Real-time Lexicon-free Scene Text Localization and Recognition

Lukáš Neumann, Jiří Matas

Abstract—An end-to-end real-time text localization and recognition method is presented. Its real-time performance is achieved by posing the character detection and segmentation problem as an efficient sequential selection from the set of Extremal Regions. The ER detector is robust against blur, low contrast and illumination, color and texture variation. In the first stage, the probability of each ER being a character is estimated using features calculated by a novel algorithm in constant time and only ERs with locally maximal probability are selected for the second stage, where the classification accuracy is improved using computationally more expensive features. A highly efficient clustering algorithm then groups ERs into text lines and an OCR classifier trained on synthetic fonts is exploited to label character regions. The most probable character sequence is selected in the last stage when the context of each character is known.

The method was evaluated on three public datasets. On the ICDAR 2013 dataset the method achieves state-of-the-art results in text localization; on the more challenging SVT dataset, the proposed method significantly outperforms the state-of-the-art methods and demonstrates that the proposed pipeline can incorporate additional prior knowledge about the detected text. The proposed method was exploited as the baseline in the ICDAR 2015 Robust Reading competition, where it compares favourably to the state-of-the art.

Index Terms—text-in-the wild, scene text, end-to-end text recognition, photo OCR

1 Introduction

Scene text localization and recognition (also known as *photo OCR* or *text-in-the-wild* problem) is an open problem with many interesting applications, ranging from helping the visually impaired, language translation with automatic input of text written in an unknown script, to indexing large image and video databases by their textual content (e.g. Google Street View, Flickr, etc.).

Unlike traditional printed document OCR, no scene text recognition methods has yet achieved sufficient accuracy for practical applications - the winning method in the most recent ICDAR 2013 contest was able to *localize only* 66% words correctly [1] despite the fact that the dataset is not fully realistic - the word orientations are only horizontal, they occupy a significant part of the image, there is no perspective distortion or significant noise. In the competition, text recognition was evaluated in an artificial setup where the words are manually localized by a human annotator. The winner [2] was able to correctly *recognize* 82% of such "cut-out" words; end-to-end text recognition combining both localization and recognition was not included because of too few potentially participating methods.

Text localization can be computationally very expensive because in an image of N pixels generally any of its 2^N subsets can correspond to text. Generally, two approaches to deal with this issue exist in the literature.

The methods in the first group exploit a sliding-window approach to localize individual characters [2], [3] or whole words [4], drawing inspiration from other object detection problems where this approach has been has been successfully applied [5]. Strengths of such methods include robustness to noise and blur, since features aggregated over the whole region of interest are exploited. The main drawback is that the number of rectangles that needs to be evaluated grows rapidly when text with different scale, aspect, rotation and other distortions has to be found.

YMCA

Fig. 1. Scene text localization and recognition by the proposed method.

The second, recently more popular approach [6]–[8] is based on localizing individual characters as connected components using local properties of an image such as color, intensity or stroke-width. The complexity of these methods does not depend on the parameters of the text as characters of all scales and orientations can be detected in one pass and the connected component representation also provides a character segmentation which can be exploited in an OCR stage. The biggest disadvantage of such methods is a dependence on the assumption that a character is

a connected component, which is brittle - a change in a single image pixel introduced by noise can cause an unproportional change in the connected component size, shape or other properties, which subsequently may affect its classification.

In this paper, we present an end-to-end real-time[1] text localization and recognition method, which does not rely on any prior knowledge of words to be detected (unlike lexicon-based methods [3], where the method output is limited to a relatively small set). The method falls into the latter category since it first detects individual characters and then subsequently builds more complex structures (words, text lines). The real-time performance is achieved by posing the character detection problem as an efficient sequential selection from the set of *Extremal Regions* (ERs). The ER detector is robust against blur, low contrast and illumination, color and texture variation. Its complexity is $O(pN)$, where p denotes number of channels (projections) used and N denotes the number of image pixels.

In the first stage, the probability of each ER being a character is estimated using features calculated with $O(1)$ complexity and only ERs with locally maximal probability are selected for the second stage, where the classification accuracy is improved using more computationally expensive features. A highly efficient clustering algorithm then groups ERs into text lines and an OCR classifier trained on synthetic fonts is exploited to label character regions (see Figure 1). Finally, the most probable character sequence (of segmentations and their labels) is selected in the very last stage of the processing when the context of each character in a text line is known[2].

For the standard ICDAR 2013 dataset and protocol [1], the proposed method achieves state-of-the-art results in text localization (f-measure 76.3%). On the more challenging Street View Text dataset, the proposed method achieves the f-measure of 68.1% for end-to-end text recognition, which significantly outperforms the state-of-the-art methods and demonstrates that the proposed pipeline can be altered to incorporate additional prior knowledge about the detected text (lexicon) if required. The proposed method was also exploited as the baseline in the ICDAR 2015 Robust Reading competition [9], where it compares favourably to the state-of-the art.

Although several individual parts of the proposed method have been previously presented [10]–[13], this paper is the first comprehensive description of the proposed method, which has inspired many derivative works of other authors.

The rest of the paper is structured as follows: In Section 2, an overview of previously published methods is given. The proposed method is described in Section 3. In Section 4, the experimental evaluation is presented. The paper is concluded in Section 5.

2 Previous Work

Numerous methods which focus solely on text localization in real-world images have been published [4], [6], [7]. The "sliding-window" based methods [4] use a window which is moved over the image and the presence of text is estimated on the basis of local image features.

The majority of recently published methods for text localization however uses the connected component approach [6]–[8], [14], [15]. The methods differ in their approach to individual character detection, which could be based on edge detection, character energy calculation or extremal region detection. While the methods are paying great attention to individual character detection, the decision about final segmentation is done at a low level using only local features. The method of Ephstein et al. [6] converts an input image to a greyscale and uses the Canny detector [16] to find edges. Pairs of parallel edges are then used to calculate stroke width for each pixel and pixels with a similar stroke width are grouped together into characters. This method is sensitive to noisy and blurry images because it depends on successfully detected edges and it provides only a single segmentation for each character which might not necessarily be the best one for an OCR module. A similar edge-based approach with different connected component algorithm is presented in [17]. The winning method in text localization of Yin et al. [8] of the ICDAR 2013 Robust Reading competition [1] detect characters as Maximally Stable Extremal Regions (MSERs) [18] and then uses a single-link clustering algorithm with a trained distance.

Other methods focus only on text recognition, where the text is manually localized by a human annotator. The text is recognized either on the character [19], [20] or the whole word level [2], [14], [21]–[24]. The winning method [2] was able to correctly recognize 82.8% of the manually cropped-words in the latest ICDAR Robust Reading competition [1]. Although the methods for cropped-word recognition give an upper-bound on currently achievable text recognition performance, they in fact assume there exists a text localization method with a 100% accuracy, which currently is far from being true. Moreover, since the text was localized by a human, it is not clear that such text localization is even possible without the recognition, because the human annotator could have used the actual content of the text to create the annotation for localization. In other words, it may be impossible to correctly localize text without the recognition phase and therefore joining the text localization and the text recognition into a single

1. We consider a text recognition method real-time if the processing time is comparable with the time it would take a human to read the text.

2. A www service allowing testing of the method is available at http://www.textspotter.org/

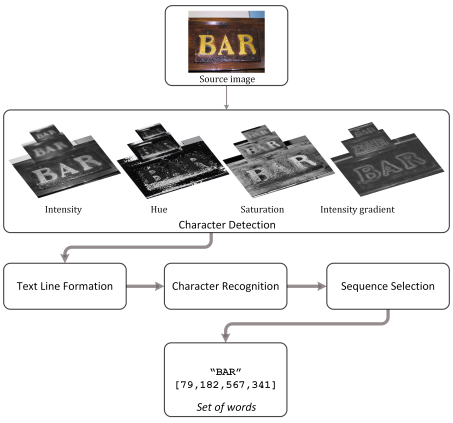


Fig. 2. Overview of the method. For an input image the method finds a set of words, where each word is assigned a Unicode string and its position in a form of a rectangular bounding-box. Character candidates are effectively detected across multiple scales and image projections (channels). Note that this enumeration can be easily parallelized.

end-to-end pipeline may improve the overall system performance.

Only few methods for end-to-end text localization and recognition have been published. The method of Wang and Belongie [3] finds individual characters as visual words using the sliding-window approach and then uses a lexicon to group characters into words. Similarly, the method of Wang et al. [25] uses a sliding-window approach combined with convolutional neural network classifiers. These methods are able to cope with noisy data, but their generality is limited as a lexicon of words (which contains no more than 500 words in their experiments) has to be supplied for each image and the sliding-window approach is limited to horizontal rectangles with a limited number of scales and aspects.

For an exhaustive survey of text localization and recognition methods refer to the ICDAR Robust Reading competition results [1], [26].

