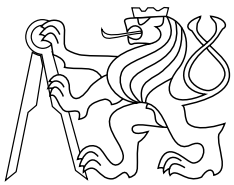
CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

DOCTORAL THESIS

ISSN 1213-2365

Scene text localization and recognition in images and videos

A Doctoral Thesis presented to the Faculty of the Electrical Engineering of the Czech Technical University in Prague in fulllment of the requirements for the Ph.D. Degree in Study Programme No. P2612 Electrical Engineering and Information Technology, branch No. 3902V035 - Articial Intelligence and Biocybernetics, by

Ing. Lukáš Neumann

Thesis Advisor
Prof. Ing. Jiří Matas, Ph.D.

May 2017

Available at
http://cmp.felk.cvut.cz/~neumalu1/thesis.pdf

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Scene text localization and recognition in images and videos

Ing. Lukáš Neumann

May 2017

Abstract

Scene Text Localization and Recognition methods find all areas in an image or a video that would be considered as text by a human, mark boundaries of the areas and output a sequence of characters associated with its content. They are used to process images and videos taken by a digital camera or a mobile phone and to "read" the content of each text area into a digital format, typically a list of Unicode character sequences, that can be processed in further applications.

Three different methods for Scene Text Localization and Recognition were proposed in the course of the research, each one advancing the state of the art and improving the accuracy. The first method detects individual characters as Extremal Regions (ER), where the probability of each ER being a character is estimated using novel features with $O(1)$ complexity and only ERs with locally maximal probability are selected across several image projections for the second stage, where the classification is improved using more computationally expensive features. The method was the first published method to address the complete problem of scene text localization and recognition as a whole - all previous work in the literature focused solely on different subproblems.

Secondly, a novel easy-to-implement stroke detector was proposed. The detector is significantly faster and produces significantly less false detections than the commonly used ER 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, which as well as the previous stages is scale- and rotation- invariant and supports wide variety of scripts and fonts.

The third method exploits a deep-learning model, which is trained for both text detection and recognition in a single trainable pipeline. The method localizes and recognizes text in an image in a single feed-forward pass, it is trained purely on synthetic data so it does not require obtaining expensive human annotations for training and it achieves state-of-the-art accuracy in the end-to-end text recognition on two standard datasets, whilst being an order of magnitude faster than the previous methods - the whole pipeline runs at 10 frames per second.

Abstrakt

Čtení textu patří mezi aktuální a otevřené problémy v oblasti počítačového vidění a umělé inteligence. Metody pro lokalizaci a rozpoznávání textu v obraze automaticky vyhledávají všechny textové oblasti ve fotografii nebo videu, nachází jejich pozici a rozpoznávají jejich textový obsah jako posloupnost znaků. Tyto metody zpracovávají obrázky a videa pořízené běžným fotoaparátem nebo mobilním telefonem a ,,čtou“ obsah každé nalezené textové oblasti do digitálního formátu, který může být snadno použit v navazujících aplikacích.

V průběhu výzkumu byly navrženy tři metody pro lokalizaci a rozpoznávání textu, přičemž každá metoda posunula stav poznání v této oblasti a zlepšila celkovou přesnost rozpoznávání. První metoda detekuje znaky jako Extremální Regiony (ER), kde pravděpodobnost, že daný region představuje znak, je efektivně odhadována pomocí příznaků s konstantní výpočetní složitostí a pouze regiony s lokálně maximální pravděpodobností jsou vybrány pro druhou fázi algoritmu, kde je klasifikace vylepšena již pomocí více výpočetně náročnějších příznaků. Metoda je celosvětově první publikovanou metodou, která se zaměřuje na kompletní problém lokalizace i rozpoznání textu – všechny dříve publikované metody se zaměřovaly pouze na samostatné podproblémy.

Druhá metoda představuje nově navržený detektor znaků, který je významně rychlejší a který generuje výrazně méně chybných detekcí než standardně používaný ER detektor. Tento detektor zároveň generuje segmentace jednotlivých znaků, které jsou následně rovnou využity při rozpoznávání. Jednotlivé znaky jsou spojeny ve slova pomocí efektivního algoritmu pro shlukování textu, který, stejně jako všechny předešlé kroky, je invariantní vůči zvětšení a rotaci a který podporuje velkou škálu jazyků a fontů.

Třetí metoda využívá novou architekturu hluboké neuronové sítě natrénované pro lokalizaci i rozpoznávání textu. Trénování využívá syntetická data, čímž odpadá nutnost drahého a zdlouhavého ručního anotování. Tato metoda lokalizuje a rozpoznává veškerý text v obrázku v rámci jediného průchodu a dosahuje celosvětově nejlepší přesnosti na dvou standardních datových sadách. Zároveň je o jeden řád rychlejší než předchozí metody – algoritmus lokalizuje a rozpoznává text rychlostí 10 snímků za sekundu.

Acknowledgement

I am especially grateful to my advisor Jiří Matas for introducing me to computer vision and for attracting my interest towards science. I would not be able to pursue my PhD degree without his support, ideas and motivation. I would also like to thank Michal Bušta for his insights and for helping me certain aspects of the code and ultimately taking over the code maintenance and implementation.

I would like to thank my CMP colleagues Ondra Chum, Michal Perdoch, Tomáš Vojíř, Andrej Mikulík and Jan Šochman for their comments, ideas and for helping me through my PhD studies, and I would also like to thank Yash Patel for reviewing the text of the thesis on such a short notice.

I also would like to acknowledge the support of the Google PhD Fellowship and the Google Research Award, which made my PhD studies possible.

Last but not least, I am grateful to my loving wife Martina (and to my son Kryštof, although he did not have much say about all this) for supporting me in the pursue of my academic career and for putting up with the late nights, weekend work and the travel.

Contents

1 Introduction

1.1 The Objective

Methods for *Scene Text Localization and Recognition* find all areas in an image or a video that would be considered as text by a human, mark boundaries of the text areas and output a sequence of characters associated with its content (see Figure 1.1). They are used to process images and videos taken by a digital camera or a mobile phone and to "read" the content of each text area into a digital format, typically a list of Unicode character sequences, that can be processed in further applications.

Figure 1.1 The *Scene Text Localization and Recognition* problem. All legible areas in an image with text are output as sequences of characters, which can be easily processed by a computer

Scene Text Localization and Recognition is an open and complex problem of computer vision, because text typically occupies only a small fraction of the image, it has non-uniform background, it suffers from noise, blur, occlusions and reflections and perspective effects need to be taken into account. Moreover real-world texts are often short snippets written in different fonts and languages, text alignment does not follow strict rules of printed documents and many words are proper names which prevents an effective use of a dictionary. All the aforementioned factors make the *Scene Text Localization and Recognition* problem significantly harder and standard printed document recognition (OCR) methods and applications cannot be used [99, 37] (see Figure 1.2).

Standard computer vision object detection methods cannot be directly exploited for text either, because their output is an object class and position (e.g. a "plane" or a "cow"), but detecting "text" as a class without its content is not very useful (although it is already quite a challenging problem, see Section 2.1). One could try to overcome this issue by representing each possible character sequence as its own separate class, but such an approach can only be applied when all the sequences (words) to be detected are known in advance [134, 49]. Moreover, this approach does not generalize to arbitrary text detection and recognition, because the space of text content is exponential - given an alphabet $\mathcal{A}$ and a maximal text length L, there can be up to $\mathcal{A}^L$ different text classes.

The goal of the thesis is to advance the state of the art by introducing methods specialized in text localization and recognition, without constraining the methods to a particular domain of text, a font, a script, or a type of scene the text is captured

Figure 1.2 The difference between printed document OCR (left) and *Scene Text Localization and Recognition* (right)

Figure 1.3 An aid for visually impaired people. Any detected text is automatically read out loud using a speech synthesizer

in. Methods for scene text localization and recognition are very useful, because text captures important information and cues about the scene and they are also an important component of general image understanding, as text is one of the most frequent classes in the real world. For example, in the COCO dataset for object detection, there are over 173 000 text instances which makes it the most frequent class [129].

There are also many possible applications of scene text localization and recognition, with some particular examples listed below.

