International Journal on Document Analysis and Recognition manuscript No.
(will be inserted by the editor)

TextCatcher: a Method to Detect Curved and Challenging Text in Natural Scenes

Jonathan Fabrizio · Myriam Robert-Seidowsky · Séverine Dubuisson · Stefania Calarasanu · Raphaël Boissel

Received: date / Accepted: date

Abstract In this paper, we propose a text detection algorithm which is hybrid and multi-scale. First, it relies on a connected component-based approach: after the segmentation of the image, a classification step using a new wavelet descriptor spots the letters. A new graph modeling and its traversal procedure allow to form candidate text areas. Secondly, a texture-based approach discards the false positives. Finally, the detected text areas are precisely cut out and a new binarization step is introduced. The main advantage of our method is that few assumptions are put forward. Thus, "challenging texts" like multi-sized, multi-colored, multi-oriented or curved text can be localized. The efficiency of *TextCatcher* has been validated on three different datasets. Two come from the ICDAR competition and the third one contains photos we have taken with various *daily life* texts. We present both qualitative and quantitative results.

Keywords Text detection · Toggle Mapping Morphological Segmentation · Text cut-out · Wavelet transform · Letter extraction and recognition · Natural scene

1 Introduction

In recent years, there has been a significant increase of research works on text detection in images and video sequences. Methods to localize scene texts are often dedicated to specific applications such as vehicle license plate recognition [3], form reading, text recognition to help visually impaired people [53], translation of street signs, reading of engineering drawings/maps, driving help, image and video indexing, *etc.* In such cases, hypotheses on color, font or size of the characters, or on the background are considered. *Daily life* text spotting, especially in urban areas, is still a challenging and open problem. Due to the variability of texts and cluttered backgrounds, it is impossible to integrate strong hypotheses in text detection algorithms. For example, text strings can be multi-color, curved, vertical and characters do not necessarily have the same size, font or alignment. Furthermore, text may suffer from artefacts such as low contrast, blur, variations of illumination (specular reflections, non-uniform illuminations, shadows), cluttered surrounding background, occlusion, *etc.*

In this paper, we propose a context free system for text localization and extraction. We developed an algorithm that can detect and extract all textual information of natural scenes. Although many works have been proposed (see Section 2 for a review of some recent ones), most of them assume strong hypotheses. On the contrary, our main objective is to limit the number of assumptions to detect as many text as possible in complex configurations. We only make two hypotheses: characters belong to the Latin alphabet and we do not deal with cursive text.

Compared to our previous work in [10] all steps have been improved except the segmentation step:

– we introduce a new shape descriptor to separate text and non text regions (notice that, this shape descriptor is a good candidate to perform, in the future, the transcription of the detected text - this

J. Fabrizio · M. Robert-Seidowsky · R. Boissel · S. Calarasanu
E-mail: jonathan/myriam/boissel/calarasanu@lrde.epita.fr
EPITA Research and Development Laboratory (LRDE)
14-16, rue Voltaire, F-94276, Le Kremlin Bicêtre, France

S. Dubuisson E-mail: severine.dubuisson@isir.upmc.fr
Sorbonne Universités, UPMC Univ Paris 06 CNRS, UMR 7222, ISIR F-75005, Paris, France

IJDAR 2016 - Springer
http://dx.doi.org/10.1007/s10032-016-0264-4

means that the transcription would be provided without additional cost),
- the grouping of candidate characters is now done by using a new connexity graph of and with an efficient graph traversal algorithm. We can now detect text in every direction, slanted and curved,
- the validation of the text candidate has been revisited to be improved. As we make only a few hypothesis on the detected text, the number of false positives can increase. Then the validation part must be very efficient,
- text is precisely spotted in the image with a simple strategy (common strategies are not precise enough when dealing with rounded or tilted text),
- the binarization of detected text is now performed by a new algorithm, able to aggregate results from different scales to give a unique binarization (it can also merge the results of detection from different color channels if one want to treat each color plane separately),

Compared to the state of the art, our method can detect:

- text slanted in an arbitrary direction or curved text,
- text containing characters without any restriction on the color or on orientation.

Methods in the literature that can deal with curved text are very uncommon. Most of the state of the art algorithms are restricted to single-colored and horizontal texts. It is difficult to relax these constraints to generalize existing methods without increasing the number of false positives. On the contrary, our method is generic and due to its modular structure, it is easy to specialize some steps to get a detector dedicated to a specific context. Our precise cutting-out of detected text is also very useful for automatic text enhancement or text blurring.

The paper is organized as follows. Section 2 provides a brief survey of relevant recent works on text localization and Section 3 gives an overview of our proposed system for text localization. Following sections describe each step of the localization process. Section 4 first exposes the segmentation of the input image into a set of connected components (CCs). Section 5 details the classification process of the CCs into letter or non-letter. The grouping stage of letter regions and the mask computation which cut out each text string are depicted in Section 6. Section 7 describes our text string validation mechanism. In the last step, text is extracted using a multi-scale binarization process detailed in Section 8. Finally, the performances of our algorithm are eveluated in Section 9. Concluding remarks and perspectives are given in Section 10.

2 State-of-the-art

Many methods to localize text in natural images have already been proposed. Usually, in the literature, text localization systems are divided into two main categories depending on the features they are working on: texture-based approaches and connected component based approaches. Some of them use both and are then considered as hybrid. A typical algorithm is divided into three main steps:

1. The important information is first highlighted (on a binary map for example) using a *segmentation* or a specific *characterization* process, depending on the features (texture or connected components) the approach works on.
2. A *local classification* step separates letter and non-letter regions.
3. A *grouping* is finally often necessary to fuse different letter regions and form complete words.

Note that sometimes a global classification may be necessary after the grouping. This is often referred as a *validation* step and allows to consider global information to form words, or, more generally, text strings.

Some surveys on text localization have already been done [18,43,56]. In this section, we chose to focus on some recent algorithms (published since 2010) on Latin text detection, even if some interesting works dealing with multi-script text have also been proposed [13,23].

Even if the framework of most of the algorithms is common, multiple strategies have been explored for each individual step.

For CC-based approaches (and for hybrid methods integrating a first step of analysis of CCs), the segmentation step is of crucial importance. For that, the Stroke Width Transform (SWT) introduced by Epshtein [8], has been widely used [52,19]. The Maximally Stable Extremal Regions (MSER) introduced by Matas et al. [27] has also been widely used [25,29,50,36]. Both provide interesting results. The mathematical morphology field has been investigated with the Toggle Mapping Morphological Segmentation (TMMS) [10]. Classical binarization methods, like Niblack thresholding, have also been tested [11,28]. Then, to analyze the segmented regions, different classifiers and descriptors have been explored: Support Vector Machines (SVM) with Histogram of Oriented Gradients (HOG), geometric features or shape descriptors [24,48,10,28], Adaboost [36, 50,11], Random Forests [20,52] and k-Nearest-Neighbors (k-NN) [19]. Most of these approaches can only deal with horizontal text zones, or almost horizontal ones [31]. Note that the method proposed in [42] is the only one that can manage curved text by using a Quadtree-based technique.

For texture-based approaches (and for hybrid methods with a characterization by texture), the texture analysis is performed in different spaces after a data transformation: Discrete Cosine Transform (DCT) [38], HOG [41,46,4,10,11], Discrete Shearlet Transform (DST) [55], SIFT [26], edges analysis [4,37,46,53,11] and Gradient Vector Flow (GVF) [37] have been investigated.

As specified before, some methods mix CC-based strategies and texture-based strategies [2,10,11,28,36].

Recently, deep learning based algorithms have emerged and gave very promising results [47,16,15]. Their main advantage is they can learn interesting features on very large datasets. However, they are still limited to horizontal texts.

We put forward a graphical presentation in Tables 1, 2 and 3 that summarizes, for each method, how information is highlighted (features are in italic blue and tools are in bold red), classified (letter *versus* non-letter) and grouped (letters into text strings). These tables also give, for each algorithm, its advantages (*i.e.* the characteristics it can deal with) and its limitations.

Note that we do not mention in this table the scores given by these methods. Indeed, these scores are not comparable because they do not use the same test sets and, sometimes, have been obtained with different evaluation protocols. Moreover, only few of them give the average computation time and explain how the algorithm performs in the worst case (images with a lot of textures for example).

As shown in the two last columns of Tables 1, 2 and 3, recent methods mainly focus on few difficulties concerning the text characteristics. In this article, we propose a method called *TextCatcher* which is dedicated to *daily life* text detection because it can manage:

- multidirectional (horizontal, vertical, tilted), perspective and curved texts,
- texts with letters in different orientations or colors,
- textured texts and texts on cluttered background,
- texts with various size and fonts.

Especially, among all listed methods, only one is able to manage curved text. Moreover, our algorithm provides an accurate detection, thanks to an original binarization scheme. To the best of our knowledge, our approach is the only one which is able to handle so many difficulties by relaxing simultaneously lots of hypotheses on texts to detect. In the next section, we give a brief description of our proposed approach.