3 The Proposed Method

3.1 Character Detection

In the first stage, character candidates are efficiently detected as *Extremal Regions* (ER) selected in a two-stage classification process, operating on a coarse Gaussian scale space pyramid and on multiple image projections (see Figure 2).

3.1.1 Extremal Regions

Let us consider an image $\mathbf{I}$ as a mapping $\mathbf{I} : \mathcal{D} \subset \mathbb{N}^2 \to \mathcal{V}$, where $\mathcal{V}$ typically is $\{0, \ldots, 255\}^3$ (a color image). A channel $\mathbf{C}$ of the image $\mathbf{I}$ is a mapping $\mathbf{C} : \mathcal{D} \to \mathcal{S}$ where $\mathcal{S}$ is a totally ordered set and $f_c : \mathcal{V} \to \mathcal{S}$ is a *projection* of pixel values to a totally ordered set. Let A denote an adjacency (neighborhood) relation $A \subset \mathcal{D} \times \mathcal{D}$. In this paper we consider 4-connected pixels,

(a) (b) (c)

Fig. 3. Intensity gradient magnitude channel ∇. (a) Source image. (b) Projection output. (c) Extremal Regions at threshold $\theta = 24$ (ERs bigger than 30% of the image area excluded for better visualization)

i.e. pixels with coordinates $(x \pm 1, y)$ and $(x, y \pm 1)$ are adjacent to the pixel (x, y).

Region $\mathcal{R}$ of an image $\mathbf{I}$ (or a channel $\mathbf{C}$) is a contiguous subset of $\mathcal{D}$

$$\forall p_i, p_j \in \mathcal{R}\ \exists p_i, q_1, q_2, \ldots, q_n, p_j : p_i A q_1, q_1 A q_2, \ldots, q_n A p_j \tag{1}$$

Outer region boundary $\partial\mathcal{R}$ is a set of pixels adjacent but not belonging to $\mathcal{R}$

$$\partial\mathcal{R} = \{p \in \mathcal{D} \setminus \mathcal{R} : \exists q \in \mathcal{R} : pAq\} \tag{2}$$

Extremal Region (ER) is a region whose outer boundary pixels have strictly higher values than the region itself

$$\forall p \in \mathcal{R}, q \in \partial\mathcal{R} : \mathbf{C}(q) > \theta \geq \mathbf{C}(p) \tag{3}$$

where θ denotes threshold of the Extremal Region.

In this paper, we consider RGB and HSI color spaces [27] and additionally an *intensity gradient* channel (∇) where each pixel is assigned the value of "gradient" approximated by the maximal intensity difference between the pixel and its neighbors (see Figure 3):

$$\mathbf{C}_\nabla(p) = \max_{q \in \mathcal{D} : pAq} \{|\mathbf{C}_\mathbf{I}(p) - \mathbf{C}_\mathbf{I}(q)|\} \tag{4}$$

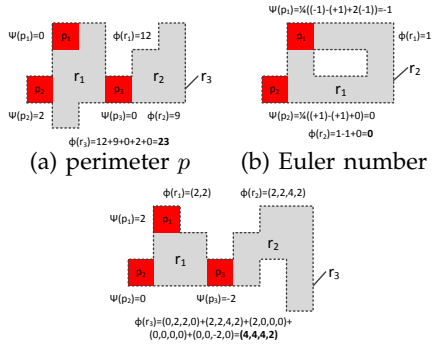
In real-world images there are certain instances where characters are formed of smaller elements (see Figure 4 left) or a single element consists of multiple joint characters (see Figure 4 right). By pre-processing the image in a Gaussian pyramid, in each level of the pyramid only a certain interval of character stroke widths is amplified - if a character consists of multiple elements, the elements are merged together into a single region and furthermore, serifs and thin joints between multiple characters are eliminated. This does not represent a major overhead as each level is 4 times faster than the previous one.

3.1.2 Incrementally Computable Descriptors

The key prerequisite for fast classification of ERs is a fast computation of region descriptors that serve as features for the classifier.

An ER r at threshold θ is formed as a union of one or more (or none) ERs at threshold $\theta - 1$ and pixels of value θ. This induces an *inclusion relation* amongst ERs where a single ER has one or more predecessor ERs (or no predecessor if it contains only pixels of a single value) and exactly one successor ER (the ultimate successor is the ER at threshold 255 which contains

Fig. 4. Processing with a Gaussian pyramid (the pyramid scale denoted by s). Characters formed of multiple small regions merge together into a single region (left column). A single region which corresponds to characters "ME" is broken into two regions and serifs are eliminated (right column)



(a) perimeter p (b) Euler number η

(c) Horizontal crossings $\mathbf{c}_i$

Fig. 5. Incrementally computable descriptors. Regions already existing at threshold $\theta - 1$ marked grey, new pixels at threshold θ marked red, the resulting region at threshold θ outlined with a dashed line

all pixels in the image). As proposed by Zimmerman and Matas [28], it is possible to use a particular class of descriptors and exploit the inclusion relation between ERs to incrementally compute descriptor values.

Let $R_{\theta-1}$ denote a set of ERs at threshold $\theta - 1$. An ER $r \in R_\theta$ at threshold θ is formed as a union of pixels of regions at threshold $\theta - 1$ and pixels of value θ,

$$r = \left(\bigcup u \in R_{\theta-1}\right) \cup \left(\bigcup p \in \mathcal{D} : \mathbf{C}(p) = \theta\right) \quad (5)$$

Let us further assume that descriptors $\phi(u)$ of all ERs at threshold $u \in R_{\theta-1}$ are already known. In order to compute a descriptor $\phi(r)$ of the region $r \in R_\theta$ it is necessary to combine descriptors of regions $u \in R_{\theta-1}$ and pixels $\{p \in \mathcal{D} : \mathbf{C}(p) = \theta\}$ that formed the region r,

$$\phi(r) = \left(\oplus\, \phi(u)\right) \oplus \left(\oplus\, \psi(p)\right) \quad (6)$$

where $\oplus$ denotes an operation that combines descriptors of the regions (pixels) and $\psi(p)$ denotes an initialization function that computes the descriptor for given pixel p. We refer to such descriptors where $\psi(p)$ and $\oplus$ exist as *incrementally computable* (see Figure 5).

It is apparent that one can compute descriptors of all ERs simply by sequentially increasing threshold θ from 0 to 255, calculating descriptors ψ for pixels added at threshold θ and reusing the descriptors of regions ϕ at threshold $\theta - 1$. Note that the property implies that it is necessary to only keep descriptors from the previous threshold in the memory and that the ER method has a significantly smaller memory footprint when compared with MSER-based approaches. Moreover if it is assumed that the descriptor computation for a single pixel $\psi(p)$ and the combining operation $\oplus$ has constant time complexity, the resulting complexity of computing descriptors of all ERs in an image of N pixels is $O(N)$, because $\phi(p)$ is computed for each pixel just once and combining function can be evaluated at most N times, because the number of ERs is bound by the number of pixels in the image.

In this paper we used the following incrementally computed descriptors:

Area a. Area (i.e. number of pixels) of a region. The initialization function is a constant function $\psi(p) = 1$ and the combining operation $\oplus$ is an addition (+).

Bounding box $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$. Top-right and bottom-left corner of the region. The initialization function of a pixel p with coordinates (x, y) is a quadruple $(x, y, x + 1, y + 1)$ and the combining operation $\oplus$ is $(\min, \min, \max, \max)$ where each operation is applied to its respective item in the quadruple. The width w and height h of the region is calculated as $x_{\max} - x_{\min}$ and $y_{\max} - y_{\min}$ respectively.

Perimeter p. The length of the boundary of the region (see Figure 5a). The initialization function $\psi(p)$ determines a change of the perimeter length by the pixel p at the threshold where it is added

$$\psi(p) = 4 - 2|\{q : qAp \wedge \mathbf{C}(q) \leq \mathbf{C}(p)\}| \quad (7)$$

and the combining operation $\oplus$ is an addition (+). The complexity of $\psi(p)$ is $O(1)$, because each pixel has at most 4 neighbors.

Euler number η. Euler number (genus) is a topological feature of a binary image which is the difference between the number of connected components and the number of holes. A very efficient yet simple algorithm [29] calculates the Euler number by counting 2×2 pixel patterns called quads. Consider the following patterns of a binary image:

$$Q_1 = \left\{\begin{matrix}1 & 0\\ 0 & 0\end{matrix}, \begin{matrix}0 & 0\\ 0 & 0\end{matrix}, \begin{matrix}1 & 0\\ 0 & 0\end{matrix}, \begin{matrix}0 & 0\\ 0 & 1\end{matrix}, \begin{matrix}1 & 0\\ 1 & 0\end{matrix}\right\} \quad (8)$$

$$Q_2 = \left\{\begin{matrix}0 & 1\\ 1 & 1\end{matrix}, \begin{matrix}1 & 1\\ 1 & 1\end{matrix}, \begin{matrix}1 & 1\\ 1 & 0\end{matrix}, \begin{matrix}1 & 0\\ 0 & 1\end{matrix}\right\} \quad (9)$$

$$Q_3 = \left\{\begin{matrix}0 & 1\\ 1 & 0\end{matrix}, \begin{matrix}1 & 0\\ 0 & 1\end{matrix}\right\} \quad (10)$$

Euler number is then calculated as

$$\eta = \frac{1}{4}\left(C_1 - C_2 + 2C_3\right)) \quad (11)$$

where C_1, C_2 and C_3 denote number of quads Q_1, Q_2 and Q_3 respectively in the image.