Helping visually impaired. It is hard or even impossible for visually impaired or elderly people to read text, but text often provides crucial information, which sometimes cannot even be possibly obtained in any other way - for example, the only way to distinguish between two different types of drugs can be just by reading the label on its packaging. Linking the scene text recognition algorithm with a speech synthesiser and deploying it on a mobile phone gives an easy-to-use and affordable assistive tool, which is able to help by automatically detecting any text in the video stream and reading it out loud (see Figure 1.3). The method must not only be accurate but also efficient, because text has to be automatically detected directly from the live camera stream, as it is not reasonable to expect that visually impaired people would take pictures of text.

Urban navigation. Navigation in urban environments or inside buildings cannot rely on position information from GPS, so alternative methods have to be exploited. One of the options is using business labels and other signs as one of the cues to determine the position, and a scene text recognition algorithm is therefore required to transform the surrounding textual information into a form which can be further processed. The same applies for self-driving cars, where GPS and maps are available, but they may not include temporary signs, which are often text-based.

Automated translation. One of the issues in machine translation applications is the user input. Traditionally, a user has to type in the word/phrase that should be translated, however this can be quite slow, error-prone and sometimes close to impossible, if the user is not familiar with the alphabet of the text which he is trying to translate (for example a European tourist in China trying to translate Chinese text). Automated scene text recognition overcomes the issue by automatically recognizing text in a picture taken by the user, possibly with a guidance from the user to select which areas of the image to translate, effectively eliminating the need for any manual input (see Section 7.1.2).

Indexing and searching image databases by textual content. An arbitrary image or video databases can be automatically indexed by its textual so that user can search by text queries, which is the most common input to search engines. A user could find his favorite restaurant in a database such as Google Street View using just a restaurant's name, he could find a movie poster on Flickr or he could find an interview with a certain person just by searching a TV news video archive.

1.2 Contributions

The main contributions of this thesis towards extending the state of the art in scene text localization and recognition are

1. An *end-to-end method* joining both text localization and text recognition into a single pipeline is proposed (see Chapter 4). Our method [88] was the first one to address this problem as a whole, thus allowing practical applications - all previous work focused solely on different subproblems.

2. Character detection is posed as an efficient sequential selection from the set of *Extremal Regions* (see Section 4.1.1). The ER detector is robust to blur, illumination, color and texture variation and handles low-contrast text better than the standard MSER detector [80]. The newly introduced features, which are *incrementally computed* (see Section 4.1.2), allow the method to run in real time [91], which was a significant improvement over previous methods.

3. We propose a novel easy-to-implement stroke detector (see Section 5.1), 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 and then exploited to obtain stroke segmentations.

4. The concept of *text fragments* is introduced (see Section 5.3), where a *text fragment* can be a single character, a group of characters, a whole word or a part of a character, which allows to drop the common assumption of region-based methods that one region corresponds to a single character. A novel *Character Strokes Area* is introduced, effectively approximating "strokeness" of text fragments, which plays an important role in the discrimination between text and a background clutter.

5. A *lexicon-free* text recognition method trained purely on *synthetic data* is presented (see Section 4.2.1). The method does not rely on any prior knowledge of text to be detected (lexicon), unlike the majority of the methods in the literature (see Section 2.5). The proposed method also shown that synthetic data can be

successfully exploited for training scene text recognition, which is an idea that has been successfully built upon and extended by several authors [45, 41].

6. The proposed method is amongst the first ones to detect and recognize text in scene videos. Combined with the standard FoT tracker [131], it set a baseline for *end-to-end video text recognition* (see Section 4.3.4) and it outperformed all other participants. To our knowledge, no other method has published better results in the end-to-end video text recognition.

7. A deep model which is trained for both text detection and recognition in a single learning framework is presented (see Chapter 6). The model is trained for both text detection and recognition in a single end-to-end pass and it outperforms the combination of state-of-the-art localization and state-of-the-art recognition methods.

1.3 Publications

This thesis build on the result previously published in the following publications

- **A method for text localization and detection, ACCV 2010** [88] - the paper introduces a first version of the end-to-end text detection and localization pipeline, initially based on the MSER [80] detector.

- **Real-Time Scene Text Localization and Recognition, CVPR 2012** [91] - an efficient Extremal Region classifier, detailed in Section 4.1.3, is introduced, allowing the whole text detection and recognition pipeline to run in real time.

- **On Combining Multiple Segmentations in Scene Text Recognition, ICDAR 2013** [92] - the paper describes the sequence selection algorithm to generate the final output, detailed in Section 4.2.2. The paper received the **ICDAR 2013 Best Student Paper Award**.

- **Efficient Scene Text Localization and Recognition with Local Character Refinement, ICDAR 2015** [94] - the paper introduces the concept of text fragments and the Character Strokes Area feature, described in Section 5.3.1. The paper received the **ICDAR 2015 Best Paper Award**.

- **Real-time Lexicon-free Scene Text Localization and Recognition, TPAMI** [95] - the journal paper describes the text detection and recognition pipeline of Chapter 4

- **FASText: Efficient Unconstrained Scene Text Detector, ICCV 2015** [17] - the paper introduces the FASText scene text detector, described in Chapter 5.

- **Deep TextSpotter: An End-to-End Trainable Scene Text Localization and Recognition Framework** [18] - the paper presents the Single Shot Text Detection and Recognition pipeline, described in Chapter 6.

The following publications were not included in the thesis, in order to keep the thesis more focused and easier to follow

- **Estimating hidden parameters for text localization and recognition, CVWW 2011** [89]
- **Text Localization in Real-world Images using Efficiently Pruned Exhaustive Search, ICDAR 2011** [90]
- **Scene Text Localization and Recognition with Oriented Stroke Detection, ICCV 2013** [93]
- **ICDAR 2015 competition on robust reading, ICDAR 2015** [52]
- **COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images, WACV 2016** [129]

1.4 Authorship

I hereby certify that the results presented in this thesis were achieved during my own research, in cooperation with my thesis advisor Jiří Matas. The work presented in Chapter 5, Chapter 6 and Section 7.1.2 is a joint work with Michal Bušta, who contributed to the ideas and experiments presented in these sections.

2 Related Work

In the research community, the problem of Scene Text Recognition was originally broken down into several sub-tasks, because a complete scene text recognition problem was deemed as too complex to solve. The split into these subtasks was also motivated by the traditional printed document OCR pipeline, which had been already successful in printed text recognition [72], so the idea was to only replace the initial stages of the printed document pipeline to make it work for scene text as well.

2.1 Text Localization

Historically, the main focus of the scene text research community was given to the text localization problem, because it is the first stage of any scene text recognition pipeline and it was shown to be already very challenging, even without a subsequent recognition phase.

In the *text localization* task, given an image $\mathbf{I}$ a method must find all rectangular areas (x, y, w, h), which a human (annotator) would consider as text

$$\mathbf{I} \rightarrow \left\{(x_i, y_i, w_i, h_i)\right\}^n \quad x_i, y_i, w_i, h_i \in \mathbb{N} \tag{2.1}$$

where x and y denote top-left corner co-ordinates and w (h) denote the width (resp. the height) of the rectangle.

The quality of the output is measured by comparing the set of the detected rectangles $D = \{D_i\} = \left\{(x_i, y_i, w_i, h_i)\right\}$ with the set of rectangles provided by a human annotator $G = \{G_i\}$. The standard evaluation protocol, as proposed by Wolf and Jolion [139], defines the "recall" r and the "precision" p as

$$r = \frac{\sum_{i=1}^{n} m(D_i, G)}{|G|} \tag{2.2}$$

$$p = \frac{\sum_{i=1}^{n} m(G_i, D)}{|D|} \tag{2.3}$$

where $m(r, R)$ is a "match function" whose value depends on whether the rectangle r is matched to one or more rectangles in the rectangle set R.

