FASText: Efficient Unconstrained Scene Text Detector

Michal Bušta, Lukáš Neumann, Jiří Matas
Centre for Machine Perception, Department of Cybernetics
Czech Technical University, Prague, Czech Republic
`bustam@fel.cvut.cz, neumalu1@cmp.felk.cvut.cz, matas@cmp.felk.cvut.cz`

Abstract

We propose a novel easy-to-implement stroke detector based on an efficient pixel intensity comparison to surrounding pixels. Stroke-specific keypoints are efficiently detected and text fragments are subsequently extracted by local thresholding guided by keypoint properties. Classification based on effectively calculated features then eliminates non-text regions.

The stroke-specific keypoints produce 2 times less region segmentations and still detect 25% more characters than the commonly exploited MSER detector and the process is 4 times faster. After a novel efficient classification step, the number of regions is reduced to 7 times less than the standard method and is still almost 3 times faster.

All stages of the proposed pipeline are scale- and rotation-invariant and support a wide variety of scripts (Latin, Hebrew, Chinese, etc.) and fonts. When the proposed detector is plugged into a scene text localization and recognition pipeline, a state-of-the-art text localization accuracy is maintained whilst the processing time is significantly reduced.

1. Introduction

Scene text localization and recognition, a.k.a. the text-in-the-wild problem, is a key component of potential applications such as automated translation, image/video database indexing or assistance to the visually impaired. So far, unlike printed document OCR, no method has reached sufficient accuracy and speed for a practical exploitation.

We propose a novel easy-to-implement stroke detector which is significantly faster and produces significantly less false detections than the detectors commonly used by scene text localization methods. Following the observation that text in virtually any script is formed of strokes, stroke keypoints are efficiently detected (see Figure 1) and then exploited to obtain stroke segmentations (see Figure 2).

As the second contribution, we propose an efficient classification step to eliminate regions which do not correspond to text fragments. Text fragment can be a single character, a group of characters, a whole word or a part of a character. The classifier exploits features already calculated in the detection phase and an effectively approximated "strokeness" feature, which plays an important role in the discrimination between text fragments and a background clutter.

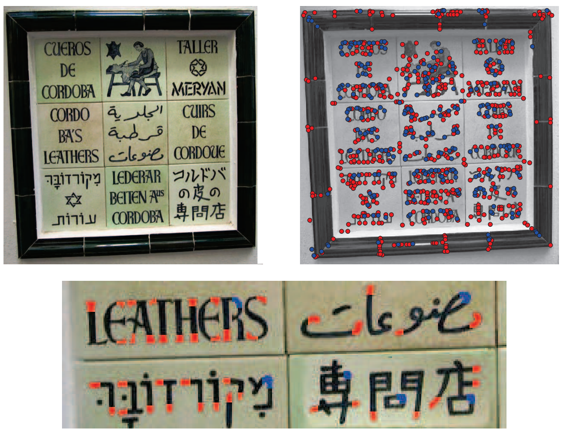


Figure 1: The FASText detector output. *Stroke End Keypoints* (SEK) marked red, *Stroke Bend Keypoints* (SBK) marked blue.

Last but not least, an efficient text clustering algorithm based on text direction voting is proposed, in order to aggregate detected regions into text line structures and to allow processing by subsequent stages (e.g. an OCR module). When the proposed detector is plugged into a scene text localization and recognition pipeline, the state-of-the-art text localization results are maintained whilst the processing time is significantly reduced.

The proposed detector has an application potential for example in real-time scene text detection in a video stream on mobile phones and embedded systems since a straightforward, single-thread non-optimized version of the algorithm runs in near real-time. Since all stages are scale- and rotation- invariant and they are not script-specific, a wide variety of fonts and scripts such as Latin, Hebrew, and Chinese can be detected - see Figure 11.

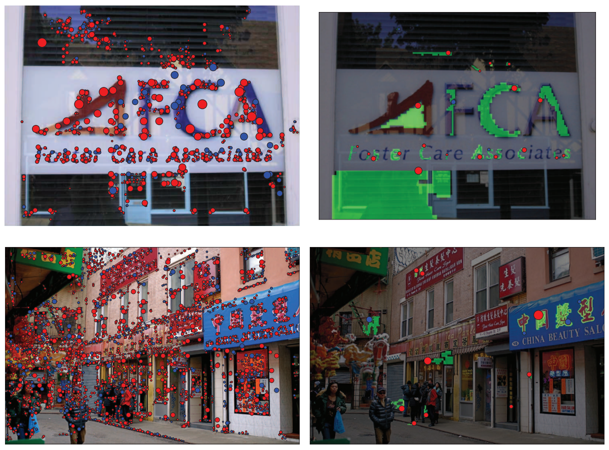


Figure 2: Detected FASText keypoints are effectively exploited to produce stroke segmentations. Detected FASText keypoints (left) and the resulting region segmentations produced by several keypoints selected for illustration purposes (right).

The rest of the paper is organized as follows: in Section 2, an overview of previously published methods is presented. Section 3 introduces the proposed text detection method. The experimental evaluation is covered in Section 4. The paper is concluded in Section 5.

