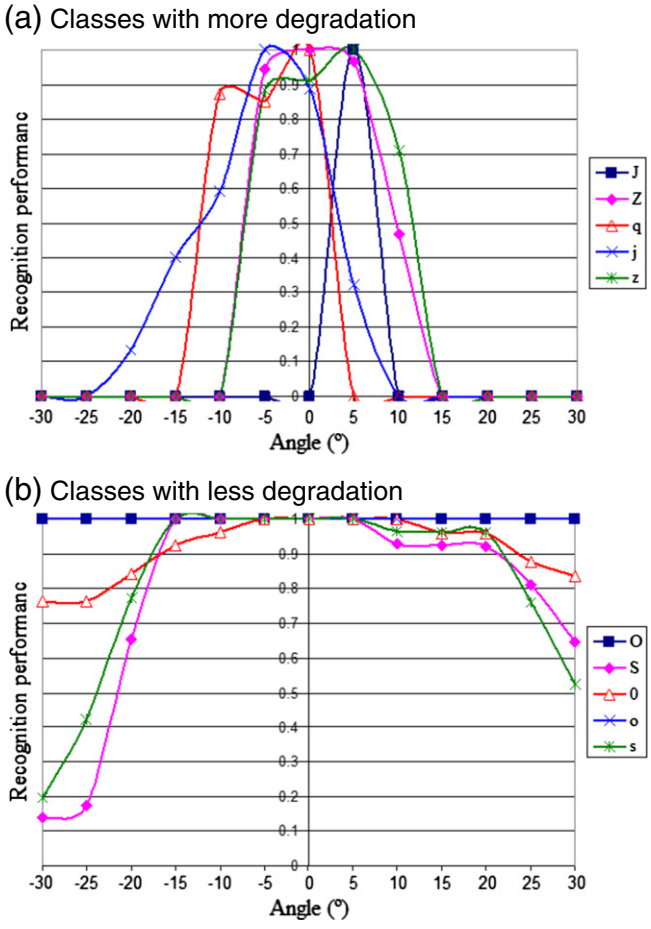

Fig. 24. Effect of rotations on character recognition.

Table 11
Text recognition on ICDAR '03.

Algorithm	Correct recognition (%)	Total edit distance
Our system	**47.43**	**616.87**

The approach proposed in this paper aims at detecting horizontal text, as it is the most common layout when using English and other western languages. However, our approach is able to detect text even when slight deviations and rotations with respect to the horizontal axis take place. In terms of text detection, it has been seen that the proposed approach is able to correctly detect words that have up to 30° of deviation with respect to the horizontal axis. However, in terms of character recognition, the performance depends on the class (letter or number). A study on how rotations affect the accuracy of the character recognizer is presented below. In general, the error committed when recognizing letters increases as the rotation grows up. However, the degradation is bigger for some letters. Specifically, among the 62 classes (52 letters and 10 digits), those that suffer a faster degradation are the following: 'J', 'Z', 'q', 'j' and 'z'. As it is shown in Fig. 24(a), the accuracy of the recognition for these letters falls below 30% from 5° or 10° upwards. On the other hand, those classes that hardly suffer degradation are 'O', 'S', '0', 'o' and 's'. The recognition accuracy remains above 80% even when the rotation angle reaches 25° or 30°, as shown in Fig. 24(b).

7.2.2. Word recognition

Table 11 shows the recognition rate achieved for the ICDAR 2003 word recognition task. There were no participants in this contest in the 2003 and 2005 editions. From our knowledge, no one has used this dataset as benchmark in terms of word recognition in the last decade. Therefore, we cannot compare our results to any other methods. On the other hand, Tables 12 and 13 show the performance of our algorithm using the datasets released for both challenges in the ICDAR 2011 competition. In this case, there were several entries. We compare our algorithm to them. It can be clearly seen that we score first in both challenges in terms of correct recognition rate. However, we obtain a higher value for the total normalized edit distance. This is due to the fact that we are applying a word recognizer based on

Table 12
Text recognition on ICDAR '11 Chall. 1.