In the ICDAR Robust Reading competition [118, 53, 52], text is annotated on the word level[1], so methods are expected to output bounding boxes of individual words, but detecting multiple words with a single rectangle or detecting a single word with multiple rectangles should not be penalized as heavily as not detecting the word at all. This requirement therefore resulted into the following match function definition

$$m(r, R) = \begin{cases} 1 & \text{if } \exists! t \in R : \sigma(r, t) \geq 0.8 \wedge \tau(r, t) \geq 0.4 \textbf{ (one-to-one match)} \\ 0.8 & \text{if } \exists S_0 \subset R : \sum_{s_i \in S_0} \sigma(r, s_i) \geq 0.8 \wedge \forall_{s_i \in S_0} \tau(r, s_i) \geq 0.4 \textbf{ (one-to-many match)} \\ 0.8 & \text{if } \exists S_0 \subset R : \forall_{s_i \in S_0} \sigma(r, s_i) \geq 0.8 \wedge \sum_{s_i \in S_0} \tau(r, s_i) \geq 0.4 \textbf{ (many-to-one match)} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

[1] Even though the competition and the data set works solely with word bounding boxes, the definition of *word* is not given. For example, it is not clear whether a phone number is single or multiple words, or whether hyphens should break a word into two

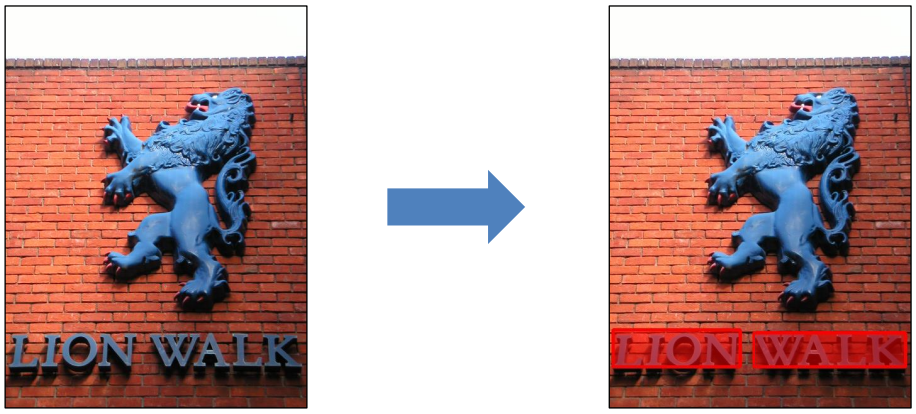


Figure 2.1 Text localization output example. Given given an image **I** (left) two rectangular areas $(4, 522, 207, 583)$, $(230, 521, 471, 582)$ are detected (right).

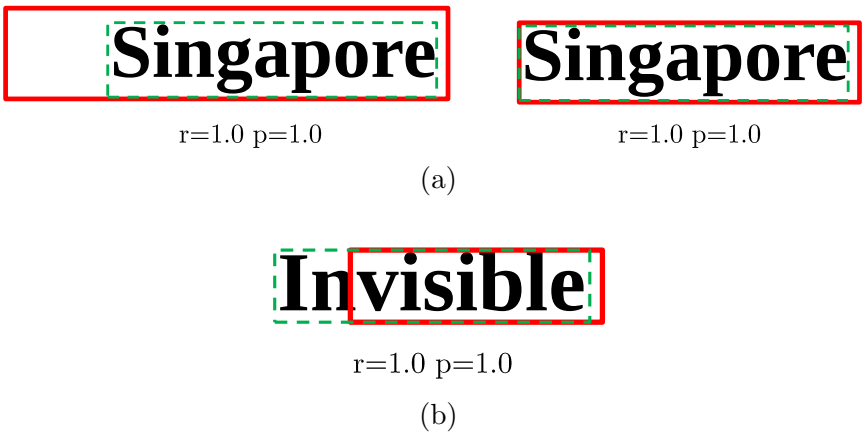


Figure 2.2 The standard text localization protocol [139] does not penalize loose bounding boxes (a) or partial detections which could change meaning of the word (b) - all detections illustrated above would achieve 100% recall and 100% precision. Detections denoted red, ground truth in green

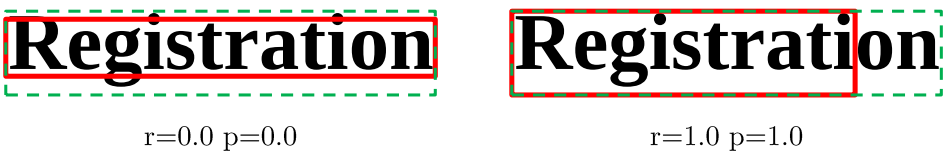


Figure 2.3 The text localization protocol [139] penalizes errors more in the vertical direction - the detection on the left could still be correctly recognized in the OCR stage, yet it achieves 0% recall and precision, whilst the detection on the right achieves 100% recall and precision, but it is impossible to get a correct recognition since a part of the word is missed completely. Detections denoted red, ground truth in green

where σ and τ denote rectangle "recall" and "precision" area ratio

$$\sigma(x, y) = \frac{\text{area}(x \cap y)}{\text{area}(y)} \tag{2.5}$$

$$\tau(x, y) = \frac{\text{area}(x \cap y)}{\text{area}(x)} \tag{2.6}$$

Although this protocol has been heavily used in the literature as well as in the recent Robust Reading competitions [118, 53, 52], there are several inherent problems in it. The protocol motivates methods to over-estimate the text area, since the precision threshold for successful detection is quite permissive ($\tau(r, t) \geq 0.4$) and therefore the protocol doesn't penalize bounding boxes which are only loosely around text (see Figure 2.2a).

Equally, the required area overlap of 80% with the ground truth ($\sigma(r, t) \geq 0.8$) does not guarantee that the whole word is detected, and the undetected 20% of the word can completely change the word's meaning (see Figure 2.2b).

The dependence purely on the area ratio without any regard for the direction of text implies that for longer words the protocol penalizes errors more in the vertical direction, which is completely counter-intuitive - a tight word bounding box in the vertical direction can be interpreted as not a match, even though all characters inside it are still readable, whereas not detecting a part of the word in the horizontal direction is still considered a valid match (see Figure 2.3), as in the previous case.

These limitations of the standard protocol should be taken into account when interpreting text localization performance and when making comparison between text localization methods, since even a method which would achieve 100% recall and 100% precision could still be impractical for subsequent scene text recognition, and vice-versa, a method with lower localization recall and precision can still achieve better overall recognition performance.

Localizing text in an image can be a computationally very expensive task as generally any of the 2^N subsets can correspond to text (where N is the number of pixels). Existing methods for general text localization can be categorized into two major groups - methods based on a sliding window (Section 2.1.1) and methods based on connected components (Section 2.1.2). The CNN-based methods (Section 2.1.3) also in principle fall into the first group.

2.1.1 Sliding-window Methods

The "sliding-window" methods use a window which is moved over the image and the presence of text is estimated on the basis of local image features, which is an approach successfully applied in generic object detection [130]. While these methods are generally more robust to noise in the image as the features are aggregated over a larger image area, their computation complexity is high because of the need to search with many rectangles of different sizes, aspect ratios, rotations and perspective distortions, which is an effect that does not occur in generic object detection.

Chen and Yuille [20] use AdaBoost classifier [116] combining mean intensity features, intensity variance features, derivative features, histogram features and features based on edge linking. A variant of Niblack's adaptive binarization algorithm [97] is then used to obtain segmentation. The method is computationally expensive (it takes 3 seconds to process a 2MPix image), it requires manual annotation of many subwindows for training purposes (see Figure 2.4) and its localization performance is not clear (standard evaluation protocol was not used).

Figure 2.4 Positive training data samples used for sliding-window classifier training. Image taken from [20]

The method was improved by Pan et al. [103] by incorporating a combination Histogram Of Gradient (HOG) and multi-scale Local Binary Pattern feature in the text detection stage. Furthermore, a Markov Random Field (MRF) is employed to to group segmented characters into words, in contrast to the heuristic rules applied in [20]. The method claims better localization performance than the winner of ICDAR 2005 Text Locating Competition [76], which is not fully accurate because different evaluation units were used (words vs. lines of text). Additionally, the method suffers from high computational complexity (average processing time 1.5s on a 1MPix image).