3 Overview of the Method

The work presented in this article is an extension of the one proposed in [10] and describes a full system for efficient text localization in natural images. Even if the main goal is to localize texts in images, we also propose a binarization method that permits to extract each letter of the text region. In this extended version, although the global processing chain remains the same, each of its steps has been improved by using more efficient and original techniques. Figure 2 shows the global flowchart of our method, from the multi-scale pyramid of images to the extraction of the letters. Each stage of the chain is briefly summarized in this section, and fully described in the next ones. Our method is hybrid because it uses connected components (CC) to generate text candidates and texture features to validate or discard these candidates.

The first stage consists in localizing potential zones by using a CC-based approach. For that, the image is first segmented using the Toggle Mapping Morphological Segmentation (TMMS) algorithm (see Section 4) whose description is given in [9]. Then, we use two different classification steps to tag each CC as letter or non-letter. The first method uses common and fast geometric criteria such as size or aspect ratio of CCs with fixed thresholds (see Section 5.1). The second one relies on machine learning (see Section 5.2). We introduce a new shape descriptor based on a fast wavelet decomposition. The combination of this shape descriptor and a machine learning approach (based on a k-Nearest-Neighbours algorithm) allows, not only to get a label for each CC (letter or non-letter), but also, without any additional cost, to identify these letters. In the last step of this localization stage, CCs are grouped along multiple possible directions using a graph representation (see Section 6).

After this localization stage, the second stage based on a texture analysis permits to validate or discard the previous grouped CCs. Since the first stage is local to each CC, we now consider global information, *i.e.*, a group of letters is supposed to form a text string. For that, we consider different kinds of features: Gray Level Co-occurence Matrix, Haralick texture features [14], Histogram of Oriented Gradients and Local Binary Patterns [34]. A Support Vector Machine (see Section 7) is used to validate grouped CCs as text strings or to discard false positives (non-text strings).

These two previous stages, candidate generation (including segmentation, local classification and grouping steps) and candidate validation, are applied at different scales of the input image. Between two scales, the size of the image (width and height) is reduced by a factor of 2. This is repeated until the width or the height of the image becomes smaller than a given threshold (150 pixels). This allows to detect letters with different sizes in images. At each scale, the process is performed on

Table 1 Some texture-based works proposed since 2010 (DST: Discrete Shearlet Transform, HVS: Human Visual System, TSM: Tree-Structured model, NMS: Non Maximum Suppression, SW: Stroke Width, SWT: Stroke Width Transform), CNN: Convolutional Neural Network, RSLA: Run Length Smoothing Algorithm. Tools are in bold red and features in italic blue.

		Segmentation / Characterization	Local classification	Grouping	Advantages	Limitations
Texture	Bai [4]	*Edge density* *SW consistency*	**SVM** *HOG*	Color, SW Distance Direction	Any size Complex backgrounds	Many thresholds Contrasted text No curved text
	Mao [26]	**Neural Network** **Region growing** *SIFT*	**Thresholding**	Distances Color Area Hough transform	Any orientation Any font Complex background	Computation time No curved text
	Phan [37]	**Gradient Vector Flow** *Edges*	**Clustering** *Symmetry components*	Size Color Stroke thickness Gap	Any font Complex backgrounds	Bounded text size Horizontal text
	Prakash [38]	*DCT* *AC component*	*SWT* **Dilation** **Heuristics**	Aspect Size	Any orientation Any font No tuning	Bounded text size No curved text
	Shi [41]	**Dynamic programming** *HOG* **TSM**	**Optimization** *Spatial constraints*		Detect and recognize	Simple background No curved text
	Wang [46]	*Edges* *HOG*	**Random Ferns** **NMS**	Pictorial structures	Detect and recognize Any font	Only horizontal No curved text
	Yi [53]	*Stroke orientation* *Edge distribution* *Gradient*	**Adaboost** *Block patterns*	Height Alignment Area Overlapping	Complex background Any Illumination Any font	Manual pre localization Horizontal text
	Zagoris [54]	*HVS maps* *Center-surround regions* **ON/OFF cell modeling**	**Thresholding** **Heuristics** *Geometric features*	Size Distance Width/height Hole number Overlapping	Any size	Many thresholds Horizontal text Simple background
	Zhang [55]	*DST* **Thresholding** **Dilation**	**Multi-scale voting**		Any size Complex background	Coarse detected regions No curved text
	Wang [47]	*32x32 patches*	**CNN**	**NMS**	Recognize text	Only horizontal
	Jaderberg [16]	*24x24 patches*	**CNN**	**RLSA**	Recognize text	Only horizontal Simple background
	Huang [15]	**MSER**	**CNN**	**NMS**		Only horizontal

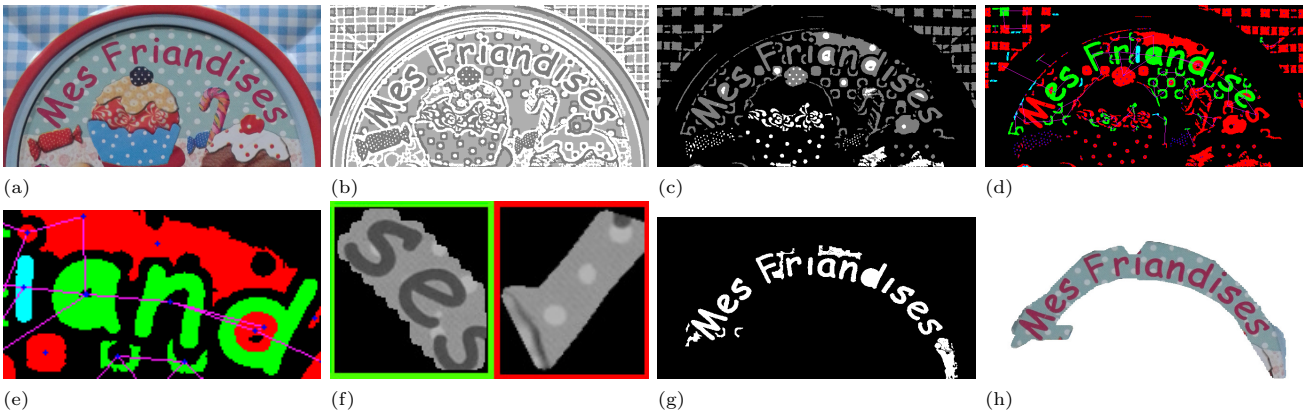


Fig. 1 Steps of our chain at one scale and with one polarity (a) Input image (b) TMMS segmentation (c) Geometric filters (d) Shape analysis (e) Zoom on graph grouping (f) Valid and invalid candidate text areas (g) Binarization (h) Image restricted to the precise text localization result.

each polarity (*i.e.*, the original image and the negative image).

The work proposed in [10] is dedicated to horizontal or tilted mono color text detection. In this paper, all steps, except the segmentation, have been entirely revisited both to be more accurate and deal with curved and challenging texts by making only few hypotheses on texts to detect. The local classification step is totally new. Because we only use one descriptor (based on wavelets), it is easier and faster to compute than the previous classification which used a cascade SVMs with three different descriptors. The grouping step is also new, based on a new graph modeling, that allows to regroup connected components in any direc-

Table 2 Some connected component based works published since 2010 (eMSER: Edge preserving MSER, CBIF: Contrast-based Blur Invariant Features, CSER: Class Specific Extremal Regions). Tools are in bold red and features in italic blue.

		Segmentation / Characterization	Local classification	Grouping	Advantages	Limitations
Connected component	Cong [52]	*SWT* *Edges*	**Tresholding** **Random forest**	SW Size Color	Any orientation Any font Any illumination Complex background	No curved text
	Kan [19]	*SWT* *Edge projection*	**k-NN** *Multi-orientation responses*	Distance Position Any font	Detect and recognize Any size	Only horizontal No curved text
	Karaoglu [20]	**Connected opening** **Gamma correction** **Difference** **Binarization**	**Random forest** *Geometric features* *Shape features* *Corners* *CBIF*	Distance Gap	Any size Any illumination Any font	Horizontal text Simple background
	Karaoglu [21]	*Color saliency* *Curvature saliency* *Contextual priors* **Linear combination**	**Thresholding** *Edges*	Dilation	Few tuning Any illumination Other applications Any orientation	Simple background No curved text
	Li [24]	**Thresholding** *Gradients*	**SVM** *HOG*	Graphcuts Min/max flow	Any size Any illumination	Horizontal text Simple background
	Li [25]	*eMSER* *Surrounding context*	**Graphcuts** **Mincut/maxflow**	Distance	Any size Any font	Horizontal text Simple background
	Merino [29]	*Hierarchical MSER tree*	**Tree pruning** **Region filtering**	Relative position Edge angle Size	Computation times	Horizontal text Simple background Many false positives
	Neumann [33]	*CSER* *Geometric features* **Thresholding**	*Graph* **Dynamic programming**	Pruning Exhaustive search	Any font, size Computation time Complex background	Straight text Short strings
	Shivakumara [42]	*RGB channels* **Max-min clustering**	**k-mean** **Symmetry** *SWT*	Quadtree Spatial relationships Region growing	Any orientation Curved text Computation time	Contrasted text
	Tomer [44]	*Gradient* *Color* **Partitioning**	**Ant colony clustering**	Position Orientation		Simple background No curved text
	Wang [48]	**MRF** **Superpixel segmentation** *Local contrast* *Color* *Gradient*	**SVM** *Separate component channel*	Gap Distance	Any font, size Complex background Any illumination	Contrasted text No curved text
	Xu-Cheng [50]	**Tree pruning** *MSER*	**Adaboost**	**Single-link clustering** Distance, size alignment, color	Any size Any font	No curved text Only horizontal

Table 3 Some hybrid works proposed since 2010 (CRF: Conditional Random Field, TMMS: Toggle Mapping Morphological Segmentation, CSH: Center-Surround Histogram, RLSA: Run Length Smoothing Algorithm, CART: Classification And Regression Trees. LTP: Local Ternary Pattern). Tools are in bold red and features in italic blue.