It follows that the algorithm can be exploited for incremental computation by simply counting the change in the number of quads in the image. The value of the initialization function $\psi(p)$ is determined by the change in the number of the quads Q_1, Q_2 and Q_3 by changing the value of the pixel p from 0 to 1 at given threshold $\mathbf{C}(p)$ (see Figure 5b),

$$\psi(p) = \frac{1}{4}\left(\Delta C_1 - \Delta C_2 + 2\Delta C_3\right)) \qquad (12)$$

The complexity of $\psi(p)$ is $O(1)$, because each pixel is present in at most 4 quads. The combining operation $\oplus$ is an addition $(+)$.

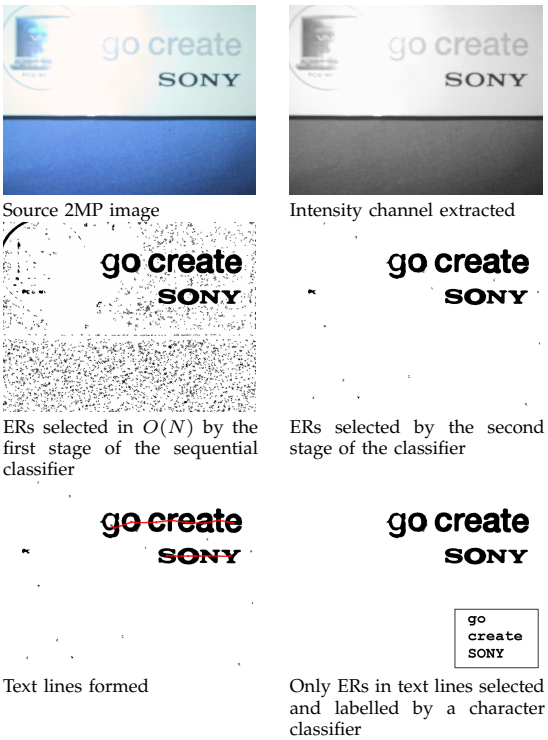
Horizontal crossings $\mathbf{c}_i$. A vector (of length h) with number of transitions between pixels belonging $(p \in r)$ and not belonging $(p \notin r)$ to the region in given row i of the region r (see Figure 5c and 8). The value of the initialization function is given by the presence/absence of left and right neighboring pixels of the pixel p at the threshold $\mathbf{C}(p)$. The combining operation $\oplus$ is an element-wise addition $(+)$ which aligns the vectors so that the elements correspond to same rows. The computation complexity of $\psi(p)$ is constant (each pixel has at most 2 neighbors in the horizontal direction) and the element-wise addition has constant complexity as well assuming that a data structure with $O(1)$ random access and insertion at both ends (e.g. double-ended queue in a growing array) is used.

3.1.3 Sequential Classifier

In the proposed method, each channel is processed separately over a coarse Gaussian pyramid (in the original and inverted projections) and ERs are detected. In order to reduce the high false positive rate and the high redundancy of the ER detector, only distinctive ERs which correspond to characters are selected by a sequential classifier. The classification is broken down into two stages for better computational efficiency (see Figure 6).

In the first stage, a threshold is increased step by step from 0 to 255, incrementally computable descriptors (see Section 3.1.2) are computed in $O(1)$ for each ER r and the descriptors are used as features for a classifier which estimates the class-conditional probability $p(\text{character}|r)$. The value of $p(\text{character}|r)$ is tracked using the inclusion relation of ER across all thresholds (see Figure 7) and only the ERs which correspond to local maximum of the probability $p(\text{character}|r)$ are selected (if the local maximum of the probability is above a global limit $p_{\min}$ and the difference between local maximum and local minimum is greater than $\Delta_{\min}$).

In this paper, a Real AdaBoost [30] classifier with decision trees was used with the following features



	run time	No. of ERs
Initial image	-	6×10^6
Char. det. (1^{st} stage)	820ms	2671
Char. det. (2^{nd} stage)	130ms	40
Text line formation	20ms	25
Character Recognition	110ms	25
Sequence Selection	15ms	12

Fig. 6. Typical number of regions and timings in each stage (character detection in a single channel only) on a standard 2GHz PC

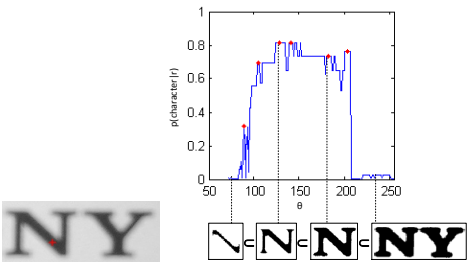


Fig. 7. In the first stage of the sequential classification the probability $p(\text{character}|r)$ of each ER is estimated using incrementally computable descriptors that exploit the inclusion relation of ERs. (a) A source image cut-out and the initial seed of the ER inclusion sequence (marked with a red cross). (b) The value of $p(\text{character}|r)$ in the inclusion sequence, ERs passed to the second stage marked red

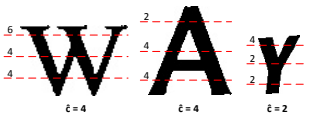


Fig. 8. The horizontal crossings feature used in the 1st stage of ER classification

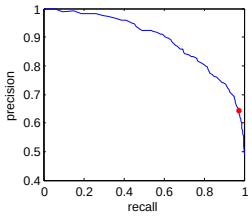


Fig. 9. The precision-recall curve of the first stage of the sequential classifier obtained by cross-validation. The configuration used in the experiments marked red (recall 95.6%, precision 67.3%)

(calculated in $O(1)$ from incrementally computed descriptors): aspect ratio (w/h), compactness ($\sqrt{a}/p$), number of holes ($1-\eta$) and a horizontal crossings feature ($\hat{c}=$ median $\{\mathbf{c}_{\frac{1}{6}w}, \mathbf{c}_{\frac{3}{6}w}, \mathbf{c}_{\frac{5}{6}w}\}$) which estimates number of character strokes in horizontal projection - see Figure 8. Only a fixed-size subset of $\mathbf{c}$ is sampled so that the computation has a constant complexity. The output of the classifier is calibrated to a probability function $p(\text{character}|r)$ using Logistic Correction [31]. The parameters were set experimentally to $p_{\min}=0.2$ and $\Delta_{\min}=0.1$ to obtain a high value of recall (95.6%) (see Figure 9).

In the second stage, the ERs that passed the first stage are classified into character and non-character classes using more informative but also more computationally expensive features. In this paper, an SVM [32] classifier with the RBF kernel [33] was used, the parameter values σ and C were found by cross-validation on the training set. The classifier uses all the features calculated in the first stage and the following additional features:

- **Hole area ratio**. a_h/a where a_h denotes number of pixels of region holes. This feature is more informative than just the number of holes (used in the first stage) as small holes in a much larger region have lower significance than large holes in a region of comparable size.
- **Convex hull ratio**. a_c/a where a_c denotes the area of the convex hull of the region.
- **The number of outer boundary inflexion points** κ. The number of changes between concave and convex angle between pixels around the region border (see Figure 10). A character typically has only a limited number of inflexion points ($\kappa < 10$), whereas regions that correspond to non-textual content such as grass or pictograms have boundary with many spikes and thus more inflexion points.

Let us note that all features are scale-invariant, but not all are rotation-invariant - namely the aspect ratio and the horizontal crossings. However, the features are somewhat robust against small rotations (approx. $\pm 15°$), so text of multiple orientations can be detected by simply rotating the input image 6 times, at the cost of only a slightly lower precision (see Section 4.2).

(a) (b)

Fig. 10. The number of boundary inflexion points κ. (a) Characters. (b) Non-textual content

3.2 Text Line Formation

Let $\mathbf{R}$ denote the set of regions (character candidates) in all channels and scales detected in the previous stage. Even though the cardinality of $\mathcal{R}$ (and subsequently $\mathbf{R}$) is linear in number of pixels, the cardinality of the search space of all sequences is still exponential (the complexity has decreased from 2^{2^n} to 2^n only).

In the proposed method, the space of all sequences is searched by effectively finding all region triplets that could correspond to a (sub-)sequence of characters. Such triplets are then formed into text lines by an agglomerative clustering approach, which exploits bottom line estimates and additional typographical constraints as the distance measure between individual clusters.