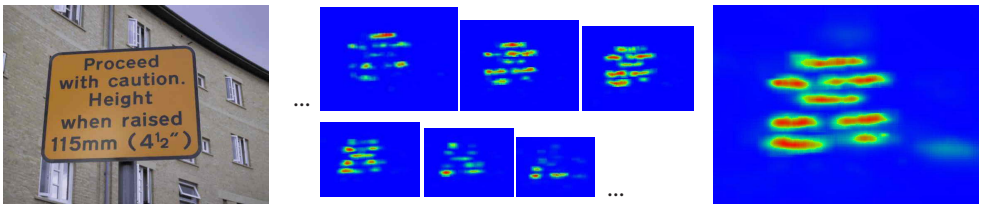
More recently, Lee et al. [64] further improved the approach by incorporating more discriminative but also more computationally expensive features, which slightly improved text localization performance, but significantly slowed down the method (processing time is several minutes per image).

Coates et al. [23] use unsupervised machine-learning techniques independently for character detection and recognition. A 32-by-32 pixel window is shifted over the image in multiple scales and each patch is classified using a linear SVM classifier as text or non-text. Features used by the classifier are generated automatically in the training stage using a variant of the K-means algorithm. Cropped characters are recognized by resizing them into a fixed 32-by-32 pixel window and applying the same process. The method however does not provide end-to-end text recognition as the characters are cropped manually.

Zhu and Zanibbi [149] build on top of this work by introducing multi-stage generation and validation of character detections using convolutional, geometric and contextual features, and by exploiting a confidence-weighted AdaBoost classifier to obtain text/non-text saliency map.

2.1.2 Region-based Methods

Region-based methods exploit a bottom-up strategy for text localization. At first, certain local features are calculated for each pixel in the image and then pixels with similar feature values are grouped together using connected component analysis to form characters, assuming low variance of the used feature(s) within a single character. The methods can be scale-invariant and they inherently provide character segmentation, which can be then used in an OCR stage. The biggest drawback is their sensitivity to noise and to low-resolution images, because they require low variance of the local features and a single pixel with different feature value can cause the connected component analysis to fail.



(a) (b) (c)

Figure 2.5 Text confidence map. (a) the original image. (b) text confidence maps for the image pyramid. (c) the text confidence map for the original image. Images taken from [104]

One of the first methods for general character localization was introduced by Ohya et al. [100]. In their method, a local adaptive thresholding in grey-scale images is used to detect candidate regions and regions with sufficient contrast are selected as characters. Li et al. [67] apply thresholding in a quantized color space and they group individual characters into text blocks by simple alignment rules. Both methods assume that characters are upright without any rotation, that the contrast is high and that background is uniform, which may be sufficient for signs or licence plates, but not for general text localization.

Kim et al. [56] combine three independent detection channels (color continuity, edge detection and color variance) to find candidate regions. Candidate regions are grouped into blocks by size and position constraints and each block is then divided into overlapping 16×16 pixel subblocks, which are verified by an SVM classifier [24], exploiting wavelet transform to generate features. If a ratio of subblocks marked as text is higher than a predefined threshold, the block is marked as text. The method is not scale-invariant because of the constant size of subblocks used for classification and the precision is relatively bad because many small text-like patches are detected.

Takahashi and Nakajima [123] use a graph scheme, where vertices represent characters and edges represent a predecessor-successor relation in a block of text. Candidate regions are found using Canny edge detector in the CIELUV color space [21] and regions that pass a set of heuristic constraints are considered vertices of the graph. Edges are created between neighboring vertices and each edge is assigned a weight based on spatial distance, shape and area similarity. Finally, a minimum spanning tree (MST) algorithm is applied and edges with distance or angle over a predefined threshold are removed. The method cannot handle illumination variations in foreground or background and optimal edge weight estimation remains an open question.

Pan et al. [104, 105] create a text confidence map (see Figure 2.5) on a grey-scale image pyramid using a calibrated Waldboost [122] classifier with Histogram Of Gradient (HOG) features. Candidate regions are detected independently on a grey-scale image using Niblack's binarization algorithm [97] and a Conditional Random Field (CRF) [59] is employed to label regions as text or non-text, considering the text confidence as one of the unary features. Then, a simple gradient graph energy minimization approach is applied to form block of texts. The method is computationally expensive because of the image pyramid and the CRF inference and its localization performance cannot be compared to other methods a proprietary evaluation metric was used.

An image operator Stroke Width Transform (SWT) was introduced by Epshtein et al. [29]. The SWT method finds edges using Canny detector [19] and then estimates stroke width for each pixel in the image (see Figure 2.6). Connected component algo-

(a) (b)

Figure 2.6 Stroke Width Transform (SWT). (a) the original image converted to grey-scale. (b) stroke width estimation for each pixel. Images taken from [29]

rithm is then applied to form pixels with similar stroke width into character candidates, which are merged into text blocks using several heuristic rules. The biggest limitation of the method is its dependency on successful edge detection which is likely to fail on blurred or low-contrast images. The method was further improved by Yao et al. [141] where the heuristic rules for character candidate detection and text block formation are replaced by trained classifiers with rotationally-invariant features. A similar edge-based approach with different connected component algorithm is presented in [148].

Many of the text localization methods rely on the Maximally Stable Extremal Regions (MSER) detector [80]. The idea to use MSERs for character detection (and recognition) was first exploited by Donoser et al. [28], where they classified MSERs using cross-correlation with training templates and then used standard search engine to select the most probable text hypothesis and correct any misspellings.

Kang et al. [51] exploits Higher-Order Correlation Clustering [57] to group individual MSERs into text lines and to eliminate non-textual MSERs. Yin et al. [144], the winner of the ICDAR 2013 Robust Reading competition [53], uses a single-link clustering algorithm with a trained distance. Later, the method was improved by using adaptive hierarchical clustering for grouping of MSERs and by supporting multi-oriented text [143]. Huang et al. [44] combine the MSER detector with a CNN classifier to distinguish between text and non-text regions and to split individual MSERs which correspond to multiple characters.

Our work in the text localization and recognition in Chapter 4 also builds on top of the MSER detector and its generalizations [81].

2.1.3 Convolutional Neural Networks

Tian et al. [124] combine the traditional sliding-window approach (see Section 2.1.1) with a CNN classifier [61]. Candidate regions are first detected by sliding window which is classified as text or non-text using a fast cascade boosting algorithm. The text line extraction is then formulated as a min-cost flow problem, where each node is the candidate region (patch) with an associated cost provided by a CNN classifier. The CNN classifier is trained on the image patches obtained by the character detector and it only has three convolutional layers and two fully-connected layers.

He et al. [42] train a CNN to classify 32×32 patches as text/non-text. The training exploits multi-task learning [31] paradigm as the optimized cost function is not simply the text/non-text label, but it also incorporates explicit pixel-level segmentation and the character label. In the testing phase, only regions detected as MSERs [80] in a

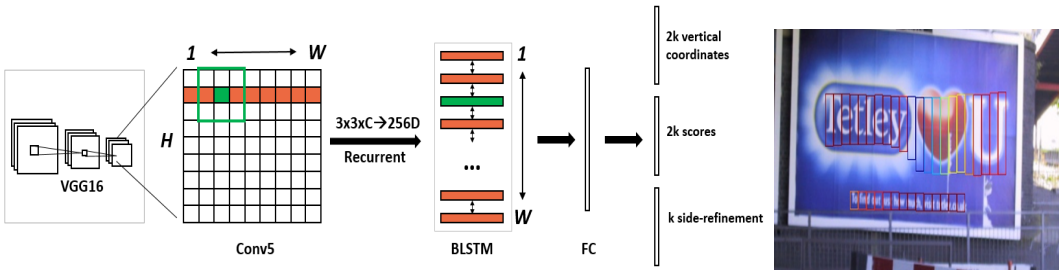


Figure 2.7 The architecture of Tian et al. [125] employs a 3×3 sliding-window on the last convolutional layer as an input to a RNN, which jointly predicts the text/non-text score, the y-axis coordinates and the anchor side-refinement (a). Sample network output, color indicates the text/non-text score (b). Images taken from [125]

contrast-enhanced image are considered for the classification by the CNN, resulting in a text saliency map.

Gupta et al. [41] propose a fully-convolutional regression network, drawing inspiration from the You Look Only Once (YOLO) approach for object detection [108]. An image is divided into a fixed number of cells (14×14 in the highest resolution), where each cell is associated with 7 values directly predicting the position of text: bounding-box location (x, y), bounding-box size (w, h), bounding-box rotation $(\cos\theta, \sin\theta)$ and text/non-text confidence (c). The values are estimated by 7 local translation-invariant predictors built on top of the first 9 convolutional layers of the popular VGG-16 architecture [120], trained on synthetic data (see Section 3.8).