Algorithm	Correct recognition (%)	Total edit distance
Our system	**66.88**	**226.8**
Baseline	63.94	231.2
TH-OCR	61.98	189.1

Table 13
Text recognition on ICDAR '11 Chall. 2.

Algorithm	Correct recognition (%)	Total edit distance
Our system	**46.9**	**639.15**
TH-OCR	41.2	176.23
KAIST AIPR	35.6	318.46
Neumann's method	33.11	429.75

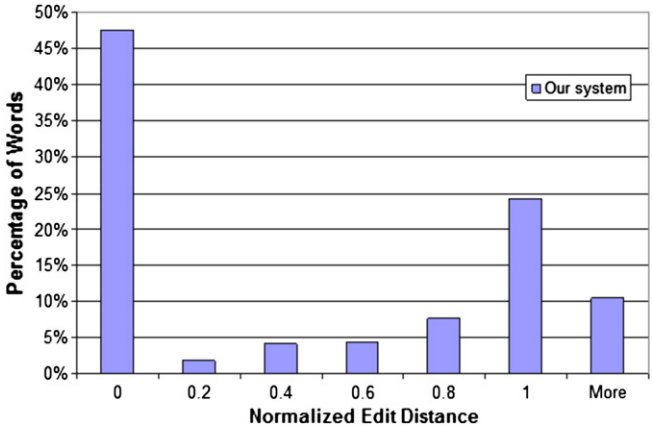

Fig. 25. Histogram of normalized edit distances for ICDAR 2003.

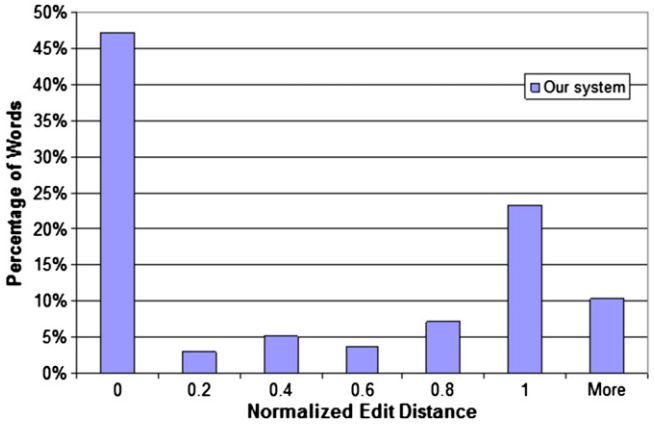

Fig. 27. Histogram of normalized edit distances for ICDAR 2011 Challenge 2.

probabilistic inference, while other works do not. This can lead to give outputs which are completely different to the input, i.e. they have a high normalized edit distance, close to 1 or even above 1, while other methods do not but the word recognition rate is much lower. This effect can be seen in Figs. 25–27, which show the histogram of normalized edit distances. It is notable that the percentage of cases with a normalized edit distance equal or higher than 1 (corresponding to all characters being changed), is higher for the proposed method than for other methods. However, we firmly think that the best way to assess the performance of a word recognition algorithm is to measure the number of correctly recognized words, as the final task is to recognize single words, as most as possible.

8. Conclusions and future work

We have presented a method to localize and recognize text in natural images. The text location is a CC-based approach that extracts and discards basic letter candidates using a series of easy and fast-to-compute features. These features have been analyzed on a challenging train dataset which contains different types of text in a huge variety of situations and they have proved to follow a Gaussian distribution. It means that they can be used with any dataset independently from their size, color or font. Actually, the proposed method has been tested on several test benchmarks and the achieved results show the competetiveness of the method. Unlike other methods, a strong point is the use of feedback in order to restore those characters that might have been erroneously filtered out after computing the text features for each letter candidate. It has been also proposed to use a classifier based on simple features such as mean, standard deviation and gradient directions computed over image blocks in order to remove certain structures that can be easily confused with text lines, such as bricks or fences.