		Segmentation / Characterization	Local classification	Grouping	Advantages	Limitations
Hybrid	Anthimopoulos [2]	**Random forest** *MACeLBP* *Color* **Region growing**	**Multi-resolution analysis**	RLSA	Complex background Any size Natural/artificial text	Computation times No curved text
	Fabrizio [10]	**SVM** *HOG* **TMMS**	**SVM** *Zernick moments* *Fourier descriptors* *Polar coordinates*	Size Distance	Computation time Any size Complex background	Horizontal text No curved text
	Gao [11]	**Adaboost** *Texture and statistical features (LBP, HOG, SW, edges)* **Niblack binarization**	**Transfer learning** **Adaboost** *CART*	Size Color Occupation SW Overlapping	Context adaptative Any size	Horizontal text Normal illumination Many thresholds
	Meng [28]	*Multi-scale contrast* *Color spatial distribution* *CSH*, *SWT* **CRF** **Niblack binarization**	**SVM** **Heuristics** *Geometric features*	Color SW Width/height		Horizontal text Contrasted text Bounded text size
	Opitz [36]	*MSER*	**Adaboost** *LTP*	Color Height Distance		Horizontal text Normal illumination Simple background

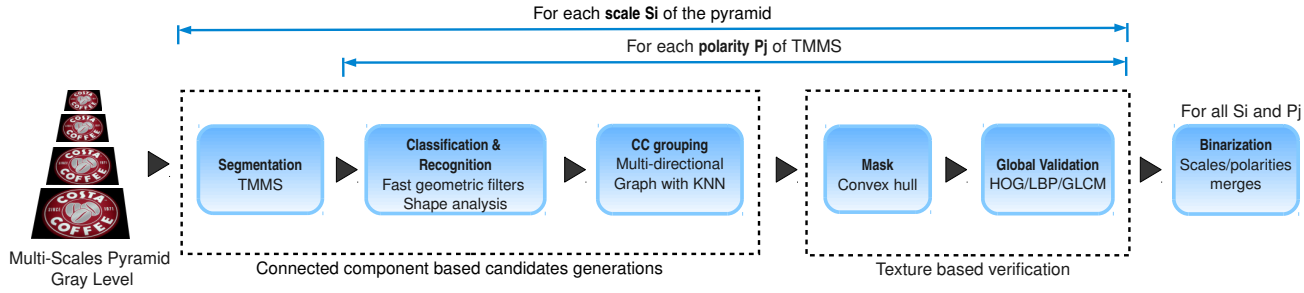


Fig. 2 Overview of our system for text localization and extraction.

tions (diagonal, circular, *etc.*), contrary to most of the recent state-of-the-art approaches. The validation step has been improved in order to deal with heterogeneous texts (perspective views, for example). The text spotting is also much more precise (mask creation, see Figure 1 (h)), which is an originality with regards to the state of the art. This improvement allows to detect and extract curved text for example. Hence, we provide a new binarization scheme (Figure 1 (g)) which is able to merge detected texts at different scales with multi-polarities (integration of results obtained in each color channel). Note that we usually work on grayscale images but it is also possible to consider each color channel separately and merge the results provided by the application of the global chain on each of the color channel. In the rest of the paper, we fully describe each step.

4 Segmentation

The initial segmentation is made using the Toggle Mapping Morphological Segmentation (TMMS) algorithm proposed in [9]. This operator is based on the Toggle Mapping [39], and segments a grayscale image f using a set of two functions h_1 (a morphological erosion of f) and h_2 (a morphological dilation of f).

For that, we define the function s, that corresponds to a segmentation of f, whose value $s(x)$ of any pixel x is given by:

$$s(x) = \begin{cases} a \text{ if } & |h_1(x) - h_2(x)| < k \\ b \text{ if } & |h_1(x) - h_2(x)| \geq k \quad and \\ & |h_1(x) - f(x)| < p \times |h_2(x) - f(x)| \\ c \text{ otherwise} & \end{cases} \quad (1)$$

where k corresponds to a minimal contrast and p is a percentage. The segmentation s is then computed by comparing pixel by pixel the original image and its morphological erosion and dilation. If h_1 and h_2 are too close to each other, then $s(x)$ gets value a. If a pixel value $f(x)$ is $p\%$ closer to its corresponding in h_1 (erosion), then $s(x)$ gets value b. If it is $(100-p)\%$ closer to its corresponding value in h_2 (dilation), $s(x)$ gets value c. In practice, a small homogeneous region gets b or c values depending on its neighborhood (see [9] for more details). To fix the threshold k, which determines if the region is homogeneous, we use a hysteresis thresholding strategy (we then have two thresholds k_{min} and k_{max}). Figure 1 (b) gives an example of the segmentation given by the TMMS approach. This process has shown to be efficient in natural images as well as in document images: it reached the second place among 43 methods at DIBCO 2009 contest [12].

In practice k_{min} is set to 20, k_{max} is set to 45, and p is set to 60. The advantage of the hysteresis strategy is that the whole process is less sensitive to k_{min} and k_{max}. In our case, we consider two polarities: the original image and the negative one. To get the segmentation of the first polarity, we apply the TMMS on the original image by using the p value in Eq. 1. To get the segmentation of the second polarity, we apply the TMMS on the original image by using the $(100-p)$ value. This latter result is identical to the one obtained by computing the TMMS with the p value on the negative image, but faster.

5 Connected Components Classification

After the segmentation of the input image, we get a collection of regions which are connected components (CCs). This list contains a majority of non-letter regions, but also letter regions. The goal of this classification stage is to remove as many as possible the regions that are non-letter (background area for example) from this collection. As in a natural (and then textured) image, the number of extracted CCs may be huge, this classification must be as fast as possible. Note that thanks to the multi-scale strategy and a polar representation, our algorithm deals with different letter sizes and thus becomes more general. Our classification is divided into two main steps:

1. The first step coarsely discards non-letter regions using fast geometric filters (see Section 5.1).

2. The second step makes a finer analysis of the shape of the remaining regions in order to separate letter CCs from non-letter CCs and moreover to recognize letters (Section 5.2).

5.1 Fast Geometric Filters

This first step consists in removing from the set of extracted CCs (see Section 4), those that do not positively respond to a set of criteria described below.

- Size criterion. Regions with a surface smaller than 5 pixels or larger than 10000 pixels are removed. Regions whose height or width are smaller than 2 pixels or larger than 500 pixels are also removed.
- Shape criterion. Too much twisted regions are removed using a simple concavity measure given by the maximum number of intersections of a horizontal line with the contour of the CC (in our case it is set to 5).
- Density criterion. After a dilation of the segmented image, we count the number of regions that have been merged. If this number is higher than 1000, we assume that it is texture and therefore considered as background. To be less time-consuming, we use an horizontal 10 pixel long segment as structuring element.
- Loneliness criterion. Isolated regions are discarded.

During this first classification step, the multi-scale strategy is important and offers many advantages. A pyramidal analysis permits to be independent of the text size, even if for a specific scale we remove too small or too large CCs. In Figure 1 (c) we give an example of the output of this first classification step. On the ICDAR Natural Scene image database, these filters remove about 80% of unnecessary regions. In the next step, we analyse the shape of the remaining CCs.

5.2 Shape Analysis

After the first step, which removed some non-letter regions from the CCs collection, the goal of the second step is to tag the remaining CCs as letter or non-letter. This shape analysis based classification is divided into two main stages:

- For each CC, a descriptor is first computed based on the wavelet transform (Section 5.2.1).
- All descriptors are classified with a k-Nearest-Neighbours algorithm (Section 5.2.2).

5.2.1 Descriptor Computation

The descriptor computation follows this procedure:

1. The image is transformed into a polar representation,
2. Pixels in this polar representation are accumulated vertically and horizontally to give two signals. These two signals are concatenated into one vector,
3. A wavelet transform is computed on this vector.

Cartesian to Polar Transformation. Cartesian coordinates do not provide a rotation invariant representation, as needed for letter recognition. We first change the CC representation from Cartesian coordinates to polar ones, that allows to design easily rotation invariant descriptors (see Figure 3). We use an optimization of the computation of the polar image coordinates by taking advantage of properties of trigonometric functions, inspired by the work proposed in [1] on rasterization.

Polar to Signal Transformation. Each CC is now represented in a polar space (ρ, θ). All lines and all columns are summed up to get two vectors, each one containing a 1-dimensional signal. Figure 3 presents the projection of pixels in a polar representation on the two axes. The final signal is the concatenation of these two vectors. Note that only the vector built by summing all components horizontally is rotation invariant. But in our case, we do not need a fully invariant descriptor. There are two reasons. Firstly because our training set contains rotated samples and second, because in our tests, we get similar performances by using fully or partially the rotation invariant descriptors.