2. Previous Work

Scene text localization is a complex problem and several strategies have emerged in the literature. The first approach is based on a sliding window which is shifted across the image and at each position a presence of a character [26, 2, 8] or a word [9] is checked by a classifier. The main drawback of these methods is that the number of windows that needs to be evaluated grows rapidly if text with different parameters (scale, rotation, aspect) is to be found. Typical processing time ranges from tens of seconds [26] to minutes [9] per a single 1MPx image.

This is the main reason why the region-based approach [5, 28, 16] has become increasingly exploited as text of different parameters can be detected in a single or only a few passes. The recently most successful methods [17, 24, 31, 28, 10, 7] and the ICDAR Robust Reading competition winner [30, 11] also fall into this category as individual characters are found by the MSER detector [14].

Despite being faster than sliding-window methods, the fastest region-based methods have running times ranging from half a second to a second per a 1MPx image. The main cause is that the region detector is not text-specific and therefore the false detections require additional classification step, which slows down the processing. Moreover in the case of the most popular MSER detector, the processing time of the detector itself is still around 0.3s for a 1MPx pixel even though its complexity is linear in the number of pixels [19]. Let us also note that the exact processing time statistics for the published methods are not always available, because time is not measured as part of ICDAR Robust Reading competitions and not all authors publish this information.

The observation that text is formed of strokes has been first exploited in the context of text detection in the wild by Epsthein et al. [5] and more recently by Neumann and Matas [18]. In both cases this observation was used to to design a discriminative feature for classification of regions obtained by a standard, non text-specific detector. The method of Epsthein et al. detects candidate regions using the Canny edge detector [3]. Pairs of parallel edges are used to estimate the stroke width and to group pixels into regions on the basis of a similar stroke width value. The main drawbacks are the method's sensitivity to noise and motion blur (as it relies on successful edge detection) and the relative slowness - the authors state the average processing time is 0.94s per a 1MPx image of the ICDAR 2011 dataset.

The method of Neumann and Matas [18] exploits the MSER detector and then classifies the detected MSERs as either a character, a multi-character or background using the "strokness" feature. The method achieves state-of-the-art results in scene text localization and provides text recognition output, but as in the previous case the main drawback is its reliance on a detector which is not text-specific. The average processing time for both the text localization and recognition stages is 0.8s per image.

Other methods focus only on recognition of text manually found by a human annotator, either as cropped characters [4, 12] or cropped words [27, 15, 20, 2, 29, 13], thus assuming there might exist a text localization method with a 100% accuracy. In the latest ICDAR Robust Reading competition [11], the winner [2] was able to correctly recognize 82.8% of the cropped words.

We refer the reader to the latest ICDAR Robust Reading competition results [11] for a more thorough scene text methods survey.

3. The Proposed Method

3.1. FASText Keypoint Detector

The proposed FASText keypoint detector is, as the name suggests, inspired by the well-known FAST corner detector by Rosten and Drummond [21, 22]. The standard FAST fires on certain letters' corners (e.g. the letter "L" or "P") or corners of character stroke endings if the character is sufficiently thick (e.g. the ending of the letter "l"), but is unable to detect characters whose stroke does not have a corner or an ending (e.g. the letter "O" or the digit "8"). Moreover, in a typical scene image the standard FAST detector produces many false (non-text) and repeated detections (see

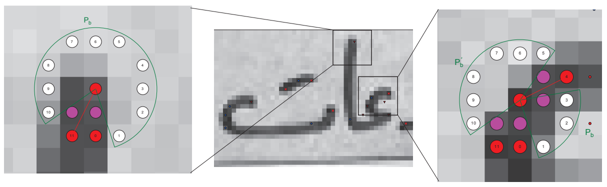


Figure 3: The *Stroke Ending Keypoint* (left) and the *Stroke Bend Keypoint* (right). Pixels in the P_s set marked red, pixels in the P_b set in white. Inner pixels P_i for the connectivity test in purple.

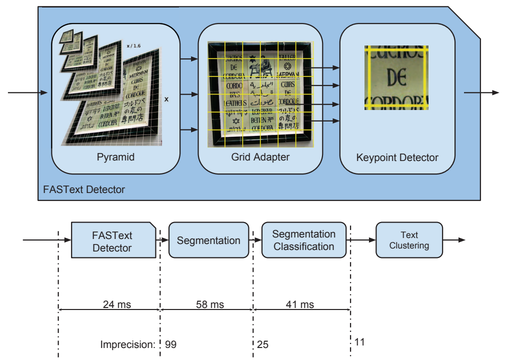


Figure 4: The proposed pipeline. The average processing time and the imprecision (number of false and repeated detections) for a 1MPx image denoted below each stage.

Section 4.1), which unnecessarily slows down the processing.

Considering we are only interested in detecting character strokes, two novel keypoint classes are introduced: the *Stroke Ending Keypoint (SEK)* fires on a stroke ending, whilst the *Stroke Bend Keypoint (SBK)* fires on a curved segment of a stroke (see Figure 3).

For each pixel p in an image, pixel intensities I around a circle of 12 pixels $x \in \{1 \dots 12\}$ are examined and each pixel x is assigned one of three labels

$$L(p,x) = \begin{cases} d, & I_x \leq I_p - m \quad (darker) \\ s, & I_p - m < I_x < I_p + m \quad (similar) \\ b, & I_x \geq I_p + m \quad (brighter) \end{cases} \tag{1}$$

where m is a *margin*, which is a parameter of the detector.