Data: a set of regions $\mathbf{R}$
Result: a set of triplets $\mathbf{T}$
$\mathbf{T} \longleftarrow \emptyset$;
for $r^1 \in \mathcal{R}$ **do**
  for $r^2 \in \mathcal{N}(r^1)$ **do**
    if $v(\{r^1, r^2\}) = 0$ **then**
      continue;
    end
    for $r^3 \in \mathcal{N}(r^2)$ **do**
      if $v(\{r^2, r^3\}) = 0$ **then**
        continue;
      end
      $t \longleftarrow \{r^1, r^2, r^3\}$;
      if $v'(t) = 1$ **then**
        $\mathbf{T} \longleftarrow \mathbf{T} \cup t$;
      end
    end
  end
end

Algorithm 1: Exhaustive enumeration of region pairs and triplets to form initial text line candidates

In the first step of the text line formation (see Algorithm 1), character candidates $r^1 \in \mathcal{R}$ are exhaustively enumerated and region pairs and triplets are formed by considering region's r^1 neighbours $r^2 \in \mathcal{N}(r^1)$ and neighbours of the neighbours $r^3 \in \mathcal{N}(r^2)$. In our method, a region r^2 is considered a neighbour of r^1 $\left(r^2 \in \mathcal{N}(r^1)\right)$, if r^2 is amongst $K=5$ closest regions to r^1, where the distance of two regions is measured as the distance of their centroids. Additionally, a left-to-right direction of the text is enforced by limiting the set $r^2 \in \mathcal{N}(r^1)$ to regions r^2 whose centroid is to the right of the centroid of r^1, i.e. $c_x(r^2) > c_x(r^1)$ where $c_x(r)$ denotes the x-coordinate of the region's r centroid.

In the exhaustive search, region pairs (r^1, r^2) and (r^2, r^3) and region triplets (r^1, r^2, r^3) are pruned by constraints v (respectively v'), which verify that the region pair (resp. region triplet) corresponds to the trained typographical model and which ensure the exhaustive search does not combinatorially explode. In our method both constraints are implemented as an AdaBoost classifier [30]; the binary constraint v uses height ratio and region distance normalized by region width as features, whilst the ternary constraint v' uses distance from the bottom line normalized by text line height and region centroid angle. In our experiments, the classifiers were trained on the ICDAR 2013 Training set.

Data: a set of triplets $\mathbf{T}$
Result: a set of text lines $\mathbf{L}$
// Estimate parameters of each triplet and create an initial set of text lines
$\mathbf{L} \longleftarrow \emptyset$;
for $\{r^1, r^2, r^3\} \in \mathbf{T}$ **do**
  $b \longleftarrow$ bottom line estimate of $\{r^1, r^2, r^3\}$;
  $\underline{x} \longleftarrow$ minimal x-coordinate of $\{r^1, r^2, r^3\}$;
  $\overline{x} \longleftarrow$ maximal x-coordinate of $\{r^1, r^2, r^3\}$;
  $\overline{h} \longleftarrow$ maximal height of $\{r^1, r^2, r^3\}$;
  $l \longleftarrow \big(\{r^1, r^2, r^3\}, b, \underline{x}, \overline{x}, \overline{h}\big)$;
  $\mathbf{L} \longleftarrow \mathbf{L} \cup l$;
end
// Agglomerative clustering of text lines
repeat
  // Find two closest text lines
  $d \longleftarrow d_{\text{max}}$;
  for $l \in \mathbf{L}$ **do**
    for $l' \in \mathbf{L} \setminus l$ **do**
      if $dist(l, l') < d$ **then**
        $d \longleftarrow \text{dist}(l, l')$;
        $m \longleftarrow l, m' \longleftarrow l'$;
      end
    end
  end
  if $d < d_{max}$ **then**
    // Merge the two text lines
    $M = \{r^1, \ldots, r^n \in m\} \cup \{r^1, \ldots, r^{n'} \in m'\}$;
    estimate $b, \underline{x}, \overline{x}, \overline{h}$ for M;
    $\mathbf{L} \longleftarrow \mathbf{L} \cup \big(M, b, \underline{x}, \overline{x}, \overline{h}\big)$;
    $\mathbf{L} \longleftarrow \mathbf{L} \setminus m^1, m^2$;
  end
until $d < d_{max}$;

Algorithm 2: Text line formation

In the second step (see Algorithm 2), each triplet is first turned into a text line (of a length 3) and an initial bottom line direction b is estimated by Least Median of Squares. In addition to the bottom line estimate, a horizontal bounding-box which contains all (3) text line regions is calculated and its co-ordinates $\underline{x}$ (left), $\overline{x}$ (right) and $\overline{h}$ (height) are kept as well.

Next, text lines l and l' with smallest mutual distance $\text{dist}(l, l')$ are found, the two sets of their regions are merged together and a new bottom line direction and bounding-box co-ordinates are updated based on the merged set of text line regions. The distance between two lines $\text{dist}(l, l')$ is defined as normalized vertical distance between their bottom lines measured at the beginning and end of a bounding-box formed as a union of the two text lines' bounding-boxes

$$\begin{aligned} \text{dist}(l, l') &= \frac{\max(|b_l(\chi) - b_{l'}(\chi)|, |b_l(\chi') - b_{l'}(\chi')|)}{\min(\overline{h}_l, \overline{h}_{l'})} \\ \chi &= \min(\underline{x}_l, \underline{x}_{l'}) \\ \chi' &= \min(\overline{x}_l, \overline{x}_{l'}) \end{aligned} \tag{13}$$

The process continues until the smallest distance between any two text lines is below the threshold d_{max} (in our experiments, we set $d_{\text{max}} = 0.2$).

As a final step of the text line formation, conflicting text lines are eliminated - two (or more) text lines are in conflict if they contain the same region, which is not permitted as in this paper we assume that a character can be present in one word only (see Section 3.4). Conflicting text lines are typically created in images where the text is arranged into multiple rows - in such case, the same region is a member of two or more different triplets (this is quite common as there is no limitation in Algorithm 1 to ensure unique assignment of a region into one triplet), but in the second step the triplets are not merged together into a single text line because their bottom line direction is completely different.

In order to eliminate the conflicts, text lines which share at least one region with another text line are first grouped into clusters based on the presence of identical regions - two text lines are a member of the same cluster if they have at least one region in common. This process therefore forms isolated text clusters consisting of multiple text lines, which are (possibly transitively) "connected" to each other by a shared region(s), and on contrary each region is present in precisely one cluster. Each cluster is then processed individually in the following iterative process: First, the text line with the the highest number of regions in the cluster is added to the output and all text lines which share a region with the selected text line (including the selected one) are removed from the cluster. The process then continues until the text cluster does not contain any text line. This can be viewed as a voting process, where in each cluster text lines vote for text direction and the text line with highest number of regions (i.e. the text direction of the longest text line) gets selected for the output, whilst all text lines which shared any region with the selected text line (which must have different text direction, otherwise they would have been merged into the selected text line in the previous process) are eliminated.

Note that the algorithm is not affected by the order in which the text lines are processed, because in the first step each text cluster is a transitive closure (where the binary relation between two text lines is given by the existence of a shared region), and in the second step the ordering is given by the descending number of regions in each text line.

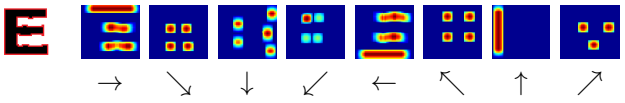


Fig. 11. Character recognition features: Input character (left). Features of the chain-code bitmap for each direction (right).

3.3 Character Recognition

Let $\overline{\mathbf{R}}$ denote a set of all text lines' regions, i.e.

$$\overline{\mathbf{R}} = \bigcup_{l \in \mathbf{L}} \{r^1, \ldots, r^n \in l\} \tag{14}$$

Each candidate region $r \in \overline{\mathbf{R}}$ is labelled with a Unicode code(s) using an (approximative) nearest-neighbour classifier.

A set of labels $\hat{L}(r) \in \mathcal{A}$ of a region r is defined as

$$\hat{L}(r) = \left\{ l(t) : t \in \mathcal{N}_K(f(r)) \mid \|f(t) - f(r)\| \leq \bar{d}(l(t)) \right\} \tag{15}$$

where $l(t)$ denotes label of the training sample (template) t, $\mathcal{N}_K(f(r))$ denotes K nearest-neighbours to the region r in the character feature space f, $\bar{d}(l)$ is a maximal distance for the label (class) l and $\mathcal{A}$ is the set of supported Unicode characters (alphabet). In our experiments, the alphabet consisted of 26 uppercase and 26 lowercase Latin characters and 10 digits ($|\mathcal{A}| = 62$). Let us also note that a region might not be assigned any label, in which case it is rejected in the following step (see Section 3.4).