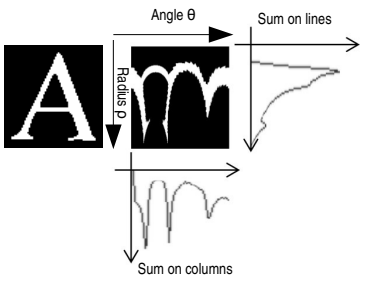


Fig. 3 From Cartesian coordinates to 1D signal. Left: the original letter. Middle: its representation into the (ρ, θ) space. Summation of lines (right) and columns (bottom) of polar image.

Signal to Wavelet Descriptor. Before the classification of vectors obtained by the previous stage, a last stage is

required. Our signal is transformed in order to highlight its specific properties useful for our classification issue. We tested three different kinds of wavelets: Haar [6], Daubechies 4 [6], Cohen-Daubechies-Feauveau 9/7 [45]. Experiences on the classification problem (letter *versus* non-letter) have shown that the most appropriate wavelets in our case were Daubechies 4 (D4) ones, that have given the highest scores. Moreover, these wavelets are fast to compute. Indeed, Daubechies *et al.* [7] introduced an efficient method called wavelet lifting. Instead of applying separately the low-pass and high-pass filters on the input signal, wavelet lifting reduces the number of required instructions by simplifying, factorizing and reordering the operations performed by the filters. Then the input signal is not duplicated and not resampled. It allows to compute the wavelet transform in $O(n)$ (where n is the size of the signal), as compared to $O(nlog(n))$ in our previous chain [10]), using Fourier transform. The small amount of operations needed for this wavelet transform is the key point of our method's efficiency.

5.2.2 K-Nearest-Neighbour Classification

Our classification system tags each CC as letter or non-letter. When our process tags a CC as letter, it simultaneously recognizes it.

CC Classification. We classify each descriptor provided by previous steps. Here a descriptor is also called a sample. Two common classifiers have been considered: the k-Nearest-Neighbors (k-NN) and Support Vector Machine (SVM). We have selected the k-NN for its speed and because it outs performed SVM when both were optimized for the target task. SVM was tested with a Radial Basis Function (RBF) kernel and one of the following strategies: 1. one-vs-one for letter/non-letter classification, 2. one-vs-all with 36 classes for each letters. The k-NN algorithm is a supervised machine learning algorithm that classifies a sample depending on the class of its k neighbors according to a distance criterion. Parameter k was tuned by training different KNN classifiers on the learning database, with $k = 3, 5, 7, 9, 11$. By comparing them, we get correct results for $5 \leq k \leq 9$ and choose $k = 5$ for speed reasons.

For our experiments, we use 32400 samples for learning and 3800 other samples for testing (all with arbitrary font, size and orientation). All these binary samples are mainly extracted from the Itowns[1] database but also from our dataset (a set of pictures taken or collected specifically for this task containing challenging texts present in our daily life). 16200 learning samples containing characters manually labeled are split into 36 classes (26 with uppercase letters and 10 with lowercase letters). Letters (like "c") similar in uppercase and lowercase are not duplicated. Each one contains 450 samples. We obtained 83% of classification rate. 90% of the letters are correctly classified. However only 75% of the background samples are correctly classified. It is partially due to the fact that many elements look like letters. As the classification provides false negatives, CC regions classified as non-letter are only tagged, but not removed from the set.

In Figure 1 (d), one can observe some results of CC classification. Regions in red are elements classified as non-letter, and in green are elements classified as letter. One can notice some classification errors. Indeed, some elements in the background that look like letter have been tagged as letter. This makes sense since each CC is analyzed separately and no context information is taken into account, which limits this technique. That is why this step should then be combined with other more global approach to obtain an accurate text localization system. Notice that before tagging a region as letter or non-letter, we check quickly if the region looks like a bar. Such regions get a special tag. They can as well be a part of the background or a letter like i or l. This tag is specifically used to adapt some of the grouping rules. In Figure 1 (e), one can see the letter i is tagged as a bar.

Letter Recognition. In addition, one of the benefits of our method is that, for CCs tagged as letter, it also identifies the letter without any additional computational cost. However, we only get this recognition at a specific scale, and are not able to merge these results obtained at difference scales for the moment. These information may be useful for the grouping step. Figure 4 shows the letter recognition in superimposed orange at a specific scale and polarity. The transcription is provided under the thumbnails.

6 Grouping

Most of the text detection algorithms integrate a grouping step whose goal is to fuse text regions and form text strings. Most of the time, this grouping only uses criteria that do not consider vertical, oblique or curved text present in natural images. We have developed a new strategy that *catches* various text styles based on a graph representation and its traversal. The process is summarized below:

[1] http://www.itowns.fr/

"TESCD" "CANCER R SEDKCHDK" "RACE EOR LIEE"

Fig. 4 Example of letter recognition.

- We build a graph that links each tagged CC with its neighbors (see Section 6.1).
- Different graph traversals are used for regularity searches. If an explored path is consistent in the graph, all CCs it links are grouped together (see Section 6.2).
- To get a precise region surrounding the text, a mask is then created which cuts out the previous grouped CCs (see Section 6.3).

6.1 Graph Creation

We use a graph whose nodes correspond to tagged CCs (letter or non-letter), and whose edges link regions close to each other (Euclidean distance on spatial positions). For each tagged CC, we look for its $k = 3$ neighbors among regions with similar characteristics, *i.e.*, regions:

- with similar thickness (maximum difference: 4 pixels) and proportions (up to a factor of 3 for width, height, surface and perimeter),
- close enough relatively to its size (distance up to a factor of 3.5 of its elongation),
- that are not concentric (maximum difference of center: 4 pixels).

Then, two regions R_1 and R_2 are linked together, *if and only if* R_1 is a k-NN of R_2 and R_2 is a k-NN of R_1. For the specific case of CCs tagged as bar (see Section 5), and to allow to link small width CCs (containing letters such as *i*, *I* or *l* for example) these constraints are relaxed. Then, we get a graph of adjacency of the CCs (see Figure 1 (e)).

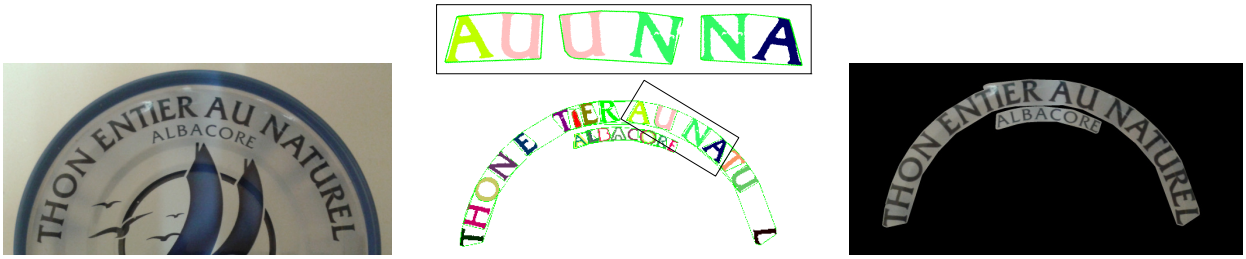
6.2 Graph Traversal

We assume that texts in natural images have some regularity properties, and consider 3-characters minimum size chains to reduce the number of false positives. We then perform a graph traversal to find these regularities, *i.e.*, consecutive connections that have similar directions (difference in $[-\frac{\pi}{5}, +\frac{\pi}{5}]$) and sizes (difference in $[-\frac{2}{3}, +\frac{2}{3}]$). Theses two constraints are checked only for consecutive regions in a graph path. In fact, in a single text string, letters may have different orientations due to the perspective view or text style, thus consecutive letters properties slightly differ. Our strategy allows to handle any orientation of text, even curved text. Thus, when a consistent path is found, all the CCs it links (also called *pattern*) are considered. If a majority of them are tagged as letter, the whole pattern becomes a text area candidate. Indeed, as some tags can be erroneous, we allow CC tagged as non-letter to be part of the text area candidates. When the number of candidates becomes too large, the graph traversal algorithm can lead to a combinatorial explosion. To avoid this, we limit the length of detected texts to 6 letters maximum. We then get a collection of small text areas which are merged into a text string if they overlap and have a similar direction (scalar product between principal components).

During the grouping process, we also discard some false positives using letter recognition. We discard text candidates that contain a series of the identical letters, corresponding to a periodic feature in the image.

6.3 Mask Creation

Most of the time, output of a text localization algorithm is a bounding box surrounding the detected text. This makes sense because most of the methods assume that the text is horizontal. On the contrary, we provide a precise location of the text in the image. It corresponds to a *mask* having the same size as the input image, in which white pixel areas are candidate text regions, and black ones are non-text area. We can not simply bind candidate text regions by rectangles because, in the cases of slanted text or text following a curve, a lot of pixels of the background would be included in the bounding boxes. To create this mask, the union of the convex hulls of all pairs of successive letters in the candidate text area is computed. This gives the exact boundary of the text within the image. Figure 5 illustrates the principle of the mask creation, in the case of curved text.



(a) (b) (c)

Fig. 5 (a) Input image; (b) Creation of the convex hulls; (c) Image restricted by the obtained mask.