Let P_d, P_s, P_b denote the sets of pixels that are labeled as *darker*, *similar* and *brighter*. Pixel p is a **Stroke Ending Keypoint** (SEK) if there exists two contiguous sets P_s and P_d (or P_s and P_b) such that $|P_s| \in \{1,2,3\}$ and $|P_d| = 12 - |P_s|$ (or $|P_b| = 12 - |P_s|$) and the two sets form a partition. In other words, the pixel p is a SEK if there exists a contiguous circle segment of at least 9 pixels which are darker (or brighter) than the pixel p, whilst the remaining pixels of the circle have a similar intensity to the pixel p. The keypoint can be either positive or negative, depending on whether the intensity of the stroke is higher or lower than the background.

Using the same notation, pixel p is a **Stroke Bend Keypoint** (SBK) if there exists four contiguous sets P_s, P_s', P_d, P_d' or (P_s, P_s', P_b, P_b') such that $|P_s|, |P_s'| \in \{1,2,3\}$ and $|P_d| > 6, |P_d'| = 12 - |P_d| - |P_s| - |P_s'|$ (or $|P_b| > 6, |P_b'| = 12 - |P_b'| - |P_s| - |P_s'|$). The pixel p is a SBK if there is a contiguous circle segment of at least 6 pixels which are darker (or brighter) than the pixel p, two distinct circle segments which have similar intensity to the pixel p and the remaining pixels on the circle are darker (or brighter) than the pixel p.

The implementation of the aforementioned tests is straightforward and the tests can be computed by a single pass around the 12 pixel circle. The computational complexity of the detector is reduced even further (by the factor of 2) by inserting a rule, which examines the opposite pixels and tests that all opposite pixels are brighter than $I_p + m$ or darker than $I_p - m$. An opposite pixel is the pixel which lies directly opposite on the 12 pixel circle. If none of the conditions is met, the pixel p cannot be a FASText keypoint and is quickly rejected without any further processing.

The final verification step of the FASText detector is a connectivity test, which ensures the inner circle pixels P_i between the pixel p and the pixels P_s also satisfy the intensity margin, i.e. $I_p - m < I_x < I_p + m \quad \forall x \in P_i$ (see Figure 3). The purpose of the test is to eliminate false detections, because if the pixel p is placed on a stroke, the pixels in the P_s partitioning(s) must be connected to it. Let us note that this test does not represent any significant overhead, as in the worst case only 3 pixels have to be examined.

In order to eliminate FASText keypoints which lie close to each other, a simple non-maximum suppression is performed on a 3×3 neighborhood. Only the keypoint with the highest contrast (i.e. $\max(I_x - I_p) : x \in P_b$ respectively $\max(I_p - I_x) : x \in P_d$) is kept.

The detector parameter values were found experimentally using the ICDAR 2013 Training dataset [11] containing 4784 characters in 229 images. A character is considered detected, if there is at least one keypoint whose position intersects with the the character ground truth segmentation. The value of each parameter was chosen to obtain the best trade-off between the detector imprecision and the number of missed characters (recall).

The detector imprecision is calculated as $\frac{|D|}{|GT|}$, where $|D|$ is the number of detections and $|GT|$ is the number of characters in the ground truth. For example, the imprecision of 10 implies that a detector produces 10 times more detections than characters in the ground truth.

The FASText detector has four parameters: circle size, margin, scaling factor and the maximum number of keypoints per image. The most important parameter is the cir-

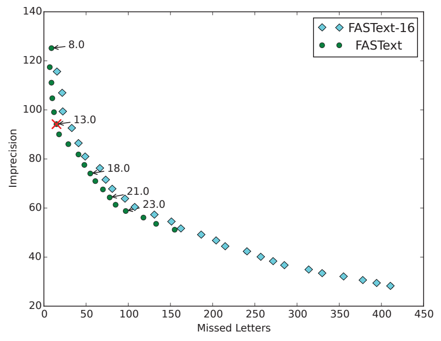


Figure 5: The margin parameter m controls the trade-off between imprecision (the number of false and repeated detections) and the number of missed characters. The value used in experiments marked by the red cross, the detector with the circle size of 16 pixels denoted FASText-16.

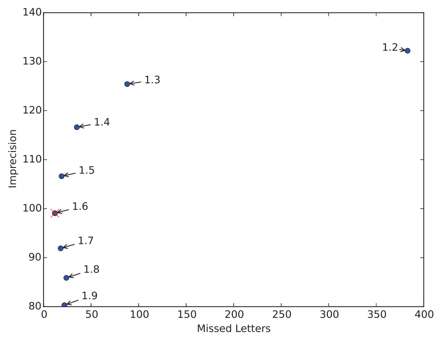


Figure 6: The scaling parameter f controls the trade-off between imprecision and the number of missed characters. The value used in experiments marked by the red cross.

cle size, whose default value of 12 pixels is lower than the original FAST [21] value of 16 pixels to allow detection of characters (strokes) which are close to each other (see Figure 5). Let us note that a circle size smaller than 12 pixels is not possible because of the connectivity test.