The region is first normalized to a fixed-sized matrix of 35×35 pixel, while retaining the centroid of the region and aspect ratio. Next, a 8-directional chain-code is generated [34] for boundary pixels and each boundary pixel is inserted into a separate bitmap depending on what direction of chain-code is assigned to it (there are 8 bitmaps of 35×35 pixels, one bitmap for each chain-code direction - see Figure 11). After Gaussian blurring ($\sigma = 1.1$) each bitmap is subsampled to a matrix of 5×5 pixels to generate 25 features for each direction. In total, 25 features $\times$ 8 directions generate 200 features per region.

In our experiments, the training set consists of images with a single black letter on a white background. In total there were 5580 training samples (62 character classes in 90 different fonts). Let us note that no further distortions, blurring, scaling or rotations were artificially introduced to the training set, in order to demonstrate the power of the feature representation. The method can be also easily extended to incorporate additional characters or even scripts, however the recognition accuracy might be affected by the increased number of classes.

The nearest-neighbor classifier $\mathcal{N}_K$ was implemented by an approximative nearest-neighbor classifier [35] for performance reasons and K was set to 11. The values $\bar{d}(l)$ were estimated for each class and each feature representation by a cross-validation on the training set as an average maximal distance over all folds, multiplied by a tolerance factor of β. The value of β represents a trade-off between detecting more characters from fonts not in the training set and more false positives. In our experiments, we used the value $\beta = 2.5$, which yields the best performance on the training subset of the ICDAR dataset (see Section 4).

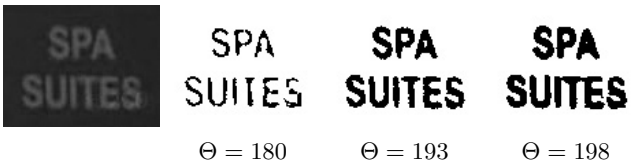


Fig. 12. Character boundaries are often fuzzy and it is not possible to locally determine the threshold value unambiguously. Note the binarization of letters "ITE" in the word "SUITES" - as the threshold Θ is increased their appearance goes from "IIE" through "ITE" to "m"

3.4 Sequence Selection

Let us consider a *word* as a character sequence. Given a set of regions $\overline{\mathbf{R}}$ from the character detector with a set of Unicode labels $L(r)$ for each region r, the method finds a set of words (i.e. a set of character sequences) for each text line where in each sequence a region with a label corresponds to a character and the order of the regions in the sequence corresponds to the order of the characters in the word. Note that a solution might not be unambiguous, because the character detector will typically output several regions for the same character in the image (the same character can be detected with different threshold or in a different projection - see Figure 12), so ultimately there can be several different regions that produce an identical character sequence.

Given regions r_1 and r_2 in a text line, r_1 is an *immediate predecessor* of r_2 $(r_1 \mathcal{P} r_2)$ if r_1 and r_2 are part of the same text line and the character associated with r_1 immediately precedes the one associated with r_2 in the sequence of characters. The predecessor relation $\mathcal{P}$ induces a directed graph G for each text line, such that nodes correspond to labeled regions

$$G = (V, E) \tag{16}$$

$$V = \{r_l \in \mathcal{R} \times \mathcal{A} : l \in \hat{L}(r)\} \tag{17}$$

$$E = \{(r^1_{l_1}, r^2_{l_2}) : r^1 \mathcal{P} r^2\} \tag{18}$$

where r_l denotes a region r with a label $l \in \mathcal{A}$. Regions which don't have any label assigned by the character classifier (i.e. $\hat{L}(r) = \emptyset$) are not part of the text line graph and are therefore eliminated in this process.

In the proposed method, the immediate predecessor relation $\mathcal{P}$ is modeled by ordering regions' centroids in the direction of the associated text line, i.e. a region r^1 is a predecessor of r^2 if the centroid of r^1 comes before the centroid of r^2 in the text line direction, the

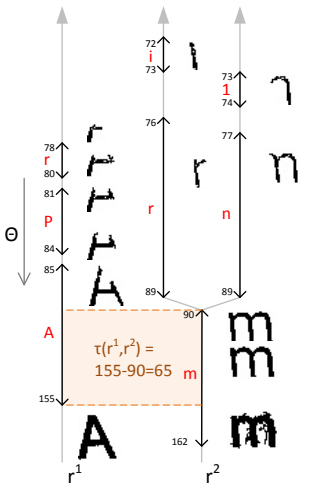


Fig. 13. Threshold interval overlap $\tau(r^1, r^2)$. A threshold interval is an interval of thresholds during which the region has not changed its OCR label (red). Note that as the threshold is increased the region grows or merges with other regions and the label changes

regions' pixels do not overlap and there is no other region closer to r^2 (measured as a distance of the centroids) which satisfies the conditions.

Each node r_l and edge $(r^1_{l_1}, r^2_{l_2})$ has an associated score $s(r_l)$ and $s(r^1_{l_1}, r^2_{l_2})$ respectively

$$s(r_l) = \alpha_1 \psi(r) + \alpha_2 \omega(r_l) \tag{19}$$

$$s(r^1_{l_1}, r^2_{l_2}) = \alpha_3 \tau(r^1, r^2) + \alpha_4 \lambda(l_1, l_2) \tag{20}$$

where $\alpha_1 \dots \alpha_4$ denote weights which are determined in a training stage.

Region text line positioning $\psi(r)$ is calculated as a negative sum of squared Euclidian distances of the region's top and bottom points from estimated position of top and bottom text line respectively. This unary term is incorporated to prefer regions which better fit on the text line.

Character recognition confidence $\omega(r_l)$ estimates the probability, that the region r has the character label l based on the confidence of the character classifier (see Section 3.3). The estimate is calculated by taking the sum of (approximative) distances in the character feature space of at most K nearest templates from training set with the label l, normalized by the distance of the nearest template $d_{\min}$

$$d_{\min}(r) = \min_{t' \in \mathcal{N}_K(f(r))} d(t', r) \tag{21}$$

$$\omega(r_l) \approx \frac{1}{K} \sum_{t \in \mathcal{N}_K(f(r)): l(t)=l} \frac{d_{\min}(r)}{d(t, r)} \tag{22}$$

$$d(x, y) = \|f(x) - f(y)\| \tag{23}$$

Threshold interval overlap is a binary term which is incorporated to express preference for segmentations that follow one after another in a word to have a similar threshold. A *threshold interval* is an interval of thresholds during which the region has not changed its OCR label. A *threshold interval overlap* $\tau(r^1, r^2)$ is the intersection of intervals of regions r^1 and r^2 (see Figure 13).

TABLE 1
Recall (R) and precision (R) of character detection by ER detectors in individual channels and their combinations. The channel combination used in the experiments is in bold

Channel	R (%)	P (%)
R	83.3	7.7
G	85.7	10.3
B	85.5	8.9
H	62.0	2.0
S	70.5	4.1
I	85.6	10.1
∇	74.6	6.3

Channel	R (%)	P (%)
I∪H	89.9	6.0
I∪S	90.1	7.2
I∪∇	90.8	8.4
I∪H∪S	92.3	5.5
I∪H∪∇	93.1	6.1
I∪R∪G∪B	90.3	9.2
I∪H∪S∪∇	**93.7**	**5.7**
all (7 ch.)	94.8	7.1

Transition probability $\lambda(l_1, l_2)$ estimates the probability that the label (character) l_1 follows after the label l_2 in a given language model. Transition probabilities are calculated in a training phase from a dictionary for a given language.

As a final step of the method, the directed graph is constructed with corresponding scores assigned to each node and edge [13], the scores are normalized by width of the area that they represent (i.e. node scores are normalized by the width of the region and edge scores are normalized by the width of the gap between regions) and a standard dynamic programming algorithm is used to select the path with the highest score. The sequence of regions and their labels induced by the optimal path is then broken down into a sequence of words by calculating a median of spacing s_m between individual regions in the whole sequence and by introducing a word boundary where the spacing is above $2s_m$. This process associates a sequence of words (or a single word if no word boundary is found) with each text line, which is the final output of the method.

4 Experiments

4.1 Character Detector

An experimental validation shows that 85.6% characters in the ICDAR 2013 dataset [1] are detected as ERs in a single channel and that 94.8% characters are detected if the detection results are combined from all channels (see Table 1). A character is considered as detected if bounding box of the ER matches at least 90% of the area of the bounding box in the ground truth. In the proposed method, the combination of intensity (I), intensity gradient (∇), hue (H) and saturation (S) channels was used as it was experimentally found as the best trade-off between short run time and localization performance.