7 Validation

The previous step provides a collection of successive CCs that form candidate text areas. Some of them can however be false positives. Since it is very hard to impose constraints on text areas (orientation, font, size, *etc.*) in natural images, objects with strong edges (window, tree branch, fence, *etc.*) can be easily classified as text and therefore decrease the localization's precision rate. A validation step is then added after the localization (Figure 1 (f)) to confirm or refute that the text area candidates really correspond to text. It is actually an additional classification step that differs from the previous local one, since it is global for each set of CCs. This validation stage is based on texture analysis with classical texture descriptors (Section 7.1) and the classification is performed by a SVM (Section 7.2).

7.1 Feature Extraction

Each potential text region area is normalized to a fixed size square using a bilinear interpolation. Our experiments have shown that a 64×64-pixel resolution is suitable for our problem. Since the localization process is based on CCs, we choose to validate the text candidates using an orthogonal approach, *i.e.*, we consider texture-based information. In our case, we extract and combine three kinds of texture features: the Gray Level Co-occurence Matrix and Haralick features, the Histogram of Oriented Gradient and Local Binary Patterns.

7.1.1 Gray Level Co-occurence Matrix and Haralick Features

The Gray Level Co-occurrence Matrix (GLCM) and Haralick's texture features [14] are used due to their ability to capture both information about the distribution of intensities and their relative spatial position. We compute four GLCMs on directions 0°, 45°, 90°, and 135°. Next, six Haralick features (contrast, homogeneity, correlation, energy, dissimilarity and entropy) are computed for each one of the GLCM, leading to a 24-size feature vector.

7.1.2 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) have been used for object detection tasks, such as person detection [5] or text detection [30]. This descriptor can successfully capture gradient orientation distributions of text contours. Here, we extract the HOG in the image restricted to the mask described in Section 6.3. It contains 33 bins: 32 bins which capture the different gradient orientations, and 1 bin dedicated to homogeneous zones (*i.e.* with small gradient magnitudes). Each HOG is normalized with respect to the number of pixels in the mask, in order to achieve a scale invariant feature.

7.1.3 Local Binary Patterns

Local Binary Patterns (LBP) [34] are texture descriptors introduced for measuring the local contrast for efficient texture classification. They are translation and illumination change invariant and therefore suitable for text extraction. Moreover, they are less time-consuming than other texture descriptors. The principle is to associate each pixel of the image with a code that can have $2^8 = 256$ possible values. For that, each pixel value (called "center pixel value") of the original image is compared to the one of its 8 neighbors in a clockwise order: a neighbor gets label 1 if its value is higher, otherwise 0. By reading binary labels in a clockwise order (from the north pixel to the north west one), we get a binary 8-value code, whose decimal values belong to $[0, \ldots, 255]$, also called texture pattern code. A texture pattern code is uniform if it contains a maximum of two bitwise transitions from 0 to 1 or 1 to 0. The LBP uniform pattern code histogram is computed, in order to get the LBP feature that contains the 60 most frequent uniform pattern codes and their frequency, leading to

a 120-size vector. In our approach, we use the Edge-LBP [51], a noise invariant version of the traditional LBP. The difference comes from the comparison used to differentiate the center pixel from its neighbors (LBP threshold is fixed to 1). In our experiments, we set it to 5, since it was a good compromise for a larger variety of letter objects.

7.2 Support Vector Machine (SVM)

SVMs are easy to learn classifiers with high generalization ability. For our validation process we used a non linear SVM, based on RBF kernel [17]. The learning process was carried out on 8628 sample images: 4314 positive examples and 4314 negative examples with their respective mask. All images are text candidates extracted by our detector from Itowns database. Samples are manually labeled and their content varies in size, orientation, font, *etc.* A 177-size feature vector was obtained by concatenating the three features, described in the previous sections. To avoid any orientation dependency, images containing text with arbitrary font size and direction were furthermore rotated in 4 different directions (0°, 45°, 90°, and 135°) during the training process.

To evaluate the efficiency of the validation, we run twice our text detection on the ICDAR 2013 Robust Reading Competition testing database [22] - one excluding the validation and the other one including the validation. Theses experiments show that the activation of the validation step successfully decreases the number of false positives. It removes more than 82% of false positives while removing only less than 4% of true positives. We noticed that the GLCM with Haralick features outperforms the two others texture features. The HOG and LBP do not improve results so much but refine them by better managing some specific cases (especially some periodical patterns) that are not well handled by Haralick features. Figure 1 (f) shows an example of an invalid and a valid text area with our classification process.

8 Multi-scale Binarization

The previous steps have yielded a set of masks corresponding to precise localizations of the text. There is one mask for each scale (depending of the image size) and for each polarity. However, a same text region may have been detected at different scales or/and two different polarities. The main goal of the binarization step is to merge all these detections to provide a binarization of each letter at the scale one.

Two problems can arise to perform this binarization. First, we need to upscale binarization for detections at a lower scale, which does not provide a satisfactory visual result. Secondly, if the same text is detected at different scales, the upscaling of its binarization at the lower scale degrades its binarization at the higher scale.

This is not an easy task nevertheless, a multi-scale detection is very important. By analyzing separately each range of region size, we can consider any size of text. Moreover, if a letter is split (dot text or textured text for example), the analysis at the higher scales fails to detect this letter. At a lower scale, the texture is simplified or all parts of the letters are merged, which enables to detect the text. Note that, because only a subset of regions are proceeded at each scale, the whole processing time is not so much increased.

In order to get correct boundaries of letters, we sequentially consider scales from the higher to the lower. At scale one (higher scale), the boundaries of letters are correct and the localization process precisely indicates which regions are letters and which are not. At this scale, we simply keep regions that have been selected by the detection process. For the lower scales, we upscale the binarization of detected letters to the highest scale. This gives a coarse result that is used to determine which regions are letters in the image segmentation at scale one. Every pixels of the binarized regions are projected onto the segmentation of the image at scale one. Each pixel of the binarized region vote for the region in which it is included in the segmentation of the image at scale one (if they have a comparable color). All regions that are covered by voting pixels are kept. If the process succeeds, the result is visually much better (Figure 1 (g)).

However, this process may fail: a letter at a lower scale might not correspond to one region or to a whole set of regions at scale one (mainly because the segmentation at scale one may have failed). In this case, and if the same region has not already been binarized at a higher scale, we keep the upscaled binarization of the letter at scale one, which gives a less accurate result.

Selecting corresponding regions in the segmentation at the scale one instead of upscaling its binarization from lower scale visually improves results. Going from the higher scale to the lower one allows to keep the more precise result for texts detected on different scales (as it is supposed to be better at higher scale).

The power of our binarization approach is that it allows to merge results from different scales and polarities (and even color channels if we want to process each color channel independently). We do not only binarize the detected text regions but take also into account the knowledge of the whole detection process. Then,

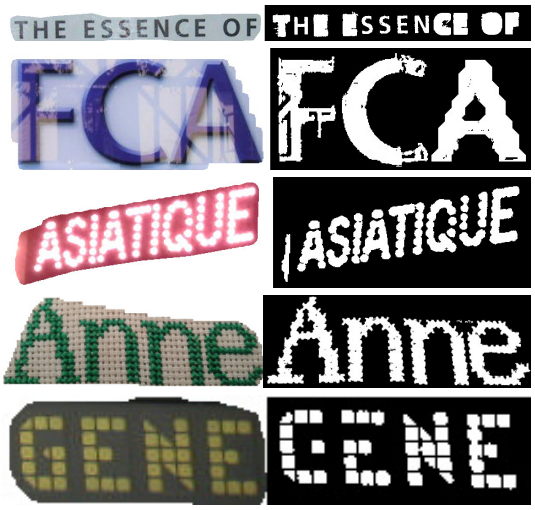


Fig. 6 Example of binarization in upscale and textured cases.

textured text detected at lower scales can also be binarized and textured background will not disrupt the binarization (see some examples in Figure 6).

9 Results and Evaluation

We have presented *TextCatcher*, a method for Latin text detection in images. Its main advantage is that it does not make any assumption on the text style. In this section, we evaluate and compare it with other recent approaches. We first present some qualitative and quantitative results on common datasets of the ICDAR 2013 competition [22]. Next, we compare it with a more challenging set to show its robustness. We also discuss the role of parameters in our algorithm.

9.1 Qualitative Results

Figures 7 and 8 give examples of detection and binarization results. They shows the bounding boxes of the detected text results, the outline of the mask, the extracted text, and the multi-scale binarization results. Despite the difficulty and the variability of texts present in these images, our algorithm has succeeded in extracting all of them. Indeed, one can see that it can efficiently handle curved, textured and multicolor text, text in perspective, cluttered background, degraded or partially occluded text.

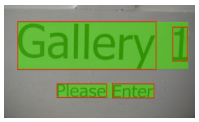


Fig. 9 Line-level detection penalization with the ICDAR 2013 evaluation method: 4 GT objects (red) are matched against 2 detections (green) leading to $R = 0.5$ and $P = 1$.

9.2 Evaluation on the ICDAR 2013 Dataset

We evaluated our text localization system on two benchmarks of database used during the ICDAR 2013 Robust Reading Competition [22].