The margin m parameter controls the trade-off between imprecision (the number of false and repeated detections) and the number of missed characters (see Figure 5). The margin value used in the experiments was $m = 13$.

Since the FASText detector is only triggered by strokes whose width is comparable to the pixel circle radius (i.e. two or three pixel wide), the keypoints are detected in an image scale-space to allow detection of wider strokes. Each level of the pyramid is calculated from the previous one by reducing the image size by the scaling factor f (in our implementation, bilinear approximation was used for image resizing). The scaling factor f used in the experiments was 1.6 (see Figure 6).

The last parameter of the detector is the maximum number of keypoints per image. An input image is partitioned into uniformly-sized cells (see Figure 4) and the number of detected keypoints in each cell is limited by ordering the keypoints by their contrast and eliminating the keypoints whose position in the ordered set is above the cell limit. The value of 4000 keypoints per image was chosen as a value commonly used by standard keypoint detectors [23].

3.2. Keypoint Region Segmentation

As successfully demonstrated by the methods based on MSERs [17, 30], individual characters can be segmented from the background using a threshold value unique for each character (in MSERs, the threshold value is found as the center of the region stability interval).

In the proposed method, the threshold value is found directly from the FASText keypoint. Given a positive FASText keypoint p and its associated set of darker pixels P_d, the segmentation threshold θ_p is the intensity value just above the intensity of the darkest pixel in P_d

$$\theta_p = \max(I_x) + 1 \,|\, x \in P_d \tag{2}$$

Similarly, for a negative FASText keypoint, the segmentation threshold θ_p is the intensity value just below the intensity of the darkest pixel in P_b

$$\theta_p = \min(I_x) - 1 \,|\, x \in P_b \tag{3}$$

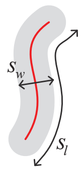
The threshold value θ_p is then effectively exploited by a standard flood-fill algorithm [25] to generate a stroke for each FASText keypoint.

3.3. Region Classification

In order to reduce the still relatively high false detection rate of the FASText detector (the average imprecision is 25 regions to one ground truth character) and to make the processing in the subsequent stages faster, an efficient classification stage is inserted to filter the output of the proposed detector.

Drawing inspiration from a successful MSER segmentation classifier [18], four rotation- and scale-invariant features are employed by a Gentle AdaBoost classifier [6] to classify FASText regions as either a text fragment (typically a character) or a background clutter: compactness, convex hull area ratio, holes area ratio and the *Character Stroke Area* (CSA). All features are efficiently calculated as part of the segmentation process (see Section 3.2), with the exception of the Character Stroke Area feature.

The *Character Stroke Area* (CSA) feature is based on the observation that a character can be drawn by taking a brush with a diameter of the stroke width and drawing through middle points of the character (see Figure 7 left). Since the area of an ideal stroke is the product of the stroke width and the length of the stroke, the "strokeness" of a region



	t / region [μs]
original [18]	895
approximated	15

Figure 7: The *Character Stroke Area* (CSA) feature is based on the observation that the area of an ideal stroke is the product of the stroke width s_w and the length of the stroke s_l (left - adapted from [18]). Character Strokes Area calculation time comparison (right).

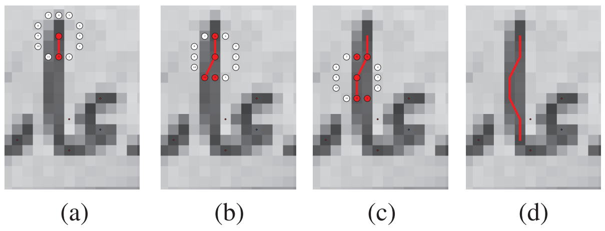


Figure 8: Character Stroke Area (CSA) approximation by FASText keypoints. Initial Stroke Ending Keypoint (a). First *Stroke Straight Keypoint* found (b). Next Stroke Straight Keypoint found (c). All Stroke Straight Keypoints found, stroke length illustrated by the red line (d).

is estimated by calculating the ratio between the Character Strokes Area and the number of pixels in the region. The ratio therefore estimates the proportion of the region pixels which are part of a character stroke and allows to efficiently discriminate between text fragments (characters, groups of characters, whole words or parts of characters) and a background clutter (see Figure 9).

In the original form [18], a distance map and an iterative NMS is employed to calculate the character strokes area. Since the high computational complexity of such process represents a significant overhead, we propose an approximate but significantly faster calculation of the Character Strokes Area feature.