Tian et al. [125] adapt the Region Proposal Networks architecture [110] by horizontally sliding a 3×3 window through on the last convolutional map of the VGG-16 [120] and applying a Recurrent Neural Network to jointly predict the text/non-text score, the y-axis coordinates and the anchor side-refinement (see Figure 2.7). Note that the architecture expects that text is only horizontal, unlike the method of Gupta et al. [41]. Similarly, Liao *et al.* [70] adapt the SSD object detector [74] to detect horizontal bounding boxes.

Ma *et al.* [78] adapt the Faster R-CNN architecture and extend it to detect text of different orientations by adding anchor boxes of 6 hand-crafted rotations and 3 aspects.

2.2 Text Segmentation

The task segmentation task formulation was introduced in the ICDAR 2013 Robust Reading competition [53], in order to obtain more fine-grained evaluation than in the text localization task (Section 2.1) and in order to address some of the aforementioned problems with the text localization protocol. In the *text segmentation* task, a method must label all pixels of an image $\mathbf{I}$ so that pixels which a human would consider as belonging to text are labelled as 1 (and the other pixels are labelled as 0).

$$\mathbf{I} \rightarrow \{0, 1\}^{w \times h} \tag{2.7}$$

where w (h) is the image width (height).

The evaluation protocol, proposed by Clavelli and Karatzas [22], uses "atoms" as units for the evaluation. Atom is a connected component (see Figure 2.9), which typically corresponds to a single character, but can also correspond to a part of a character

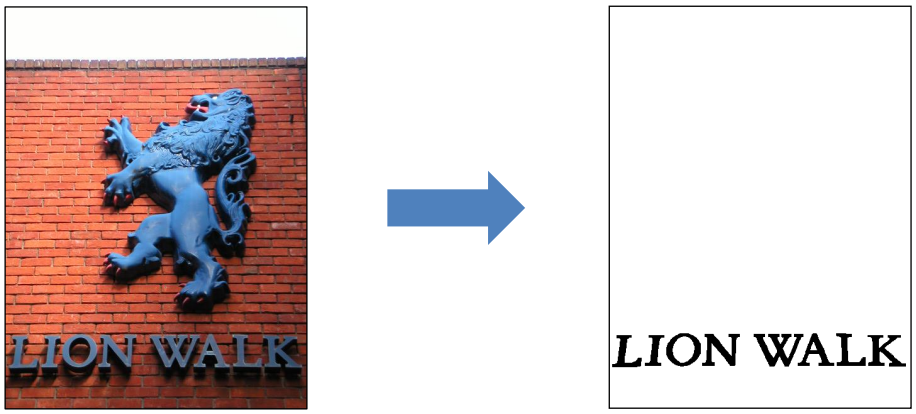


Figure 2.8 Text segmentation. Given given an image **I** (left) pixels labelled as belonging to text are marked black (right).

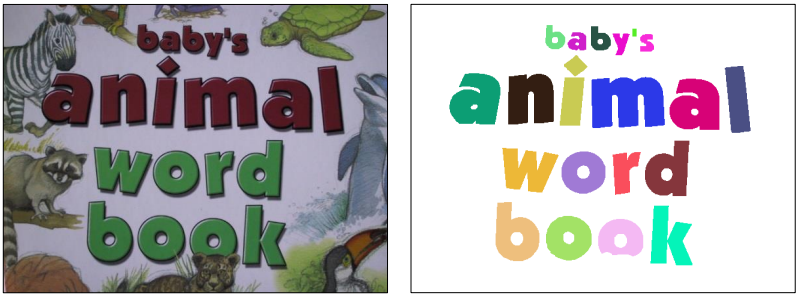


Figure 2.9 Text segmentation ground truth. Each atom denoted by a different color (right).

in case the character is broken into multiple segments, or to multiple characters, when characters are joint together.

Given a method output, each atom in the ground truth is labelled as either *Well-Segmented*, *Merged*, *Broken*, *Broken and Merged* or *Lost*, based on a match heuristics [22]. Additionally, detected components which cannot be matched to any ground truth region are marked as *False Positive*. The heuristics labels an atom as *Well-Segmented*, if at least $T_{\text{min}} = 0.5$ of the ground truth skeleton pixels are part of the atom (*Minimal Coverage* condition), and if none of the atom pixels are further than $T_{\text{max}} = \min(5, 0.9s_w)$ from the ground truth edge (*Maximal Coverage* condition), where s_w denotes ground truth stroke width.

The image recall r precision p are then calculated as

$$r = \frac{|\text{WS}|}{|G|} \tag{2.8}$$

$$p = \frac{|\text{WS}|}{|D|} \tag{2.9}$$

where WS are the *Well-Segmented* atoms, G are the ground truth atoms and D are the detected atoms.

The main drawback of the text segmentation task (and its evaluation protocol) is the dependency on pixel-level comparison, which may not always correspond to the quality one would obtain from a subsequent OCR stage. For instance, missing 30% of pixels

Figure 2.10 The text segmentation protocol [22] may not always provide good clues whether characters could be successfully processed by a subsequent OCR stage - source image (left) and sample segmentation output (right). Well-Segmented atoms in green, Broken atoms in orange

Figure 2.11 Toggle Mapping morphological segmentation [32] sample results, each connected component denoted by a different color. Images taken from [32]

(missing 30% of the skeleton) can significantly change the OCR class as for example the character "O" can change to "U", yet the evaluation protocol would still label the atom as Well-Segmented (see Figure 2.10). On contrary, missing 60% of pixels when compared to the ground truth may still leave the OCR label intact.

Moreover the task segmentation task enforces certain pipeline architectures (architectures similar to traditional printed document OCR), but the segmentation step may be successfully omitted in the end-to-end setup (see Section 2.5).

All the methods listed in the Section 2.1.2 inherently provide some form of text segmentation, due to their reliance on connected components. There are however several methods which focus purely on text segmentation. Fabrizio et al. [32] use the *Toggle Mapping* morphological operator [117], which is a generic operator which maps a function into a set of n functions. In their method, the image (function) is mapped to a set of 2 functions (morphological erosion and morphological dilatation) and the mapping is further thresholded by requiring a minimal contrast to eliminate noise in homogeneous regions.

Kumar and Ramakrishnan [58] employ Otsu binarization [101] in each of the RGB channels independently to find connected components. Then, each connected component in its respective color plane is classified as text or background based on its thinned representation and text components are formed into text lines by a horizontal clustering algorithm, which spans all three color planes.

Mancas-Thillou and Gosselin [79] study cropped word binarization by clustering in RGB color space using Euclidian and Cosine distance. Kim et al. [55] exploit user interaction on a mobile device to find initial location of text and then color clustering in HCL color space [115] is used to find initial candidate regions. The regions are then expanded in horizontal direction using a set of heuristic rules to obtain blocks of text.

Let us note that in the last text segmentation competition [53], all methods specialized in text segmentation [32, 58] were outperformed by a generic text localization method [144], where the character segmentations are simply the output of the MSER detector [80].

2.3 Cropped Character Recognition

The *cropped character recognition* is the simplest text recognition task formulation - given an image of a character, a method outputs a single character of the output alphabet $\mathcal{A}$:

$$\mathbf{I} \rightarrow c : c \in \mathcal{A} \tag{2.10}$$

Yokobayashi and Wakahara [146] binarize character images using local adaptive thresholding in one of the Cyan/Magenta/Yellow (CMY) color planes, depending on the breadth of their histogram. Then they use a normalized cross-correlation as a matching measure between the test image and a list of templates, which were synthetic images of a single font. To our knowledge they were the first ones to use synthetic data as training samples for scene text, an idea which we also exploited in Chapter 4.

Li and Tan [68] calculate Cross Ratio spectrum over the boundary of a region. The Cross Ratio is a ratio between distances of four collinear points and it remains unchanged under any perspective transformation. Comparing two Cross Ratio Spectra employs Dynamic Time Warping algorithm and therefore has a quadratic complexity for a single comparison, which is the main drawback of the method - classifying a single character can take several minutes on a standard PC.