The first benchmark (challenge 1) contains 141 samples of born digital images for a total of 1696 Ground Truth (GT) text regions. The image size ranges from 194×30 to 660×476 pixels. Although such images are not the main target of our detector, we wanted to evaluate its efficiency on different kinds of images in order to demonstrate its generic properties. The second benchmark (challenge 2) contains 233 natural scene images and a total of 1092 GT text regions. The image size ranges from 350×200 to 3888×2592 pixels.

The evaluation protocol used for ICDAR 2013 is based on the Wolf and Jolion method [49]. It uses the recall R, precision P and F-score to evaluate the detection results with respect to the GT.

However, this evaluation protocol does not always represent the real behavior of the analyzed method. For example, our tests provided a recall and precision around 25% for the challenge 1 and 50% for the challenge 2. However, our method detects more than 25% (respectively 50%) of the text[2]. The problem lies in the used matching strategy. Because the GT annotations of the ICDAR 2013 benchmark are at a word-level, they frequently penalize under and over detections. Since our text localization algorithm provides text detections at line-level, both the recall and the precision evaluated with the ICDAR protocol strongly decrease. When dealing with horizontal text, only one word of the line is most of the time validated as detected, while all others are considered as missed. Moreover, in cases of inclined text lines, no word is validated and the text line is considered as a false positive. Figure 9 shows a detection at line level in green and the GT in red at word level. As it can be seen, although all the text is correctly detected, the detector gets a low recall, $R = 0.5$. It is then impossible to fairly evaluate our method and compare it with other algorithms, because the methods providing the detections at word level are less penalized (first ones in the ranking of the ICDAR challenge). In order to compare our scores with other methods, we split all

[2] The scores of participating are freely available [22]

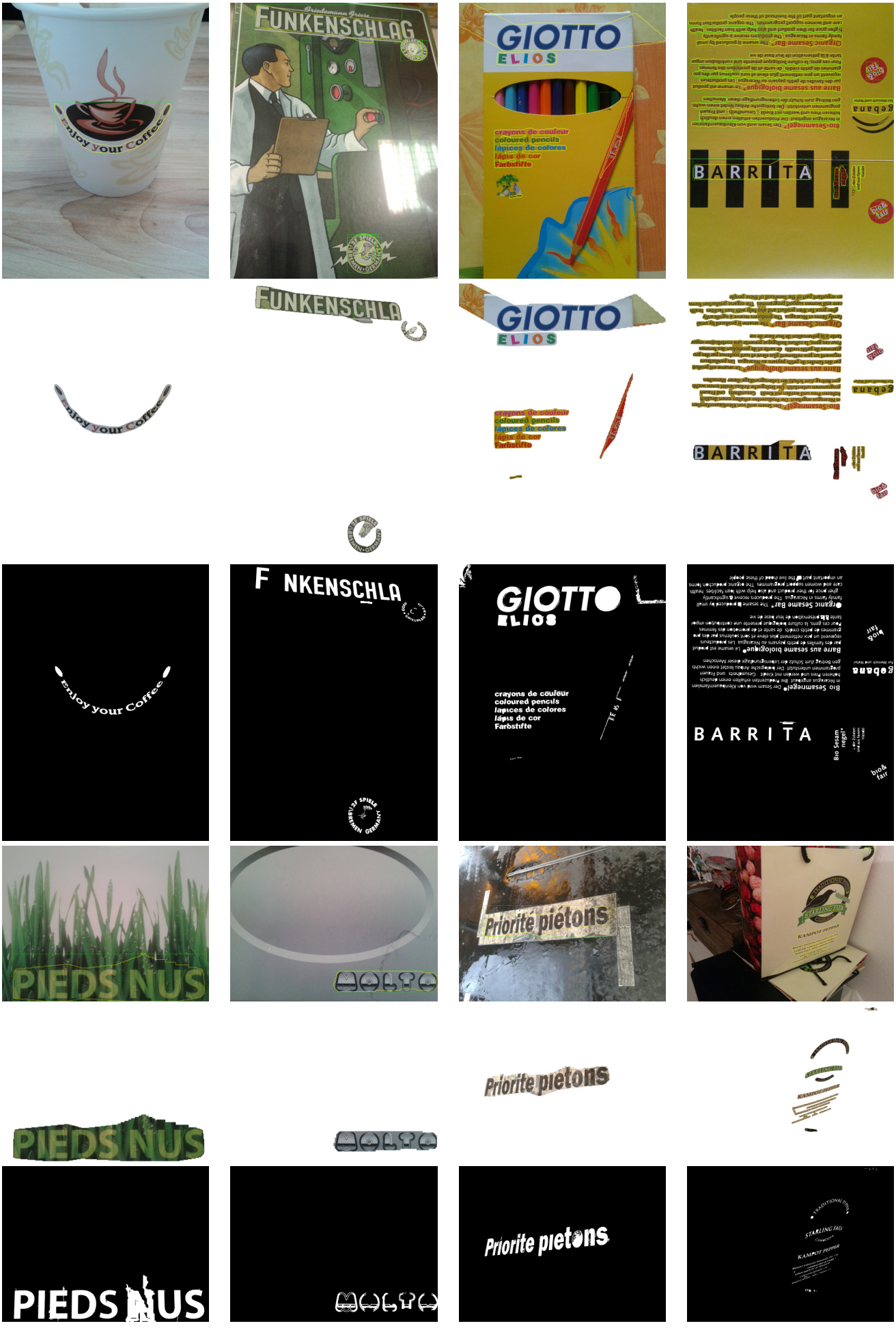


Fig. 7 Text localization and extraction results obtained with our *TextCatcher* approach on our images.

Fig. 8 Text localization and extraction results obtained with our *TextCatcher* approach on ICDAR images.

Method	R	P	F
USTB_TexStar [22]	**86.14**	**89.60**	**87.84**
Text Detection [10]	83.60	85.68	84.63
TH-TextLoc [22]	81.13	82.93	82.02
I2R_NUS_FAR [22]	79.80	83.80	81.75
TextCatcher	74.50	89.16	81.18
I2R_NUS [22]	74.79	84.21	79.22
Baseline [22]	72.00	83.98	77.53
BDTD_CASIA [22]	69.93	79.61	74.46
OTCYMIST [22]	80.33	69.14	74.32
Inkam [22]	61.30	66.23	63.67

Table 4 *TextCatcher* and all participant results on the ICDAR 2013 competition - Challenge 1 (Born-Digital image).

the bounding boxes of detections according to the annotation granularity of the GT. The analysis of the GT gives the positions where the detections need to be split so that they match the GT granularity. This choice is justified later.

Challenge 1: Born-Digital Images. Results of *TextCatcher* and those of all participants during the ICDAR 2013 robust reading competition - challenge 1 are given in Table 4. Even if our detector was not designed for this kind of images, it still provides competitive scores. Notice that our method is designed to be generic, and we make very few assumptions. However we get a precision score comparable to the one of the method ranked first and outperform all other methods that make much more hypothesis on text. The average processing times are 3.2 sec (standard deviation over the dataset $\sigma = 3.4$ sec, ranging from 0.04 sec to 19 sec, depending on the image).

Challenge 2: Scene Images. Results of *TextCatcher* and all participants of the ICDAR 2013 robust reading competition - challenge 2 are given in Table 5. On one hand, one can see that our detector gives the highest recall score (75.6%). This is due to the fact that our method is designed to detect text using fewer hypotheses than other ones. On the other hand, it also explains why our approach does not get the highest precision rate: as it is designed to detect text in any direction, color, *etc.*, it provides more false positives than common methods. Hence, our method reaches the second place among all participants. The average processing time is 6 secs (standard deviation over the dataset $\sigma = 9.5$ secs from 0.5 sec to 1.5 min, depending on the image). The computation time depends of course on the size of the image, but mainly on the number of CCs in the image and also on the number of text candidates to process. 4% of the time is spent on TMMS, 10% on fast filters, 20% to separate text regions from non text regions, 10% for the grouping process and 30% on the validation step. The remaining computation time is spent on the multi-scale pyramid generation, on the multi-scale fusion (10% for binarization) and on some inputs/outputs. These computing times have been measured using a mono-threaded C++ implementation of the process using the Milena library [35], on a Intel® Core™ i5-3450 CPU (3.10GHz).

Stability of the method. Despiteur method is designed to be generic and is not optimized for a specific dataset, *TextCatcher* got competitive results. Furthermore, it is noticeable that *TextCatcher* the F-score values on

Method	R	P	F
USTB_TexStar [22]	70.78	**87.56**	**78.28**
TextCatcher	**75.60**	75.49	75.54
Text_detector_CASIA [41,40]	68.35	82.75	74.86
I2R_NUS_FAR [22]	71.51	74.97	73.20
Text Detection [10]	67.18	78.02	72.19
I2R_NUS [22]	70.54	73.29	71.89
TH-TextLoc [22]	72.33	69.56	70.92
CASIA_NLPR [57,4]	69.55	72.17	70.84
TextSpotter [32,33]	65.97	73.83	69.68
Baseline [22]	35.27	57.66	43.77
Inkam [22]	46.78	41.01	43.71

Table 5 *TextCatcher* and all participant results on ICDAR 2013 competition - Challenge 2 (Scene Images).