Given a region r, a set of *Stroke Straight Keypoints* (SSK) is found for each Stroke Ending Keypoint (SEK) p which intersects with the region r using the following iterative algorithm (see Figure 8):

1. Start with the Stroke Ending Keypoint p

2. Move the point p to the darkest (brightest) pixel of the P_s pixels (always in the direction away from the stroke ending)

3. The point p is a *Stroke Straight Keypoint* (SSK) if there are four contiguous partitionings P_s, P'_s, P_d, P'_d or (P_s, P'_s, P_b, P'_b) such that $|P_s|, |P'_s| \in \{1, 2, 3\}$ and $|P_d| > 3, |P'_d| = 12 - |P_d| - |P_s| - |P'_s|$ (or $|P_b| > 3, |P'_b| = 12 - |P'_b| - |P_s| - |P'_s|$)

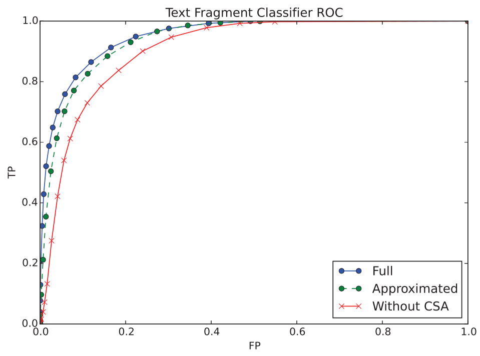


Figure 9: The impact of the Character Strokes Area (CSA) feature in the region classification. The original classifier [18] (blue), the proposed classifier with the approximated CSA feature (green), a classifier without the CSA feature (red)

4. If the point p is a SSK, repeat from the step 2, otherwise terminate

The *character strokes area* $A_s(r)$ of the region r is then calculated as

$$A_s(r) = \sum_{p \in \text{SSK}_r} 3|P_s|_p + \sum_{p \in \text{SBK}_r} 3\Big(|P_s|_p + |P_s|'_p\Big) \quad (4)$$

where SSK$_r$ and SBK$_r$ is the set of Stroke Straight respectively Stroke Bend Keypoints intersecting with the region r and $|P_s|_p$ ($|P_s|'_p$) is the size of the partitioning P_s (P'_s) associated with the keypoint p.

The proposed approximate algorithm is almost 60 times faster (see Figure 7 right), yet the impact on the classification accuracy is negligible (see Figure 9).

3.4. Text Clustering

In this stage, the unordered set of FASText regions classified as text fragments is clustered into ordered sequences, where each cluster (sequence) shares the same text direction in the image. In other words, individual characters (or groups of characters or their parts) are clustered together to form lines of text.

Let us denote the set of text fragment regions as $\mathcal{R}$. For each region $r_i \in \mathcal{R}$, all regions $r_j \in \mathcal{R}$ are found, such that r_i and r_j are neighbors. Two regions r_i and r_j are *neighbors*, denoted $N(r_i, r_j)$, if their centroids are sufficiently close to each other (normalized by their area) and they have a comparable scale:

$$N(r_i, r_j) = \begin{cases} 1, & \|c(r_i) - c(r_j)\| < \alpha\sqrt{\max(A(r_i), A(r_j))} \\ & \max\left(\frac{A(r_i)}{A(r_j)}, \frac{A(r_j)}{A(r_i)}\right) < \beta \\ 0, & otherwise \end{cases} \quad (5)$$

where $c(r)$ is the centroid of the region r and $A(r)$ is the convex hull area of the region r. The parameter values $\alpha = 4$ and $\beta = 10$ were chosen experimentally and they provide sufficient tolerance for a vast majority of typographical models (see Figure 10a). Since the region neighbor search is a simple search in a 2D space of points (centroids), it can be effectively implemented by partitioning the image into smaller cells and always considering regions just in the closest cells. Alternatively, one could use a standard (approximate) nearest-neighbor algorithm.

Each pair of neighboring regions then casts a vote for their text direction, where the text direction and position is given by the line which passes through the two centroids of the neighboring pair (see Figure 10b). Exploiting the Hough transform [1], each vote for a text direction is represented in the polar system $\rho = x\sin(\theta)+y\cos(\theta)$, so that vertical text lines can also be detected.

The two-dimensional parameter space (ρ, θ) is quantized into a fixed-sized matrix, so that small differences in the line parameters are eliminated and the text directions with the highest number of votes (i.e. the directions with the highest number of supporting pairs) can be easily found as local maxima in the matrix (see Figure 10c).

Each local maximum with its parameters (ρ, θ) then unambiguously induces a text cluster by simply taking all regions whose centroid lies on the line (ρ, θ) (or the distance is smaller than the quantization error) and ordering them in the direction of the line.

Since one region lies on multiple lines with different parameters (ρ, θ), the local maxima are processed in a decreasing order of number of their votes and each region is allowed to be included only in a single text cluster. This process ensures that longer text lines are preferred over shorter ones and that intra-line text clusters are eliminated (see Figure 10d).

4. Experiments

4.1. Character Detection

In the first experiment, the character detection ability of the FASText keypoint detector is compared with existing detectors. The evaluation uses the standard ICDAR 2013 Test dataset (which is commonly used for text localization evaluation - see Section 4.2) containing 6393 characters in 255 images annotated at the pixel level, i.e. ground truth character segmentations are provided.

In Table 1, keypoints detected by the FASText and the FAST detector [21] are compared by processing all images in the dataset and calculating keypoint statistics for each detector (see Figure 12 for a visual comparison on a sample image). The total number of detections is almost identical for both detectors, but the proposed FASText detector misses 3 times less characters (only 111 characters are missed against 328 missed characters by the FAST detector). A character is considered missed by a detector, if no pixel in the ground truth segmentation coincides with a keypoint. The FASText detector is also 20% faster than the FAST detector.