4.2 ICDAR 2013 Dataset

The proposed method was evaluated on the ICDAR 2013 Robust Reading competition dataset [1] (which is the same dataset as in the 2011 competition),

TABLE 2
Comparison with most recent text localization results on the ICDAR 2013 dataset

method	recall	precision	f-measure
proposed method	**71.3**	82.1	**76.3**
Yin et al. [8]	68.3	**86.3**	76.2
TexStar (ICDAR'13 winner) [1]	66.4	88.5	75.9
our previous method [13]	64.8	87.5	74.5
Kim (ICDAR'11 winner) [26]	62.5	83.0	71.3

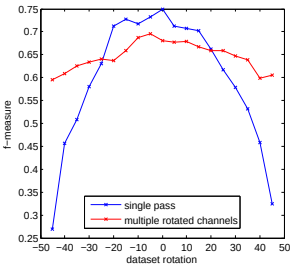


Fig. 14. Text localization performance on the rotated ICDAR 2013 dataset

which contains 1189 words and 6393 letters in 255 images. There are some challenging text instances in the dataset (reflections, text written on complicated backgrounds, textures which resemble characters), but on the other hand the text is English only, it is mostly horizontal and the camera is typically focused on the text area.

Using the ICDAR 2013 competition evaluation protocol [37], the method reaches the recall of 71.3%, precision of 82.1% and the f-measure of 76.3% in text localization (see Figure 15 for sample outputs). The average processing time is 1.6s per image.

The method achieves significantly better recall (71%) than the winner of ICDAR 2013 Robust Reading competition (66%) and the recently published method of Yin et al. [8] (68%). The improvement over our previously published result [13] is due to an improved text line clustering phase, which uses a trained classifier rather than a heuristic function for region pair/triplet verification. The overall f-measure (76%) outperforms all published methods (see Table 2), but the precision is slightly worse than the winner of the ICDAR competition.

The main cause for the lower precision in text localization in some images are missed characters, which then result in a word being only partially detected or detected as multiple words; this is heavily penalized by the evaluation protocol, because such word detection is assigned a precision of 0% (see Figure 16). The proposed method also fails to detect text where there are not enough regions to form a text line (the text formation algorithm needs to form at least one triplet - see Section 3.2), where the word consists of connected letters (even if the line is formed, such region consisting of multiple letters is then rejected by the character classifier) or where there is no threshold in any projection which separates a character from its background - see Figure 17.

In end-to-end text recognition, the method correctly localized and recognized 549 words (46%), where a word is considered correctly recognized if all its characters match the ground truth using case-sensitive

TABLE 3
Comparison with most recent end-to-end text recognition results on the ICDAR 2013 dataset (case-sensitive).

method	recall	precision	f-measure
proposed method	**45.4**	**44.8**	**45.2**
Weinman et al. [36]	41.1	36.5	33.7

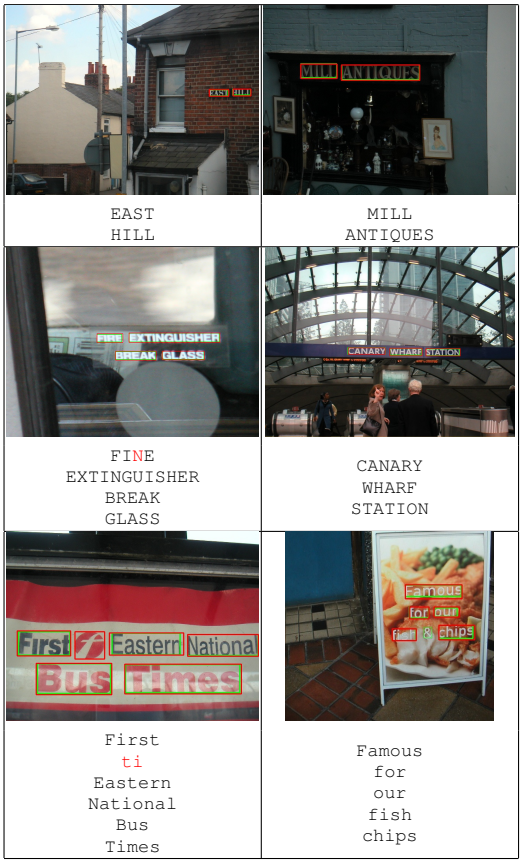


Fig. 15. Text localization and recognition results on the ICDAR 2013 dataset. Ground truth marked green, method output marked red

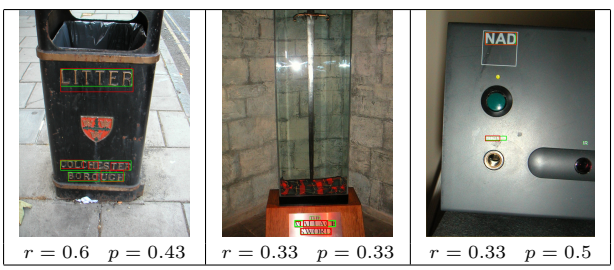


Fig. 16. The main cause for the lower precision in text localization in some images are missed characters, which then result in a word being only partially detected or detected as multiple words. Such partial/multiple detections are heavily penalized by the evaluation protocol (overall image recall r and precision p denoted below each image)

(a) (b) (c) (d)

Fig. 17. Samples of missed text in the ICDAR 2013 dataset. Not enough letters to form a text line (a), very low contrast (b), letters are connected to the surrounding area which has the same color (c) and multiple characters joint together (cursive) (d).

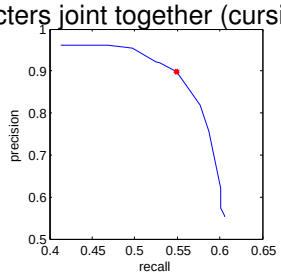


Fig. 18. Precision and recall of end-to-end word detection and recognition on the Street View Text dataset. The configuration with the highest f-measure (68.1%) marked red

comparison, and the method "hallucinated" 60 words in total which do not have any overlap with the ground truth.

The proposed method outperforms the method of Weinman et al. [36] (see Table 3), mostly benefiting from a superior text localization phase. The dataset was also exploited in the ICDAR 2015 Robust Reading competition (see Section 4.4, Table 7).

In the second experiment, the dataset was rotated by 5° increments in the $[-45°; 45°]$ interval, thus creating a synthetic dataset of 4845 images of multi-oriented text with complete annotations. The ability of the proposed method to detect text of different orientations was then evaluated by calculating the average text localization f-measure for each dataset rotation (see Figure 14). With small rotations ($\leq 15°$) the f-measure drops only slightly (the recall remains the same, but the precision is slightly worse), but both the recall and the precision drops for rotations over 30°. If the pipeline is altered to rotate the input image 6 times (using 15° increments) and to combine the rotated channels in the sequence selection stage, the precision on the original dataset drops from 82.1% to 68.1% and the recall remains virtually the same. However, for the rotated dataset the recall is maintained across all rotations and the precision is only slightly worse for rotations over 35° (see Figure 14 - red).

4.3 Street View Text Dataset

The Street View Text (SVT) dataset [3] contains 647 words and 3796 letters in 249 images harvested from Google Street View. Images in the dataset are more challenging because text is present in different orientations, the variety of fonts is bigger and the images

TABLE 4
Comparison with the most recent end-to-end word detection and recognition results on the Street View dataset

method	f-measure
proposed method	**68.1**
proposed method (general language model)	64.7
T. Wang et al. [25]	46.0
K. Wang et al. [3]	38.0

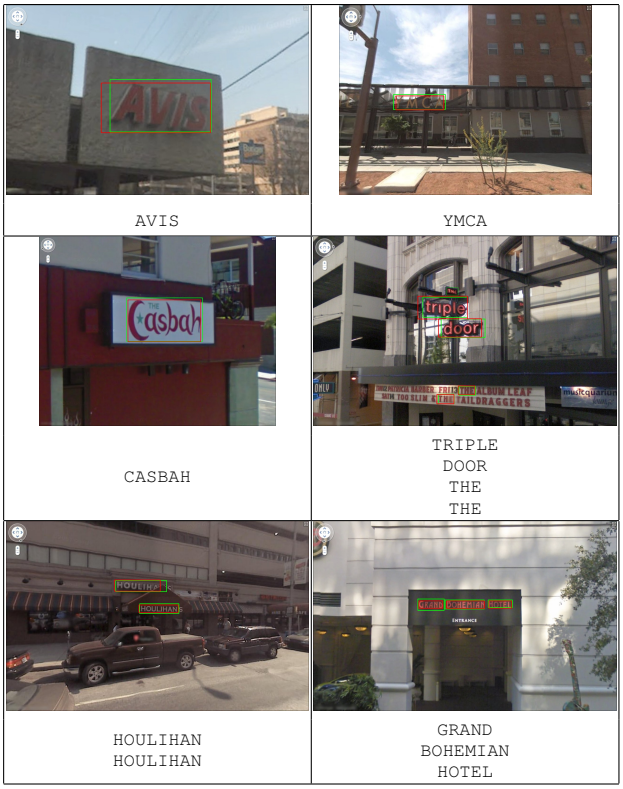


Fig. 19. Text localization and recognition results on the SVT dataset. Ground truth marked green, method output marked red

are noisy; on the other hand, the task formulation is slightly easier because with each test image the evaluated method is also given a list of words (called *lexicon*) and the method only needs to localize (and therefore recognize) words from the lexicon. Let us note that not all lexicon words are in the image (these are called *confuser words*) and vice-versa not all words present in the image are in the lexicon.