Challenge-1 and Challenge-2 are very close (a difference of only 5). This is the smallest difference obtained among all participants (USTB_texStar 78.28-87.84, Text_detection 72.19-84.63, I2R_NUS 71.89-79.22, TH_TextLoc 70.92-82.02, Inkam 43.71-63.67).

A note on the evaluation protocol. The evaluation protocol is still a problem in text detection. Indeed, no existing method can correctly handle the difference of granularity between GT annotations and detection results. In many cases, this leads to unrepresentative evaluations. In our case, the granularity of our text detector differs from the one used in ICDAR competition. To overcome this issue, there are two solutions:

- The first one is to design another evaluation protocol that can handle any type of granularity and therefore is very complex. Moreover one could argue this evaluation protocol is chosen to maximize our scores.
- The second one is to adapt the output of our text detector to the granularity of the GT. The main idea is to add an automatic splitting step at the end of the framework in order to divide results from sentences to words. This strategy is very complex to develop in particular when considering multi directional texts like in our approach and have drawbacks. Even if this splitting step can increase the F-scores, it can degrade the overall quality of the detection results. Indeed, it will help to evaluate properly results (by allowing the matching between detections and GT) but it may introduces mistakes (like dividing a word for example). The major issue is that we evaluate text detection and splitting stages both together and not only the former. Indeed, any failure in the splitting process will be assigned to the detection algorithm. Adding an automatic splitting step leads to unfair comparisons between text detection methods because we want to evaluate only the detection part.

In this paper, we propose a solution in between: we integrate the splitting into the evaluation protocol. For that, we split detection boxes according to the ground truth. Such a choice might be surprising but is relevant for the following reasons.

- The evaluation of a detector is more representative, regardless the granularity of its output (word or sentence level).
- This *ideal* splitting allows the evaluation only focussing on the detection task.
- Because the splitting is not manual, results are unbiased (not subjective choices).
- The same splitting step is applied to every participants and then comparisons are fully consistent.

The main drawback of this solution is that in some situations, it can over-estimate the recall. For example, if the text detector fails and provides the whole image as a text zone, this will give the best recall score. In practice, however, we did not encounter this situation in our evaluations. Even if this automatic splitting is probably not the best solution (the best one is to have an evaluation protocol that can handle any granularity), we do think it gives a relevant comparison between our method and the other ones.

9.3 Evaluation on our challenging Dataset

The ICDAR dataset is mostly restricted to horizontal text, which makes it unsuitable for evaluating algorithms, like ours, that are designed to detect text in harder situations, such as *daily life* scenes. To show the real capacities of our method, we have constituted a new dataset of 51 images (Figure 7) with 227 texts (and their GT) from indoor or outdoor contexts[3]. Texts are more challenging (textured, multi-color, cluttered backgrounds, curved, *etc.*). Below, we give the percentage of each text type present in our dataset (at line level). It is a "coarse" estimation because its is subjective and underestimated when multiple deformation occurs simultaneously.

- 20% of curved text (included 10% of text strictly circular),
- 12% of tilted text (2% with an high slop),
- 6.5% of vertical text,
- 2% of strings containing letters of different colors,
- 8.5% of strings containing scattered letters,
- 13% of blurred or degraded or textured text.

[3] Thumbnails are available at `https://www.lrde.epita.fr/~myriam/images_ijdar2015.zip`. If the article is accepted the database and the GT will be available online.

Method	R	P	F
TextCatcher	**0.53**	**0.33**	**0.40**
LTPTextDetector	0.30	0.32	0.30
OpenCV_detection	0.18	0.28	0.22

Table 6 Results of *TextCatcher* and two other detectors on our dataset.

We compare our detection scores with those obtained by two other efficient detectors for which the codes are available online: the approach in [36][4] that we call "LTPTextDetector" and the implementation by Gomez as a part of the work of Neumann and Matas algorithm [32][5] included in the OpenCV library, that we call "OpenCV_detection".

We used Wolf metric [49] with the DetEval tool (default parameters). We encountered some difficulties in annotating the dataset because the bounding boxes are not well adapted for inclined or circular text. Although the annotations do not always have the same granularity (depending on each situation), all three tested implementations provide detections with the same granularity (line level, however LTPTextDetector splits into words), being most of the time equally penalized.

The results of all methods are given in Table 6. While Neumann and Matas's method [32] has difficulties to detect text in such situations, the implementation of LTPTextDetector gets good results but fails whenever text is not horizontal. The difference in the efficiency of this method and *TextCatcher* is due to the fact that *TextCatcher* is able to detect this kind of text.

9.4 Discussion on Parameters

Our method relies on a set of parameters. Some of them are also simple to fix and are common to any kind of images. Those for the segmentation step are relatively easy to set up thanks to the hysteresis approach. All parameters for the fast filters and grouping are also fixed, whatever the image size is, thanks to the multi-scale approach. The choice of most of these values is then independent on the image size, or on the size of the text we want to detect in them: a text that will be discarded at a scale will be selected at another scale. Although it is possible to adapt these parameter values to a specific known context, in practice this is not necessary when working on a new set of images. In our tests, we always used the same values for all these parameters, whatever the databases. We summarize all required parameters that have been described in the previous sections as well as their range of values in Table 7. Note that all the parameter values have been chosen by testing on a large variety of pictures. In this table we also present the sensitivity of the method against each parameter. For that, we have tested our method on ICDAR Robust Reading Competition dataset (challenge 2 testing dataset). For each parameter assigned a value v, we have tested the algorithm again, once by substituting value v by $v-10\%v$ and second by substituting value v by $v+10\%v$. We then have measured the impact of the F-score of these substitutions. The mention *Low* indicates a variation lower than 10^{-2}/in the order of 10^{-3} while the mention *Medium* indicates a variation around 10^{-2}. These indications show that, even if the method relies on multiple parameters, the impact of these parameters is very limited.

All these tests show that our method is very efficient, even if we are not ranked first in the ICDAR dataset. If we evaluate our method on datasets containing more general image styles, it reaches the first place. With a non optimized and a mono-threaded algorithm, the computation time is reasonable but depends on the size of the image, the number of CCs in the image and the number of text candidates.

10 Conclusion

In this paper, we have proposed an accurate and efficient processing chain to detect and extract text in an arbitrary context (*e.g.* in natural textured scenes). We have tested and compared our method using different datasets and proved that it is competitive to the state-of-the-art methods. Our algorithm has many advantages.

First, our approach is universal because we make only few hypotheses about the text nature (Latin alphabet, block letters). Even if there are some parameters, they do not require to be tuned on a dataset. Moreover, it is versatile because the algorithm is decomposed into modular steps. It is easy to add/remove a component for a specific context: for example, during the grouping step, we easily only consider horizontal or vertical texts. It differs from many other existing chains, that are dedicated to target applications and consequently are more difficult to generalize.

Secondly, we introduce a classification of connected components using a new descriptor based on the wavelet transform. This descriptor allows both to classify candidates as "text" or "non-text", but also, in case of text, to recognize its letters without any additional computational cost.

[4] `https://github.com/mop/LTPTextDetector`

[5] `https://github.com/Itseez/opencv_contrib/blob/master/modules/text/samples/end_to_end_recognition.cpp`

Table 7 Overview of the parameters used in our framework for segmentation (S), fast filters (F), classification (C) and grouping (G) steps. For each one, we report its role in the framework, its value or range of values and its sensitivity (impact on performances).

	Name	Role	Value(s)	Sensitivity
S	k_{min}	Lower hysteresis threshold	$= 20$	Low
	k_{max}	Higher hysteresis threshold	$= 45$	Low
	p	Proximity percentage to erosion	$= 60$	Low
F	S	Surface of regions	$\in [5, 10000]$ pixels	Low
	h, w	Height and width of regions	$\in [2, 500]$ pixels	Low
	N_{inter}	Number of intersections with the horizontal	$\in [1, 5]$	Medium
	N_{merged}	Number of regions merged	$\in [2, 1000]$	Low
C	k	Number of nearest neighbors	$= 5$	
	σ	Size of wavelet descriptors	$= 32$	
G	t_{between}	Inter region thickness proportion	$\in]0, 4]$ pixels	Low
	p_{between}	Inter region proportion	$\in]0, 3]$ pixels	Low
	d_{between}	Inter region distance	$\in [4, 3w]$ pixels	Medium
	θ_{between}	Variation of orientation	$\in [-\frac{\pi}{5}, +\frac{\pi}{5}]$	Medium
	s_{between}	Variation of size proportion	$\in [-\frac{2}{3}, +\frac{2}{3}]$	Low

Third, the grouping step based on a graph allows to localize texts in every direction and even curved text. Usually, a localization algorithm provides bounding boxes containing detected text. This is correct for horizontal or vertical texts but is too coarse for inclined and curved texts (the non-text area in the bounding box is usually larger than the text area). That is why our method provides a mask which contains text strings whose boundaries have been cut out. It provides a more precise localization of the text.

Finally, our multi-scale processing chain allows to detect various text sizes. The combination of detected text at each scale is done by a new binarization process.

The main limitation of our method is that it can also provide false positives since it assumes very few hypotheses. We are currently improving the letter recognition system given by our classification procedure to identify periodical patterns (more than simply detecting repetition of same letters) and not consider them as text candidates. Another problem occurs when letters are stuck together: in such a case, the segmentation can not separate them and this can disrupt the rest of the chain. We would expect to overcome this drawback by adding a splitting process. Although the average computation time is correct, we plan to improve them in case of many of CCs to treat.