In Table 2, region segmentations produced by the detectors commonly used in scene text localization are com-

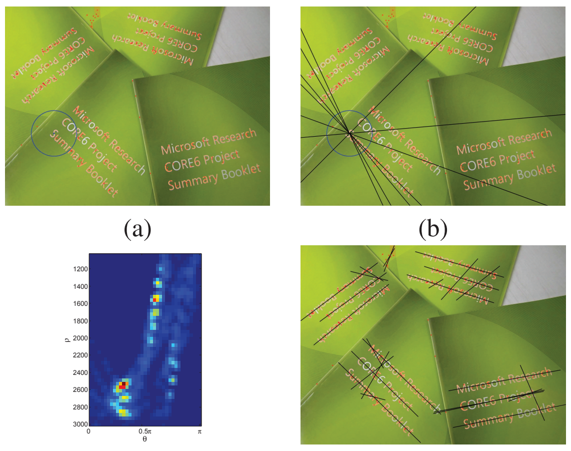


Figure 10: Regions are clustered to form lines of text. A selected region r and the radius of neighbor search (a). All neighboring pairs of the region r and their corresponding text directions (b). Quantized text direction votes in the (ρ, θ) parameter space (c). Final text line clusters (d).

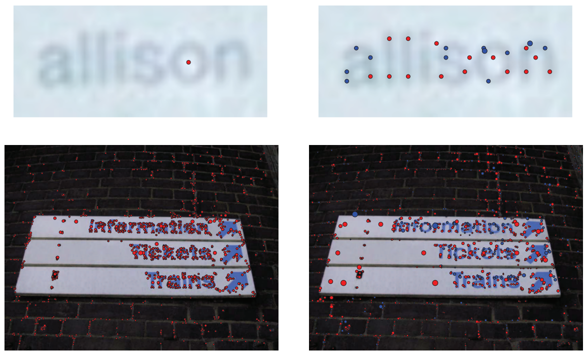


Figure 12: Keypoints detected by the FAST (left) and by the FASText detector (right). The size of the mark is proportional to the scale where the keypoint was detected.

	$\lvert D\rvert$	$\frac{\lvert D\rvert}{\lvert GT\rvert}$	$\lvert FN\rvert$	t [ms]
FAST [21]	608992	105.1	328	29.62
FASText	574713	99.1	**111**	**24.77**

Table 1: Keypoints detected on characters in the ICDAR 2013 dataset. The number of detected keypoints $|D|$, imprecision $\frac{|D|}{|GT|}$, the number of characters without a keypoint $|FN|$ and the average time per image t.

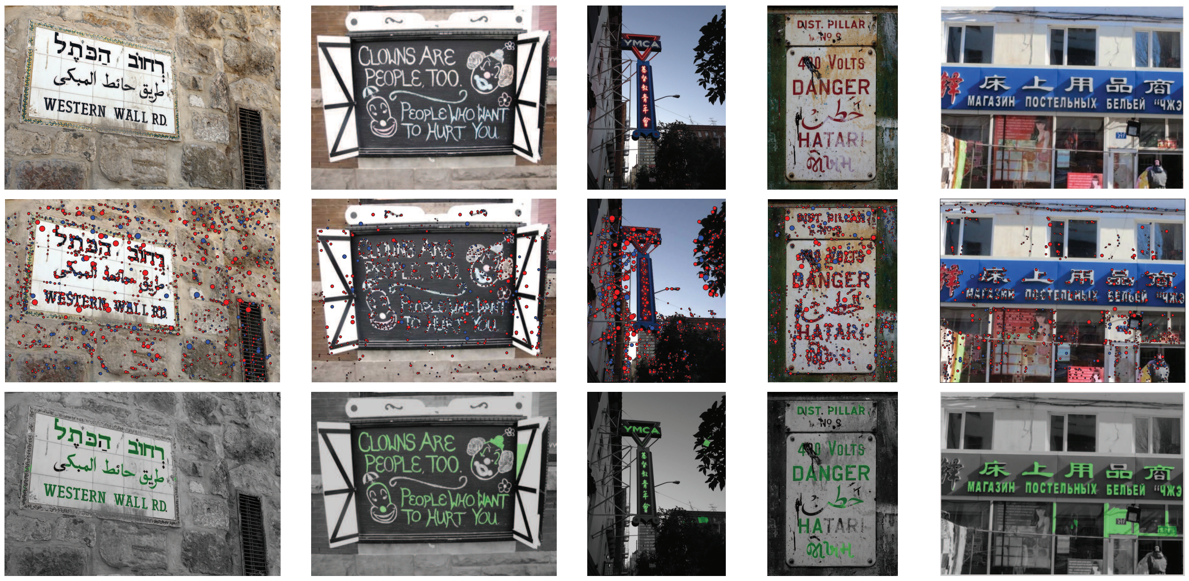


Figure 11: Scene text images with different scripts, fonts and orientations. Source images (top row), detected FASText keypoints (middle row) and resulting text segmentations (bottom row). Best viewed zoomed in color.