In order to make a fair comparison with the previously published work [3], [25] and to make the proposed method compatible with the aforementioned task formulation, the proposed pipe-line was slightly modified in order to exploit the presence of a lexicon. Firstly, the character transition probabilities $\lambda(c_1, c_2)$ (see Section 3.4) were calculated for each image individually from its associated lexicon, which makes the method prefer character sequences present in the lexicon. And secondly, the output of the method was further refined using the image lexicon - the words whose edit distance from a lexicon word is below a selected threshold were considered a match and

(a) (b) (c)

Fig. 20. Samples of missed text in the SVT dataset. Letters are connected to the surrounding area which has the same color (a), multiple letters joint together (b) and artistic fonts (c)

Fig. 21. "False positives" in the SVT dataset are frequently caused by confusing actual text for a confuser word from the lexicon (left column) or by incorrect ground truth where "confuser" words from the lexicon are actually present in the image (right column)

included in the final output, whereas words whose edit distance was above the threshold were discarded. The edit distance threshold is a parameter which makes the method accept more or less similar output words as lexicon words (see Figure 18).

Using the same evaluation protocol as [25], the proposed method achieves the f-measure of 68.1% for the best edit distance threshold, which significantly outperforms the state-of-the-art methods (see Table 4). The method is able to cope with low-contrast and noisy text and high variety of different fonts (see Figure 19 for output examples). The average processing time is 3.1s per image, as the dataset images have a higher resolution.

It can also be observed that many of the detections which are considered as false positives are caused by actual text in the image. This is either caused by the fact that the edit distance of the detected text is too close to a confuser word (see Figure 21 - left column) or by incorrect ground truth where confuser words are actually present in the image[3] (see Figure 21 - right column). The method fails to detect text where letters

3. We have contacted the authors of the dataset with suggestions to correct the ground truth

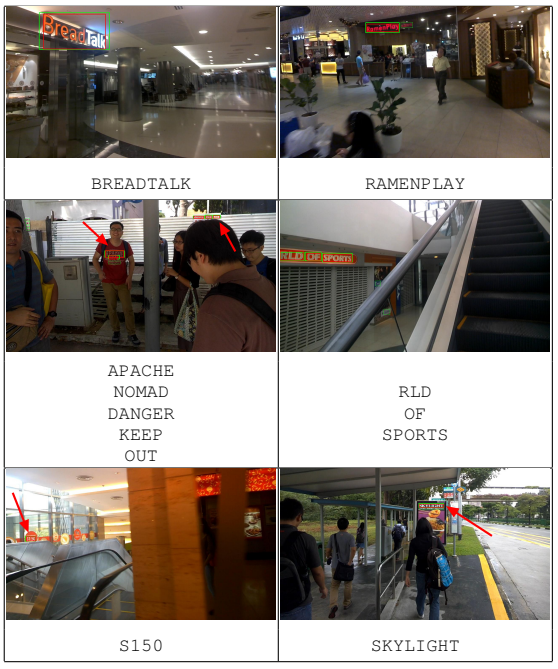


Fig. 22. Text localization and recognition results on the ICDAR 2015 Incidental scene text dataset. Best viewed zoomed in color.

are connected to the surrounding area which has the same color, where multiple letters joint together or where the font is artistic and therefore is not present in the training set (see Figure 20).

If general character transition probabilities $\lambda(c_1, c_2)$ for English (i.e. the same ones as in the Section 4.2) are used instead of lexicon-specific ones for each image, the f-measure drops to 64.7%, which suggests that the method is still competitive even with a general language model.

4.4 ICDAR 2015 Competition

The proposed method was used as the baseline method[4] in the ICDAR 2015 Robust Reading Competition [9] (see Figure 22). In the 2015 competition, the emphasis was on the end-to-end text recognition evaluation, rather than on the individual subtasks (text localization, text segmentation, cropped word recognition) as in the previous years, mainly because the interpretation of individual subtasks' results is problematic because of the evaluation methodology (see Figure 16 for an illustration of the problems in the text localization protocol). Three different datasets were exploited for the evaluation: *Incidental scene text*, *Video Text* and *Focused scene text*.

The new *Incidental scene text* dataset contains $17,548$ annotated text regions in 1670 scene text images captured with Google Glass. The dataset consists of significantly more challenging images due to blur, different text orientations, small text dimensions and

4. The proposed method did not participate in the competition directly to avoid any conflict of interest because the authors helped with data annotation of the newly introduced dataset

TABLE 5
ICDAR 2015 Robust Reading competition [9] - End-to-end Incidental text recognition

Method	Strong			Weak			Generic		
	p	r	f	p	r	f	p	r	f
Stradvision-2	67.9	32.2	**43.7**	-	-	-	-	-	-
proposed method	62.2	24.4	35.0	25.0	**16.6**	**19.9**	18.3	13.6	15.6
Stradvision-1	28.5	**39.7**	33.2	-	-	-	-	-	-
NJU	48.8	24.5	32.6	-	-	-	-	-	-
BeamSearch CUNI	37.8	15.7	22.1	33.7	14.0	19.8	29.6	12.4	**17.5**
Deep2Text-MO [8], [15]	21.3	13.8	16.8	21.3	13.8	16.8	21.3	**13.8**	16.8
OpenCV+Tessaract	40.9	8.3	13.8	32.5	7.4	12.0	19.3	5.0	8.0
BeamSearch CUNI+S	**81.0**	7.2	13.3	**64.7**	5.9	10.9	**35.0**	3.8	6.9

TABLE 6
ICDAR 2015 Robust Reading competition [9] - End-to-end Video text recognition

Method	MOTP	MOTA	ATA
proposed method	**69.5**	**59.8**	**41.8**
Stradvision-1	69.2	56.5	28.5
USTB-TexVideo [8], [15]	65.1	45.8	19.8
Deep2Text-I [8], [15]	62.1	35.4	18.6
USTB-TexVideo-II-2 [8], [15]	63.5	50.5	17.8
USTB-TexVideo-II-1 [8], [15]	60.5	21.2	13.8

many textures similar to text. It was introduced to reduce the possibility of overfitting and to address the aforementioned problems of the ICDAR 2013 Dataset (see Section 4.2), which is now referred to as the *Focused scene text* dataset.

In order to make a fair comparison between methods and to see the impact of prior knowledge, each dataset comes with three lexicons of a different size: the *Strong* lexicon contains 100 words specific for each image, the *Weak* lexicon contains all words in the dataset and the *Generic* lexicon contains 90K English words.

On the *Incidental Scene Text* dataset, the proposed method placed second using the *Strong* lexicon (topped only by the deep-network based StradVision method, which is not published) and placed first using the *Weak* lexicon (see Table 5). The best result with the *Generic* lexicon is achieved by the BeamSearch method, which is based on the proposed method differing only is its more sophisticated language model.

For the cropped word recognition subtask, the proposed method recognized 14.2% words correctly. The main reason for the lower performance when compared to the end-to-end setup is the requirement to initially detect at least 3 characters on a line, which is less likely to be successful for images of individual cut out words (and impossible for words containing less than 3 characters) - 33.8% individual words were missed completely in the cropped word recognition setup because of this limitation.

TABLE 7
ICDAR 2015 Robust Reading competition [9] - End-to-end Focused text recognition

Method	Strong			Weak			Generic		
	p	r	f	p	r	f	p	r	f
VGGMaxBBNet [38]	89.6	**83.0**	**86.2**	-	-	-	-	-	-
Stradvision-1	88.7	75.0	81.3	84.0	**73.7**	**78.5**	69.5	65.0	67.2
proposed method	85.9	69.8	77.0	61.5	64.8	63.1	50.7	58.1	54.2
Deep2Text-II [8], [15]	81.7	69.8	75.3	81.7	69.8	75.3	81.7	**69.8**	**75.3**
NJU	80.2	69.6	74.5	-	-	-	-	-	-
Deep2Text-I [8], [15]	84.0	66.7	74.4	84.0	66.7	74.4	**84.0**	66.7	74.4
MSER-MRF	84.5	61.4	71.13	-	-	-	-	-	-
BeamSearch CUNI	67.9	59.0	63.1	65.1	57.5	61.0	59.4	52.9	56.0
OpenCV+Tessaract	75.7	49.0	59.5	69.5	47.1	56.0	51.0	37.6	43.3
BeamSearch CUNI+S	**92.8**	15.4	26.4	**89.1**	13.5	23.3	65.5	12.0	20.3

On the *Video Text* dataset containing 15 test video sequences, the proposed method (by processing the video frame by frame and by feeding its output to the FoT tracker [39]) outperformed all participants (see Table 6) in all three metrics [40]: the Multiple Object Tracking Precision (MOTP), the Multiple Object Tracking Accuracy (MOTA) and the Average Tracking Accuracy (ATA).