Figure 2.12 Cropped character recognition. Given an image of a character **I** (left), a single character is selected from the output alphabet (right).

De Campos et al. [26] evaluate six different local features - Shape Contexts [12], Geometric Blur [13], Scale Invariant Feature Transform [75], Spin Image [60], Maximum Response of Filters [127] and Patch Descriptor [128] - for individual character recognition. Together with the published Chars74k dataset (see Section 3.1), their work represents a useful baseline in the scene text recognition.

Newell and Griffin [96] extend the work of De Campos et al. [26] by exploiting Histogram of Oriented Gradients (HOG) [25] in a scale-space pyramid. Lee et al. [62] learn a set of discriminative features which exploit the most informative sub-regions of each character within a multi-class classification framework.

2.4 Cropped Word Recognition

In the *cropped word recognition*, a method outputs a sequence of characters based on an image **I** of a single word which was manually cropped out by a human annotator.

$$\mathbf{I} \rightarrow (c_1, c_2, \ldots, c_n) : c_i \in \mathcal{A} \tag{2.11}$$

where $\mathcal{A}$ is the output alphabet (typically "A" to "Z", sometimes lowercase "a" to "z" and "0" through "9" are added).

In the evaluation, a standard case-sensitive Levenshtein distance [66] between the method output and the ground truth is calculated for each word (image). The overall method accuracy is then given by the sum of the Levenshtein distance over the whole test set.

Figure 2.13 Cropped word recognition. Given an image of a single word **I** (left), a sequence of characters is produced (right).

The cropped word recognition tasks gives an upper-bound currently achievable in plain scene text recognition performance, but in fact it assumes that 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. Also, there are words which cannot be correctly recognized without further context, which is unavailable when the word has already been cropped out.

Weinman et al. [135] combine lexicon, similarity and appearance information into a joint model and use Sparse Belief Propagation [102] to infer the most probable string

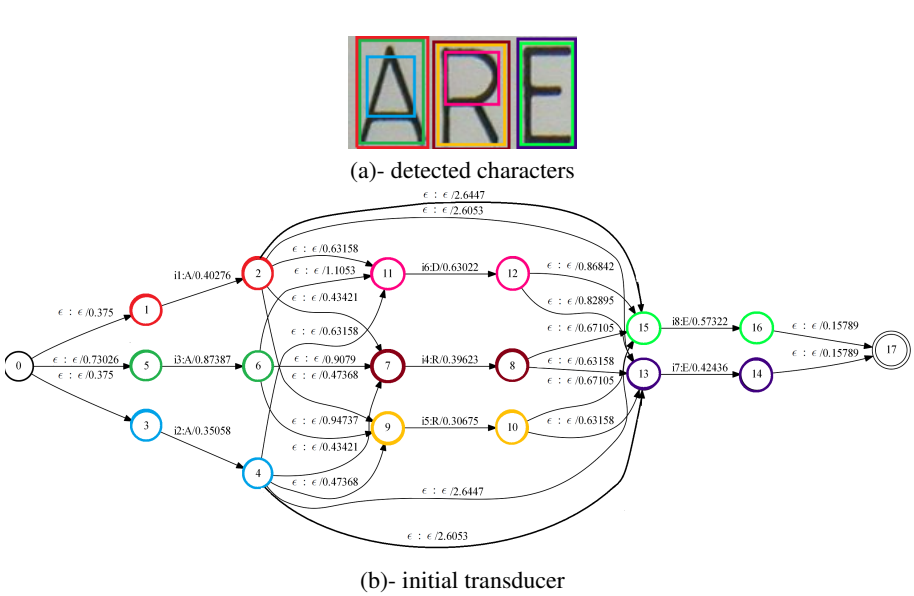


Figure 2.14 The cropped word recognition problem is formulated by Novikova et al. as a maximum a posteriori (MAP) inference in a weighted finite-state transducer [99]. Images taken from [99]

content. Similarly, Mishra et al. [84] use a joint CRF model [59] to combine individual character detection results with a language model (lexicon).

Novikova et al. [99] propose a probabilistic model which combines local likelihood and pairwise positional consistency priors with higher order priors that enforce consistency of characters and their attributes, such as font and colour. The word recognition is then formulated as a maximum a posteriori (MAP) inference and weighted finite-state transducers are exploited to find the optimal solution. A weighted finite-state transducer is a directed multigraph, where each vertex corresponds to a state, and each edge has a weight, an input character and an output character. To get the optimal solution, the shortest path in the transducer that accepts the optimal location sequence and produces the optimal word has to be found (see Figure 2.14).

Yao et al. [142] learn mid-level character representation (strokelets) in a partially supervised manner, adapting the generic patch discovery algorithm of Singh et al. [121]. In the recognition stage, a sliding window is shifted over the word image, activations of different types of strokelets are tracked at each position, and the strokelet responses are binned together to form a histogram feature vector, which is classified by a Random Forest [15].

The PhotoOCR method of Bissaco et al. [14], which won the 2013 ICDAR Robust Reading competition [53], exploits a deep neural network classifier trained on 2-level HOG features [25]. Each word image is first over-segmented into characters (or their fragment), each segment is classified by the NN classifier and finally Beam Search [114] with strong N-gram language mode is applied to find the optimal character sequence. The method also benefits from a large private training dataset with 10^7 training samples.

Almazán et al. [10] find a common embedding subspace for word images and their labels, so that they are close to each other in terms of Euclidian distance in the embedding space. Each label is first represented by a pyramidal histogram of characters representation, which embeds strings into a n-dimensional space. Then, an attribute model is trained to represent the word images and Canonical Correlation Regression is

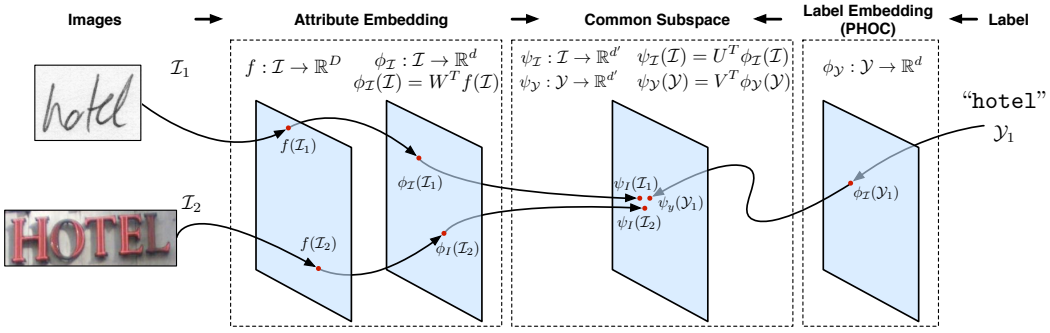


Figure 2.15 Almazán et al. find a common embedding subspace for word images and their labels, so that they are close to each other [10]. Image taken from [10]

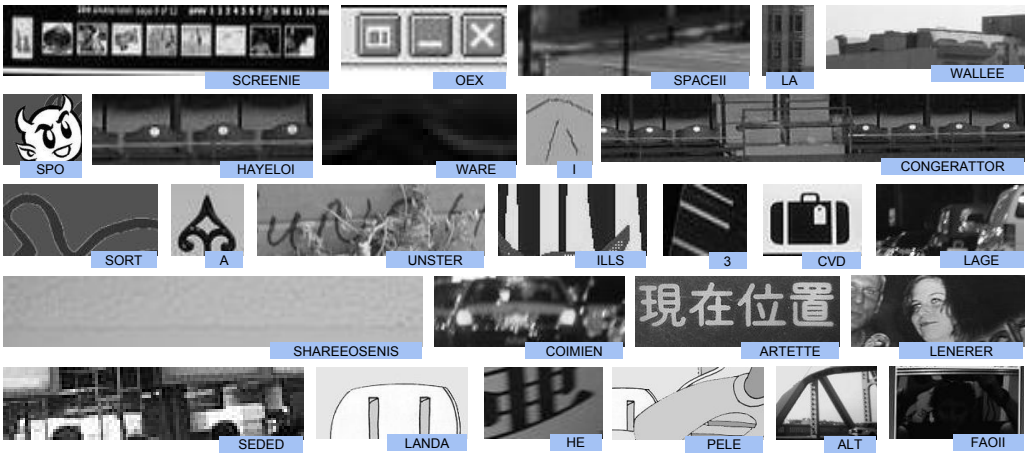


Figure 2.16 The model of Lee and Osindero [63] implicitly captures the language model of the training data, which is demonstrated by its output on images without any text. Images taken from [63]

applied to find the final common embedding subspace (see Figure 2.15). Gordo [39] further improves the embedding framework by exploiting mid-level features for character representations in the attribute model.