	$\lvert D\rvert$	$\frac{\lvert D\rvert}{\lvert GT\rvert}$	$\lvert FN\rvert$	t [ms]
MSER [14]	401972	69.4	1128	328.09
CSER [16]	636729	109.9	657	1068.41
FASText	215325	37.2	857	**82.01**
FASText+AdaBoost	**62394**	**10.8**	1240	122.10

Table 2: Detected region segmentations in the ICDAR 2013 dataset. The number of region segmentations $|D|$, imprecision $\frac{|D|}{|GT|}$, the number of characters without a valid region segmentation $|FN|$ and the average time per image t.

pared with the region segmentations produced by the proposed detector on all images in the ICDAR dataset. Both the standalone FASText detector and the FASText detector with the subsequent classification stage (see Section 3.3) are included, whilst the FAST detector is not included as it does not produce region segmentations. A character is considered as detected by a detector, if the detector produces a region segmentation, whose bounding-box overlap with the character ground truth bounding-box is above 60%. If no such region exists, a character is considered missed.

The FASText detector produces 2 times less region segmentations and still detects 25% more characters than the commonly exploited MSER detector [14] and at the same time it is 4 times faster. The FASText detector with the proposed subsequent classification phase produces approximately 7 times less region segmentations and is almost 3 times faster than the MSER detector. Timings are given for a 2.4GHz PC using a single thread, all stages are included.

4.2. Text Localization and Recognition

In order to evaluate scene text localization ability of the proposed detector with the state-of-the-art text localization methods, which are mostly built around the MSER detector [14], we have replaced the initial stages (MSER detection, character classification and the text line formation) of the MSER-based pipeline [18] with the proposed method. The resulting experimental pipeline thus consists of the FASText detector with the proposed region classification and the text clustering, followed by the local iterative segmentation refinement and character recognition adopted from the original pipeline.

First, the pipeline was evaluated on the most cited ICDAR 2013 Robust Reading Dataset [11] which consists of 1189 words and 6393 letters in 255 images. There are many 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 view is typically centered around the text area (see examples in Figure 13).

Using the same evaluation protocol as the ICDAR 2013 Robust Reading competition [11], the text localization accuracy compares favorably with the state-of-the-art methods (see Table 3), whilst the proposed method is significantly faster - in text localization (t_l), the proposed pipeline is 3 times faster than the best method.

Let us note that the text localization recall and precision of all the listed methods should be interpreted with caution, as the standard evaluation protocol has known issues - for instance, failing to detect 3 characters in a word of 6 characters can still result in a 100% recall, whereas detecting all

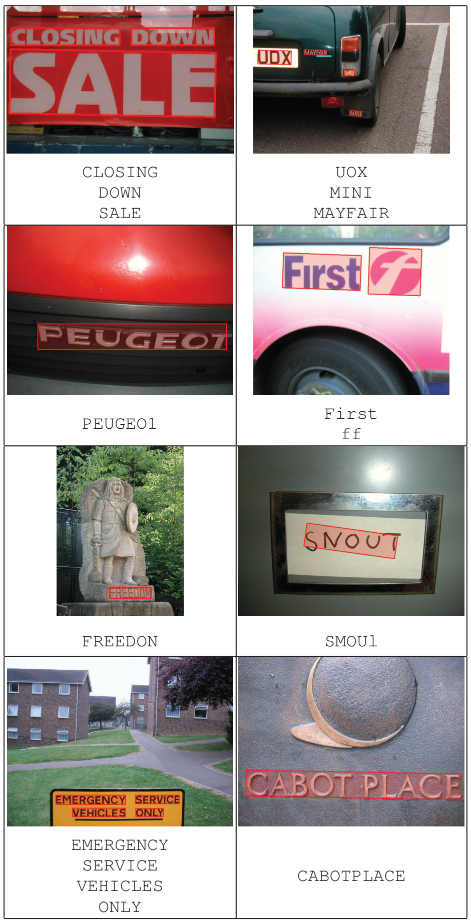


Figure 13: Text localization and recognition examples on the ICDAR 2013 dataset.

method	R	P	F	t_l	t_r	$\mathbf{t}$
proposed pipeline	69.3	84.0	76.8	**0.15**	0.4	0.55
Neumann and Matas [18]	72.4	81.8	77.1	0.4	0.4	0.8
Zamberletti et al. [31]	70.0	85.6	77.0	0.75	N/A	N/A
Yin et al. [30]	68.3	86.3	76.2	0.43	N/A	N/A
ICDAR 2013 winner [11]	66.4	88.5	75.9	?	?	?

Table 3: Text localization results and average processing times on the ICDAR 2013 dataset. Recall R, precision P, f-measure F, localization time t_l, recognition time t_r and the total processing time $\mathbf{t}$ (in seconds).

characters but missing the dot above the character “i” can result in a 0% recall.

In the second experiment, the pipeline (without the text recognition stage) was qualitatively evaluated on a dataset with a wide variety of scripts, fonts and text orientations. As demonstrated in the Figure 11, the FASText keypoint detector is able to detect many different scripts, fonts and orientations and it together with the subsequent steps can be easily exploited to produce text segmentations.