On the *Focused Scene Text* dataset (i.e. the ICDAR 2013 Dataset), the proposed method in the end-to-end setup is outperformed only by the deep network of Jaderberg et al. [38] trained on significantly more data and the deep-network based StradVision method, which is not published (see Table 7).

5 Conclusions

An end-to-end real-time text localization and recognition method is presented in the paper. In the first stage of the classification, the probability of each ER being a character is estimated using features calculated by a novel algorithm of $O(1)$ complexity and only ERs with locally maximal probability are selected across several image projections (channels) for the second stage, where the classification is improved using more computationally expensive features. A highly efficient exhaustive search is then applied to group ERs into text lines and an OCR classifier trained on synthetic fonts is exploited to label character regions. Finally, the most probable character sequence (of segmentations and their labels) is selected by standard dynamic programming in the very last stage of the processing when context of each character in a text line is known.

The method was evaluated on three public datasets. On the ICDAR 2013 dataset the method achieves state-of-the-art results in text localization (recall 71.3%, precision 82.1%, f-measure 76.3%). On the more challenging Street View Text dataset the proposed method achieves the f-measure of 68.1%, which significantly outperforms the state-of-the-art methods and demonstrates that the proposed pipeline can easily incorporate additional prior knowledge about the detected text (lexicon). Robustness of the proposed method against noise and low contrast of characters is demonstrated by "false positives" caused by actual but unannotated text present in the image.

The proposed method was also exploited as the baseline in the ICDAR 2015 Robust Reading competition, where it placed first in the *Video Text* and second (first when using the *Weak* lexicon) in the challenging *Incidental Scene Text* end-to-end recognition.

Acknowledgment

The authors were supported by the Czech Science Foundation Project GACR P103/12/G084. Lukas

would also like to acknowledge the support of the Google PhD Fellowship and the Google Research Award.

REFERENCES

[1] D. Karatzas, F. Shafait *et al.*, "ICDAR 2013 robust reading competition," in *ICDAR 2013*. IEEE, 2013, pp. 1484–1493.

[2] A. Bissacco, M. Cummins *et al.*, "PhotoOCR: reading text in uncontrolled conditions." ICCV, 2013.

[3] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *ICCV 2011*, 2011.

[4] J.-J. Lee, P.-H. Lee *et al.*, "Adaboost for text detection in natural scene," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, sept. 2011, pp. 429 –434.

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[6] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *CVPR 2010*, pp. 2963 –2970.

[7] L. Kang, Y. Li, and D. Doermann, "Orientation robust text line detection in natural images," in *CVPR 2014*, 2014, pp. 4034–4041.

[8] X.-C. Yin, X. Yin *et al.*, "Robust text detection in natural scene images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 5, pp. 970–983, May 2014.

[9] D. Karatzas, L. Gomez-Bigorda *et al.*, "ICDAR 2015 robust reading competition," in *ICDAR 2015*. IEEE, 2013, pp. 1156–1160.

[10] L. Neumann and J. Matas, "A method for text localization and recognition in real-world images," in *ACCV 2010*, ser. LNCS 6495, vol. IV, November 2010, pp. 2067–2078.

[11] ——, "Text localization in real-world images using efficiently pruned exhaustive search," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, sept. 2011, pp. 687 –691.

[12] ——, "Real-time scene text localization and recognition," in *CVPR 2012*, 6 2012, pp. 3538 –3545.

[13] ——, "On combining multiple segmentations in scene text recognition," in *ICDAR 2013*. California, US: IEEE, August 2013, pp. 523–527.

[14] A. Mishra, K. Alahari, and C. V. Jawahar, "Top-down and bottom-up cues for scene text recognition," in *CVPR 2012*, june 2012, pp. 2687 –2694.

[15] X.-C. Yin, W.-Y. Pei *et al.*, "Multi-orientation scene text detection with adaptive clustering."

[16] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–698, 1986.

[17] J. Zhang and R. Kasturi, "Character energy and link energy-based text extraction in scene images," in *ACCV 2010*, ser. LNCS 6495, vol. II, November 2010, pp. 832–844.

[18] J. Matas, O. Chum *et al.*, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, pp. 761–767, 2004.

[19] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images." in *VISAPP (2)*, 2009, pp. 273–280.

[20] M. Iwamura, M. Tsukada, and K. Kise, "Automatic labeling for scene text database," in *12th International Conference on Document Analysis and Recognition (ICDAR 2013)*, Aug. 2013, pp. 1397–1401.

[21] J. J. Weinman, E. Learned-Miller, and A. R. Hanson, "Scene text recognition using similarity and a lexicon with sparse belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 10, pp. 1733–1746, 2009.

[22] T. Novikova, O. Barinova *et al.*, "Large-lexicon attribute-consistent text recognition in natural images," in *ECCV 2012*. Springer, 2012, pp. 752–765.

[23] C. Yao, X. Bai *et al.*, "Strokelets: A learned multi-scale representation for scene text recognition," in *CVPR 2014*, 2014, pp. 4042–4049.

[24] C.-Y. Lee, A. Bhardwaj *et al.*, "Region-based discriminative feature pooling for scene text recognition," in *CVPR 2014*, 2014, pp. 4050–4057.

[25] T. Wang, D. J. Wu *et al.*, "End-to-end text recognition with convolutional neural networks," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 3304–3308.

[26] A. Shahab, F. Shafait, and A. Dengel, "ICDAR 2011 robust reading competition challenge 2: Reading text in scene images," in *ICDAR 2011*, 2011, pp. 1491–1496.

[27] H. Cheng, X. Jiang *et al.*, "Color image segmentation: advances and prospects," *Pattern Recognition*, vol. 34, no. 12, pp. 2259 – 2281, 2001.

[28] J. Matas and K. Zimmermann, "A new class of learnable detectors for categorisation," in *Image Analysis*, ser. LNCS, 2005, vol. 3540, pp. 541–550.

[29] W. K. Pratt, *Digital Image Processing: PIKS Inside*, 3rd ed. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[30] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, pp. 297–336, 1999.

[31] A. Niculescu-Mizil and R. Caruana, "Obtaining calibrated probabilities from boosting." in *UAI*, 2005, p. 413.

[32] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines*. Cambridge University Press, March 2000.

[33] K.-R. Muller, S. Mika *et al.*, "An introduction to kernel-based learning algorithms," *IEEE Trans. on Neural Networks*, vol. 12, pp. 181–201, 2001.

[34] C.-L. Liu, K. Nakashima *et al.*, "Handwritten digit recognition: investigation of normalization and feature extraction techniques," *Pattern Recognition*, vol. 37, no. 2, pp. 265 – 279, 2004.

[35] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *VISSAPP09*, 2009, pp. 331–340.

[36] J. J. Weinman, Z. Butler *et al.*, "Toward integrated scene text reading," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 2, pp. 375–387, 2014.

[37] C. Wolf and J.-M. Jolion, "Object count/area graphs for the evaluation of object detection and segmentation algorithms," *Int. J. Doc. Anal. Recognit.*, vol. 8, pp. 280–296, August 2006.

[38] M. Jaderberg, K. Simonyan *et al.*, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, pp. 1–20, 2014.

[39] T. Vojíř and J. Matas, "The enhanced flock of trackers," in *Registration and Recognition in Images and Videos*. Springer, 2014, pp. 113–136.

[40] B. Keni and S. Rainer, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, 2008.

Lukáš Neumann is a PhD student at the the Center for Machine Perception of Czech Technical University in Prague, Czech Republic. Lukáš received the MSc degree in system programming (with honors) from the Czech Technical University in 2010. His main research topic is text localization and recognition in real-world images and video. Lukáš has published at major computer vision conferences (ICCV, CVPR, ECCV, ACCV) and he received the best student paper price at the ICDAR 2013. In 2013 Lukáš received a highly prestigious Google PhD Fellowship in computer vision.

Jiří Matas is Professor at the the Center for Machine Perception of Czech Technical University in Prague, Czech Republic. Jiří received the MSc degree in cybernetics from the Czech Technical University in 1987 and the PhD degree from the University of Surrey, UK, in 1995. His research interests include object recognition, image retrieval, tracking, sequential pattern recognition, invariant feature detection, and Hough Transform and RANSAC-type optimization. Jiří has published more than 150 papers in refereed journals and conferences. His publications have approximately 17,000 citations in Google scholar, and his h-index is 48. He received the best paper prize at the BMVC in 2002 and 2005 and at the ACCV in 2007 and has served in various roles at major international conferences (e.g. ICCV, CVPR, NIPS, ECCV), co-chairing ECCV 2004 and CVPR 2007.