Lee and Osindero [63] explore several Recurrent Neural Network (RNN) [137] models with Long Short-Term Memory (LSTM) [43], built on top of a Recursive CNN [69] with "soft" attention modelling [140] to filter features which are fed into the RNN. The model can recognize word images without any supplied lexicon (unconstrained text recognition), however the language model is strongly incorporated in the RNN parameters, which is demonstrated by random predictions on images without any text (see Figure 2.16).

Shi *et al.* [119] train a fully-convolutional network with a bidirectional LSTM using the Connectionist Temporal Classification (CTC), which was first introduced by Graves *et al.* [40] for speech recognition to eliminate the need for pre-segmented data. Unlike the method presented in Section 6, Shi *et al.* [119] only recognize a single word per image (*i.e.* the output is always just one sequence of characters), they resize the source image to a fixed-sized matrix of 100×32 pixels regardless of how many characters it contains and the method is significantly slower because of the LSTM layer.

2.5 Word Spotting

Due to the complexity of the general scene text recognition problem, some methods focus on a more constrained scenario when a relatively small lexicon of words is given with each image and the aim is to localize only the words present in the lexicon. This constrained scenario still has many interesting applications, such as local navigation system for the blind (where possible names of local businesses in the area are known based on current GPS position, but their exact location needs to be determined through a vision system). This task formulation, which we refer to as *word spotting*, was first introduced by Wang and Belongie [133]; in the most recent ICDAR 2015 Robust reading competition [52], the task is referred to as "End to End", however in this thesis we will use the term "end to end" only in the context of the general scene text recognition problem.

In the *word spotting* formulation, each image $\mathbf{I}$ is accompanied with a set of words $\mathcal{L}$ called the *lexicon* and the goal is to find all rectangular areas (x, y, w, h) whose content in the image $\mathbf{I}$ corresponds to a word of the *lexicon*.

$$\mathcal{L} \xrightarrow{\mathbf{I}} D = \left\{(x_i, y_i, w_i, h_i)\right\}^m \quad x_i, y_i, w_i, h_i \in \mathbb{N} \cup \emptyset \tag{2.12}$$

$$\mathcal{L} \in \big\{(c_{1,1}, c_{1,2}, \ldots, c_{1,n_1}), (c_{2,1}, c_{2,2}, \ldots, c_{2,n_2}), \ldots (c_{m,1}, c_{m,2}, \ldots, c_{m,n_m})\big\} : c_{i,j} \in \mathcal{A} \tag{2.13}$$

Note that not all words in the lexicon are present in the image (so-called "distractor" words, denoted as $(\emptyset, \emptyset, \emptyset, \emptyset)$), and that not all words in the image have a corresponding content in the lexicon.

In the evaluation, the image recall r and precision p are calculated as

$$r = \frac{|M|}{|G|} \tag{2.14}$$

$$p = \frac{|M|}{|D|} \tag{2.15}$$

$$M = \left\{d_i \in D \mid m(d_i, G) = 1\right\} \tag{2.16}$$

where $m(d_i, G)$ is a match predicate which declares whether the detection d_i is matched to the image ground truth G.

In the most recent ICDAR competition [52], a detection d_i is matched to the ground truth ($m(d_i, G) = 1$) if

- the character sequence is identical (using case-insensitive comparison), and
- the Intersection-over-Union [30] of the detected and the ground truth bounding box is above 50%.

Wang and Belongie [133] detect and recognize individual characters using a multi-scale sliding window approach. A Histograms of Oriented Gradient (HOG) features are calculated for each window position and a nearest neighbor classifier is used to measure distance between the patch and all character templates in all classes. Then, each word in the lexicon is considered and the cost of its character configuration is estimated using pictorial structures [33] that penalize disagreement with recognized labels and layout deformation. The method was further improved in [132], where the nearest neighbor classifier was replaced by Random Ferns [15] and an SVM classifier [24] is applied for word re-scoring in order to incorporate higher-order features of word configuration (e.g.

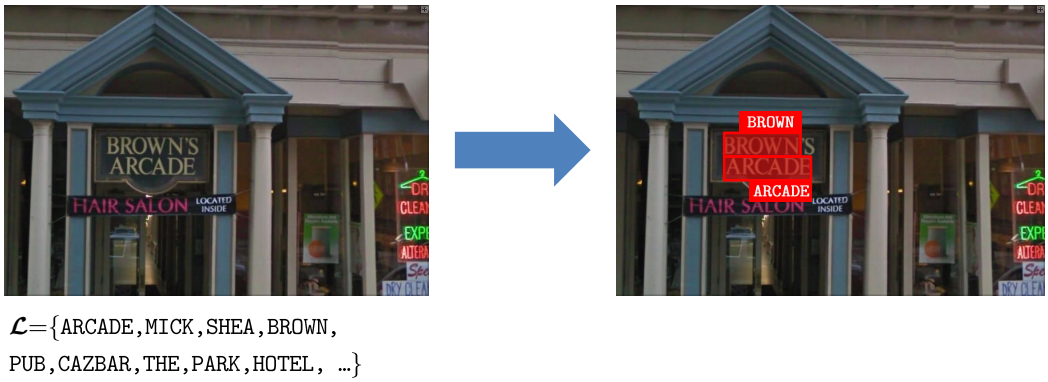


Figure 2.17 Word spotting output example. Given given an image $\mathbf{I}$ with lexicon $\mathcal{L}$ (left), two words of the lexicon are localized in the image (right).

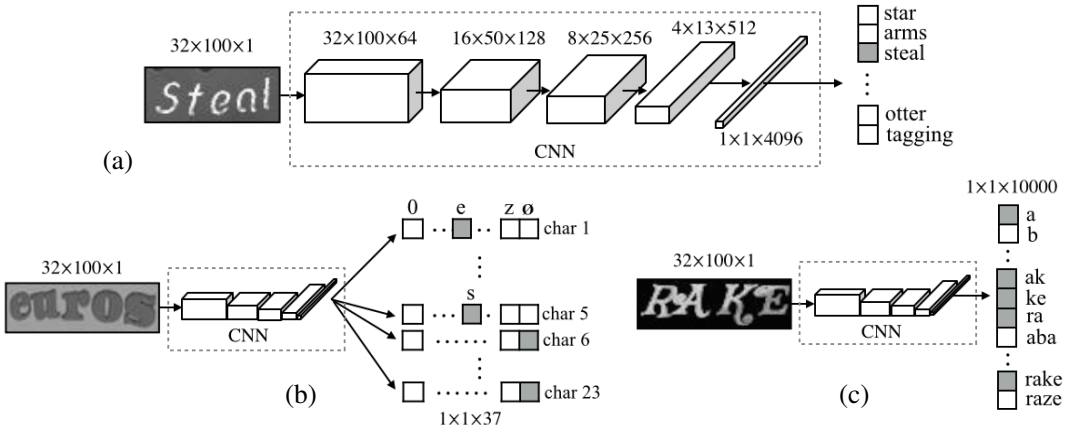


Figure 2.18 Three different word output encodings proposed by Jaderberg et al. [45] - dictionary encoding (a), character-at-position encoding (b) and character N-gram encoding (c). Images taken from [45]

standard deviation of character spacing). Similarly, the method of Wang et al. [134] uses a sliding-window approach combined with convolutional neural network classifiers. It is demonstrated that the method performs well on noisy or otherwise distorted images, however the accuracy significantly decreases with increasing size of the lexicon.

Jaderberg et al. [49] train a character-centric Convolutional Neural Network (CNN) [61], which takes a 24×24 image patch and predicts a text/no-text score, a character and a bigram class. The input image is scanned by the trained network in 16 scales and a text saliency map is obtained by taking the text/no-text output of the network. Given the saliency maps, word bounding boxes are then obtained by the run length smoothing algorithm. Finally, each word is recognized independently by taking the cropped word image and finding the optimal label sequence through the character and bigram classifier output by dynamic programming.