References

1. Abrash, Michael: Michael Abrash's Graphics Programming Black Book, 10th edn. Coriolis Group Books (1997)
2. Anthimopoulos, M., Gatos, B., Pratikakis, I.: Detection of artificial and scene text in images and video frames. Pattern Analysis and Applications **16**(3), 431–446 (2013)
3. Arth, C., Limberger, F., Bischof, H.: Real-time license plate recognition on an embedded dsp-platform. In: Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
4. Bai, B., Yin, F., Liu, C.L.: Scene text localization using gradient local correlation. In: International Conference on Document Analysis and Recognition, pp. 1380–1384 (2013)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893 (2005)
6. Daubechies, I.: Ten Lectures on Wavelets. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1992)
7. Daubechies, I., Sweldens, W.: Factoring wavelet transforms into lifting steps. Journal of Fourier Analysis and Applications **4**(3), 245–267 (1998)
8. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: Conference on Computer Vision and Pattern Recognition, pp. 2963–2970 (2010). DOI 10.1109/CVPR.2010.5540041. URL `http://dx.doi.org/10.1109/CVPR.2010.5540041`
9. Fabrizio, J., Marcotegui, B., Cord, M.: Text segmentation in natural scenes using toggle-mapping. In: International Conference on Image Processing, pp. 2349–2352 (2009)
10. Fabrizio, J., Marcotegui, B., Cord, M.: Text detection in street level image. Pattern Analysis and Applications **16**(4), 519–533 (2013)
11. Gao, S., Wang, C., Xiao, B., Shi, C., Zhang, Y., Lv, Z., Shi, Y.: Adaptive scene text detection based on transferring adaboost. In: International Conference on Document Analysis and Recognition, pp. 388–392 (2013)
12. Gatos, B., Ntirogiannis, K., Pratikakis, I.: Icdar document image binarization contest. In: International Conference on Document Analysis and Recognition (2009)
13. Gomez, L., Karatzas, D.: Multi-script text extraction from natural scenes. In: International Conference on Document Analysis and Recognition, pp. 467–471 (2013)
14. Haralick, R., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics **3**(6), 610–621 (1973)
15. Huang, W., Qiao, Y., Tang, X.: Robust scene text detection with convolution neural network induced MSER trees. In: European Conference on Computer Vision, pp. 497–511 (2014)
16. Jaderberg, M., Vedaldi, A., Zisserman, A.: Deep features for text spotting. In: European Conference on Computer Vision (2014)

17. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proc. ECML, pp. 137–142 (1998)
18. Jung, K., Kim, I.K., Jain, A.K.: Text information extraction in images and video: a survey. Pattern Recognition **37**(5), 977–997 (2004)
19. Kan, C., Srinath, M.D.: Scene text localization and recognition with oriented stroke detection. In: International Conference on Computer Vision, pp. 97–104 (2013)
20. Karaoglu, S., Fernando, B., Tremeau, A.: A novel algorithm for text detection and localization in natural scene images. In: Proc. DICTA, pp. 635–642 (2010)
21. Karaoglu, S., Gemert, J., Gevers, T.: Object reading: Text recognition for object recognition. In: Proc. ECCVW - IFCVCR, pp. 456–465 (2012)
22. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Bigorda, L.G., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., de las Heras, L.P.: ICDAR 2013 robust reading competition. In: International Conference on Document Analysis and Recognition, pp. 1484–1493 (2013)
23. Kasar, T., Agarai, G.: Multi-script and multi-oriented text localization from scene images. In: International Workshop on Camera-Based Document Analysis and Recognition, pp. 1–14 (2012)
24. Li, R., Wang, S., Shi, Z.: A two level algorithm for text detection in natural scene images. In: International Workshop on Document Analysis Systems (2014)
25. Li, Y., Shen, C., Jia, W., van den Hengel, A.: Leveraging surrounding context for scene text detection. In: International Conference on Image Processing, pp. 2264–2268 (2013)
26. Mao, J., Li, H., Zhou, W., Yan, S., Tian, Q.: Scale based region growing for scene text detection. In: International conference on MultiMedia, pp. 1007–1016 (2013)
27. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: In British Machine Vision Conference, pp. 384–393 (2002)
28. Meng, Q., Song, Y.: Text detection in natural scenes with salient region. In: International Workshop on Document Analysis Systems, pp. 384–388 (2012)
29. Merino-Gracia, C., Lenc, K., Mirmehdi, M.: A head-mounted device for recognizing text in natural scenes. In: International Workshop on Camera-Based Document Analysis and Recognition, pp. 29–41 (2011)
30. Minetto, R., Thome, N., Cord, M., Leite, N.J., Stolfi, J.: T-hog: An effective gradient-based descriptor for single line text regions. Pattern Recognition **46**(3), 1078–1090 (2013)
31. Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: Asian Conference on Computer Vision, pp. 770–783 (2011)
32. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: Conference on Computer Vision and Pattern Recognition, pp. 3538–3545 (2012)
33. Neumann, L., Matas, J.: On combining multiple segmentations in scene text recognition. In: International Conference on Document Analysis and Recognition, pp. 523–527 (2013)
34. Ojala, T., Pietikinen, M., Harwood, D.: A comparative study of texture measures with classification based on featured distributions. Pattern Recognition **29**(1), 51 – 59 (1996)
35. Olena Team: Milena, generic c++ library for image processing and pattern recognition. URL `https://www.lrde.epita.fr/wiki/Olena/Milena`
36. Opitz, M., Diem, M., Fiel S.and Kleber, F., Sablatnig: End-to-end text recognition with local ternary patterns, mser and deep convolutional nets. In: International Workshop on Document Analysis Systems (2014)
37. Phan, T.Q., Shivakumara, P., Tan, C.L.: Detecting text in the real world. In: International conference on MultiMedia, pp. 765–768 (2012)
38. Prakash, S., Ravishankar, M.: Multi-oriented video text detection and extraction using dct feature extraction and projection based rotation calculation. In: Proc. ICACCI, pp. 714–718 (2013)
39. Serra, J.: Toggle mappings. In: From pixels to features, pp. 61–72 (1989). J.C. Simon (ed.), North-Holland, Elsevier
40. Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S.: Scene text detection using graph model built upon maximally stable extremal regions. Pattern Recognition Letters **34**(2), 107–116 (2013)
41. Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S., Zhang, Z.: Scene text recognition using part-based tree-structured character detection. In: Conference on Computer Vision and Pattern Recognition, pp. 2961–2968 (2013)
42. Shivakumara, P., Basavaraju, H.T., Guru, D.S., Tan, C.L.: Detection of curved text in video: Quad tree based method. In: International Conference on Document Analysis and Recognition, pp. 594–598 (2013)
43. Sumathi, C.P., Santhanam, T., Gayathri, G.: A survey on various approaches of text extraction in images. International Journal of Computer Science and Engineering Survey **3**(4) (2012)
44. Tomer, P., Goyal, A.: Ant clustering based text detection in natural scene images. In: Proc. ICCCNT, pp. 1–7 (2013)
45. Usevitch, B.E.: A tutorial on modern lossy wavelet image compression: Foundations of JPEG 2000. IEEE Signal Processing Magazine **18**(5), 22–35 (2001)
46. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: International Conference on Computer Vision, pp. 1457–1464 (2011)
47. Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: International Conference on Pattern Recognition, pp. 3304–3308 (2012)
48. Wang, X., Song, Y., Zhang, Y.: Natural scene text detection with multi-channel connected component segmentation. In: International Conference on Document Analysis and Recognition, pp. 1375–1379 (2013)
49. Wolf, C., Jolion, J.M.: Object count/area graphs for the evaluation of object detection and segmentation algorithms. International Journal on Document Analysis and Recognition **8**(4), 280–296 (2006)
50. Xu-Cheng, Y., Xuwang, Y., Kaizhu, H., Hong-Wei, H.: Robust text detection in natural scene images. Pattern Analysis and Machine Intelligence **36**(5), 970–983 (2013)
51. Yang, H., Quehl, B., Sack, H.: A framework for improved video text detection and recognition. Multimedia Tools and Applications **69**(1), 217–245 (2014)
52. Yao, C., Xiang, B., Wenyu, L., Yi, M., Zhuowan, T.: Detecting texts of arbitrary orientations in natural images. In: International Conference on Computer Vision, pp. 1083–1090 (2012)
53. Yi, C., Tian, Y.: Assistive text reading from complex background for blind persons. In: International Workshop on Camera-Based Document Analysis and Recognition, pp. 15–28 (2011)

54. Zagoris, K., Pratikakis, I.: Text detection in natural images using bio-inspired models. In: International Conference on Document Analysis and Recognition, pp. 1370–1374 (2013)
55. Zhang, J., Chong, Y.: Text localization based on the discrete shearlet transform. In: ICSESS, pp. 262–266 (2013)
56. Zhang, J., Kasturi, R.: Extraction of text objects in video documents: Recent progress. In: International Workshop on Document Analysis Systems, pp. 5–17 (2008)
57. Zhang, Y., Huang, K., Liu, C.: Fast and robust graph-based transductive learning via minimum tree cut. In: 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011, pp. 952–961 (2011)