On the other hand, the text recognition is based on a simple and fast-to-compute feature, which has been baptized as Direction Histogram, and a KNN-based approach to recognize single characters. In addition, our method gives several solutions with single output probabilities. An application to separate erroneously detected characters has been showed here, but there could be many other applications. Finally, in order to correct the noise produced in the character recognition stage, a probabilistic inference method based on DP has been proposed. The experimental results obtained on several challenging datasets show that we achieve and exceed in some cases state-of-the-art performance.

As future work, we are interested in classifying the images into different categories from the text extracted in the image itself. The applications of this could be several, from image spam filtering to document classification for those records which are totally or mostly composed by images.

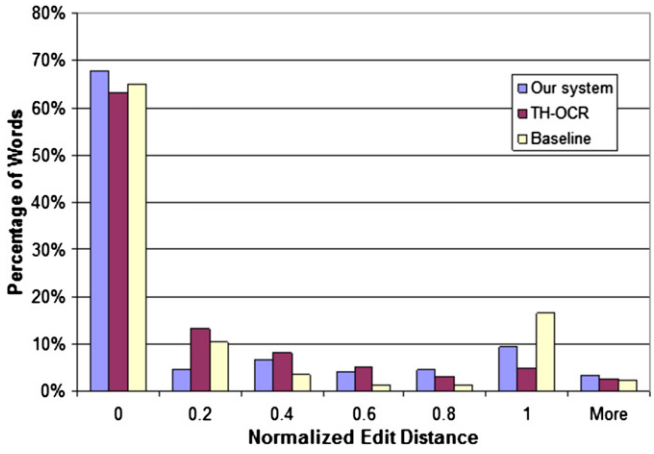

Fig. 26. Histogram of normalized edit distances for ICDAR 2011 Challenge 1.

Acknowledgments

This work has been financed with funds from the Ministerio de Economía y Competitividad through the project ADD-Gaze (TRA2011-29001-C04-01), as well as from the Comunidad de Madrid through the project Robocity2030 (CAM-S-0505/DPI000176).

References

[1] S. Escalera, X. Baró, J. Vitrià, P. Radeva, Text detection in urban scenes, CCIA, 2009, pp. 35–44.
[2] D. Karatzas, A. Antonacopoulos, Colour text segmentation in web images based on human perception, Image Vision Comput. 25 (2007) 564–577.
[3] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions, BMVC, 2002.
[4] J.-L. Yao, Y.-Q. Wang, L.-B. Weng, Y.-P. Yang, Locating text based on connected component and svm, in: Wavelet Analysis and Pattern Recognition, 2007. ICWAPR '07. International Conference on, volume 3, pp. 1418 –1423
[5] L. Neumann, J. Matas, Real-time scene text localization and recognition, Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
[6] B. Epshtein, E. Ofek, Y. Wexler, Detecting text in natural scenes with stroke width transform, CVPR, IEEE, 2010, pp. 2963–2970.
[7] H. Chen, S.S. Tsai, G. Schroth, D.M. Chen, R. Grzeszczuk, B. Girod, Robust text detection in natural images with edge-enhanced maximally stable extremal regions, ICIP, 2011.
[8] Y.-F. Pan, X. Hou, C.-L. Liu, A hybrid approach to detect and localize texts in natural scene images, IEEE Trans. Image Process. 20 (2011) 800–813.
[9] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, CVPR, 1, 2005, pp. 886–893.
[10] J. Sochman, J. Matas, WaldBoost — learning for time constrained sequential detection, CVPR, 2, 2005, pp. 150–156.