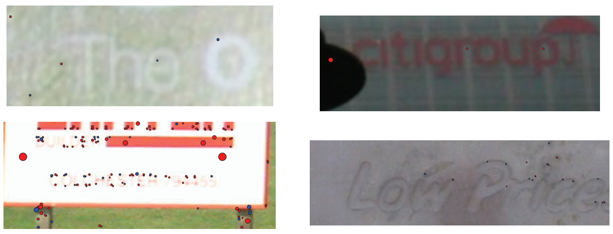


Figure 14: FASText detection and segmentation failures caused by low image contrast and by nonexistence of a threshold in the intensity channel. Best viewed in color.

5. Conclusion

A novel easy-to-implement stroke detector was proposed. The detector is significantly faster and produces significantly less false detections than the commonly used MSER detector. The detector efficiently produces character strokes segmentations, which are exploited in a subsequent classification phase based on features effectively calculated as part of the segmentation process. Additionally, an efficient text clustering algorithm based on text direction voting is proposed. The text detection and clustering is scale- and rotation-invariant and supports wide variety of scripts and fonts.

The proposed FASText detector produces 2 times less region segmentations and still detects 25% more characters than the commonly exploited MSER [14] detector and it is 4 times faster, despite the fact it was implemented only in a single thread and the code was not optimized. The FASText detector with the proposed subsequent classification phase produces approximately 7 times less region segmentations and is almost 3 times faster than the MSER detector. When the proposed detector is plugged into a scene text localization and recognition pipeline, the pipeline maintains state-of-the-art text localization results whilst reducing the processing time.

The main failure reasons for failure are low image contrast, nonexistence of a threshold in the intensity channel and characters very close to each other (see Figure 14).

Acknowledgment

The authors were supported by the Czech Science Foundation Project GACR P103/12/G084, the Technology Agency of the Czech Republic research program TE01020415 (V3C – Visual Computing Competence Center) and the CTU student grant SGS15/155/OHK3/2T/13. Lukas would also like to acknowledge the support of the Google PhD Fellowship and the Google Research Award.

References

[1] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. 6

[2] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. PhotoOCR: reading text in uncontrolled conditions. ICCV, 2013. 2

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986. 2

[4] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. *VISAPP*, 05-08 February 2009, 2009. 2

[5] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR 2010*, pages 2963 –2970. 2

[6] J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000. 4

[7] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced mser trees. In *Computer Vision–ECCV 2014*, pages 497–511. Springer, 2014. 2

[8] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *Computer Vision–ECCV 2014*, pages 512–528. Springer, 2014. 2

[9] L. Jung-Jin, P.-H. Lee, S.-W. Lee, A. Yuille, and C. Koch. Adaboost for text detection in natural scene. In *ICDAR 2011*, pages 429–434, 2011. 2

[10] L. Kang, Y. Li, and D. Doermann. Orientation robust text line detection in natural images. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 4034–4041. IEEE, 2014. 2

[11] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, L. P. de las Heras, et al. ICDAR 2013 robust reading competition. In *ICDAR 2013*, pages 1484–1493. IEEE, 2013. 2, 3, 7, 8

[12] T. Kobayashi, M. Iwamura, T. Matsuda, and K. Kise. An anytime algorithm for camera-based character recognition. In *ICDAR 2013*, pages 1172–1176, Aug. 2013. 2

[13] C.-Y. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu. Region-based discriminative feature pooling for scene text recognition. In *CVPR 2014*, pages 4050–4057. IEEE, 2014. 2

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22:761–767, 2004. 2, 7, 8

[15] A. Mishra, K. Alahari, and C. V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *CVPR 2012*, pages 2687 –2694, june 2012. 2

[16] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *CVPR 2012*, pages 3538 –3545, 6 2012. 2, 7

[17] L. Neumann and J. Matas. On combining multiple segmentations in scene text recognition. In *ICDAR 2013*, pages 523–527, California, US, August 2013. IEEE. 2, 4

[18] L. Neumann and J. Matas. Efficient scene text localization and recognition with local character refinement. In *ICDAR 2015*, pages 746–750. 2015. 2, 4, 5, 7, 8

[19] D. Nistér and H. Stewénius. Linear time maximally stable extremal regions. In *Computer Vision–ECCV 2008*, pages 183–196. 2008. 2

[20] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *ECCV 2012*, pages 752–765. Springer, 2012. 2

[21] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE, 2005. 2, 4, 6

[22] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010. 2

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011. 4

[24] C. Shi, C. Wang, B. Xiao, Y. Zhang, and S. Gao. Scene text detection using graph model built upon maximally stable extremal regions. *PR*, 34(2):107 – 116, 2013. 2

[25] A. Treuenfels. An efficient flood visit algorithm. *C/C++ Users Journal*, 12(8):39–62, 1994. 4

[26] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV 2011*, 2011. 2

[27] J. J. Weinman, E. Learned-Miller, and A. R. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *TPAMI*, 31(10):1733–1746, 2009. 2

[28] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *CVPR 2012*, pages 1083 –1090, june 2012. 2

[29] C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *CVPR 2014*, pages 4042–4049, 2014. 2

[30] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao. Robust text detection in natural scene images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(5):970–983, May 2014. 2, 4, 8

[31] A. Zamberletti, I. Gallo, and L. Noce. Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions. In *1st International Workshop on Robust Reading, IWRR 2014 (in conjunction with ACCV 2014), Singapore, November 2, 2014*, 2014. 2, 8