[11] W. Niblack, An introduction to digital image processing, Prentice-Hall, 1986.
[12] J.D. Lafferty, A. McCallum, F.C.N. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, ICML, 2001, pp. 282–289.
[13] X. Chen, A.L. Yuille, Detecting and reading text in natural scenes, CVPR, 2, 2004, pp. 366–373.
[14] S.M. Hanif, L. Prevost, Text detection and localization in complex scene images using constrained adaboost algorithm, ICDAR, 2009, pp. 1–5.
[15] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 509–522.
[16] A.C. Berg, T.L. Berg, J. Malik, Shape matching and object recognition using low distortion correspondences, CVPR, 1, 2005, pp. 26–33.
[17] D.G. Lowe, Object recognition from local scale-invariant features, ICCV, 1999, pp. 1150–1157.
[18] S. Lazebnik, C. Schmid, J. Ponce, A sparse texture representation using local affine regions, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1265–1278.
[19] M. Varma, A. Zisserman, Classifying images of materials: achieving viewpoint and illumination independence, ECCV, 3, 2002, pp. 255–271.
[20] M. Varma, A. Zisserman, Texture classification: are filter banks necessary? CVPR, 2, 2003, pp. 691–698.
[21] T.E. de Campos, B.R. Babu, M. Varma, Character recognition in natural images, VISAPP, 2, 2009, pp. 273–280.
[22] L. Neumann, J. Matas, A method for text localization and recognition in real-world images, ACCV, 3, 2010, pp. 770–783.
[23] Y. Zhu, J. Sun, S. Naoi, Recognizing natural scene characters by convolutional neural network and bimodal image enhancement, in: M. Iwamura, F. Shafait (Eds.), Camera-Based Document Analysis and Recognition, volume 7139 of *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, 2012, pp. 69–82.
[24] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D.J. Wu, A.Y. Ng, Text detection and character recognition in scene images with unsupervised feature learning, ICDAR, 2011, pp. 440–445.
[25] A.J. Newell, L.D. Griffin, Natural image character recognition using oriented basic image features, DICTA, 2011, pp. 191–196.
[26] W.-L. Chan, C.-M. Pun, Robust character recognition using connected-component extraction, IIH-MSP, 2011, pp. 310–313.
[27] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young, K. Ashida, H. Nagai, M. Okamoto, H. Yamamoto, H. Miyao, J. Zhu, W. Ou, C. Wolf, J.-M. Jolion, L. Todoran, M. Worring, X. Lin, ICDAR 2003 robust reading competitions: entries, results, and future directions, IJDAR 7 (2005) 105–122.
[28] British national corpus, http://www.natcorp.ox.ac.uk/ 2007.
[29] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young, Robust reading competitions, ICDAR, 2003.
[30] S.M. Lucas, Text locating competition results, ICDAR, 2005, pp. 80–85.
[31] D. Karatzas, S.R. Mestre, J. Mas, F. Nourbakhsh, P.P. Roy, ICDAR 2011 robust reading competition — challenge 1: reading text in born-digital images (web and email), ICDAR, 2011, pp. 1485–1490.
[32] A. Shahab, F. Shafait, A. Dengel, ICDAR 2011 robust reading competition challenge 2: reading text in scene images, ICDAR, 2011, pp. 1491–1496.
[33] C. Wolf, J.-M. Jolion, Object count/area graphs for the evaluation of object detection and segmentation algorithms, Int. J. Doc. Anal. Recognit. 8 (2006) 280–296.
[34] S.H. Lee, M.S. Cho, K. Jung, J.H. Kim, Scene text extraction with edge constraint and text collinearity, ICPR, 2010.
[35] J. Zhang, R. Kasturi, Text detection using edge gradient and graph spectrum, ICPR, 2010.
[36] A. Vedaldi, B. Fulkerson, VLFeat: an open and portable library of computer vision algorithms, http://www.vlfeat.org/ 2008.
[37] H. Bay, A. Ess, T. Tuytelaars, L.J.V. Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (2008) 346–359.
[38] P.F. Alcantarilla, Vision based localization: from humanoid robots to visually impaired people, Ph.D. thesis, University of Alcalá, Alcalá de Henares, Madrid, Spain, 2011.
[39] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 971–987.