The method is further improved in [47], where a word-centric approach is introduced. First, horizontal bounding-box proposals are detected by aggregating the output of the standard Edge Boxes [150] and Aggregate Channel Feature [27] detectors. Each proposal is then classified by a Random Forest [15] classifier to reduce the number of false positives and its position and size is further refined by a CNN regressor, to obtain

a more suitable cropping of the detected word image. Each cropped word image is then resized to a fixed size of 32×100 pixels and classified as one of the words in the dictionary (see Figure 2.18a), where in their setup the dictionary contains $\sim 90\,000$ English words (plus words of the testing set, see Section 3.7). The classifier is trained on a dataset of 9 million synthetic word images uniformly sampled from the dictionary. As the last step, duplicate and overlapping detections are eliminated by Non-Maxima Suppression and the bounding boxes are further refined by the CNN regressor in an iterative manner to obtain the final word positioning.

The requirement of a dictionary is relaxed in [46], where an unconstrained text recognition architecture is presented. The model either outputs a character class (or an empty label) for every of 23 possible character positions in the word (see Figure 2.18b), or it outputs a set of character N-grams, where each N-gram is a substring of the word up to 4 characters in length (see Figure 2.18c).

3 Datasets

3.1 Chars74K Dataset

The *Chars74K dataset* [1] was collected by de Campos et al. [26]. It contains 7705 English character images (A-Z, a-z and 0-9, in total 64 classes) and 3345 Kannada character images (647 classes), which were manually segmented from 1922 scene text images (see Figure 3.1). Additionally, the dataset contains 1416 scene images with each word annotated by a polygon and its text transcription, however not every word in the dataset is annotated.

(a) (b) (c)

Figure 3.1 Samples from the Chars74K dataset. Individual English characters (a), Kannada characters (b) and full images (c)

3.2 ICDAR 2003 Dataset

The *ICDAR 2003 dataset* [77] was created by Simon Lucas and his colleagues for the ICDAR 2003 Robust Reading competition [77]. The dataset was used in an unchanged

Figure 3.2 A sample of images used in the ICDAR 2005 and ICDAR 2011 datasets. Text in the images is mostly horizontal, it occupies a large portion of an image and it typically is present in the middle of an image.

form in the ICDAR 2005 Robust Reading competition [76], which is why in the literature sometimes the dataset is also referred to as the *ICDAR 2005 dataset*.

The dataset contains 258 training and 251 testing images with words and characters annotated by bounding boxes and their text content. 1157 word and 6185 character images (1111 word and 5430 character images) were subsequently cropped from the training (respectively testing) image set, to be used in Cropped Word Recognition (Section 2.4) and Cropped Character Recognition (Section 2.3) evaluation.

The dataset was captured by people who were specifically tasked to capture text in an outdoor environment, so as a result text in the dataset is mostly horizontal, it occupies a large portion of an image and it typically is present in the middle of an image [129], since the authors of the pictures tried to capture "nice" pictures of text (see Figure 3.2).

3.3 ICDAR 2011 Dataset

The *ICDAR 2011 dataset* [2] was created by taking all images from the ICDAR 2003 dataset (see Section 3.2), removing images with no text, adding several new images and splitting them again into a training and a testing subset. The dataset was first used in the ICDAR 2011 Robust Reading Competition [118] and then subsequently in the 2013 competition [53], which is why it is sometimes referred to as the *ICDAR 2013 dataset*. In the ICDAR 2015 Robust Reading competition [52], the dataset was used again in the *Focused Scene Text* challenge (Challenge 2).

The dataset contains 229 training and 255 testing images, with corresponding 849 training and 716 testing cropped word images. As a result of the creation process, the testing subset of the ICDAR 2011 dataset contains the same images as the training

ICDAR 2003 **training set** ICDAR 2011 **testing set**

Figure 3.3 The same images are present in the ICDAR 2003 **training set** (left) and the ICDAR 2011 **testing set** (right), which is why training on both ICDAR 2003 and 2011 training sets and then evaluating on the ICDAR 2011 testing set is not possible - a common mistake in the literature

subset of the ICDAR 2003 dataset. This unfortunately often leads to evaluation problems in the literature, where some methods are trained on both ICDAR 2003 and 2011 training sets, falsely assuming they are different datasets, and evaluated on the 2011 testing set - but the testing set contains many images from the joint training set, and therefore the accuracy evaluation is heavily skewed (see Figure 3.3).

3.4 ICDAR 2015 Dataset

The *ICDAR 2015 dataset* [2] was introduced in the ICDAR 2015 Robust Reading Competition [52] to address the problems of the ICDAR 2003/2011 datasets (see Sections 3.2 and 3.3). The dataset is used in the *Incidental Scene Text* challenge (Challenge 4).

Figure 3.4 Sample images from the ICDAR 2015 dataset (also known as *Incidental Scene Text* challenge). Many realistic effects such as occlusion, perspective distortion, blur or noise are present.

The images were collected by people wearing Google Glass devices [138] and walking in Singapore, and then subsequently by selecting and annotating only images with text.

The images in the dataset were taken "not having text in mind", and therefore contain a high variability of text fonts and sizes and they include many realistic effects - e.g. occlusion, perspective distortion, blur or noise (see Figure 3.4).

The dataset contains 1670 images with 17548 annotated words - 1500 images are publicly available, split into training and testing set, and the remaining 170 images represent a sequestered set for a future use. Each word is annotated by a quadrilateral (3 points) and its Unicode transcription, thus supporting rotated and slanted text.

3.5 Street View Text Dataset

Figure 3.5 Sample images from the Street View Text dataset.

The *Street View Text (SVT) dataset* [3] was published by Wang and Belongie [133], where the data was collected by asking annotators to find images with local businesses in the Google Street View application. The annotators were instructed to find a representative text associated with the business in the image, then to move the view point in the application to minimize the skew of the text and finally to save the screen shot. The dataset therefore contains mostly business names and business signs (see Figure 3.5), and the business names can be typically obtained in publicly available dictionaries by looking up businesses close to the GPS position of the image.

The words in the image picked by the annotators in this process are tagged by a horizontal bounding-box and a case-insensitive transcription. Note that obviously as a result only a small fraction of text in the images is labelled.

Each image is also associated with a lexicon of 50 unique words, which contains the words tagged by the annotator, as well as words from business names present near the location the image was taken. In total, the data set contains 350 images (100 training and 250 testing images) of 20 different cities and 725 labeled words.

The word annotations were also exploited to create the dataset of cropped words *SVT-50*, which contains 647 word images, each with a lexicon of 50 words. There is also a lexicon of all test words (4282 words), which is referred to as *SVT-FULL* [134].

Figure 3.6 Samples from the COCO-Text dataset and their legibility classification and text category. Images taken from [129]

3.6 COCO-Text Dataset

The *COCO-Text dataset* [4, 129] with its 63 686 images and 173 589 annotated words is the largest scene text dataset to date. The COCO-Text dataset was as the name suggests based on the MSCOCO dataset [71] and each word is associated with a horizontal bounding box, a legibility classification (legible/illegible), a text category (machine-printed, hand-written, etc.), a script (Latin/other) and a Unicode transcription in case of a Latin script. The original object categories from MSCOCO are available as well.

Since the images in the dataset were not collected with text in mind, the variety of text and its hidden parameters (font, style, script, positioning, etc.) is higher than in any of the existing datasets (see Figure 3.6).

3.7 MJSynth Dataset

In order to address the problem of small volume of training data to apply deep-learning methods, Jaderberg et al. [45] created a synthetic dataset of 9 million word images called *MJSynth* [5] (also referred to as the *Synth90k dataset*).

The dataset is based on a set 50 000 English words from the Hunspell dictionary [6], augmented with all their suffixes and prefixes, and all the words from the **testing** subsets of the ICDAR 2011 (Section 3.3), SVT (Section 3.5) and IIIT5k (Section 3.9) datasets. The need to include words from the testing set is driven by the word-centric approach to classify a word image into 1 of 90 000 classes (only words from the training dictionary can be outputted [47]), however as a result there is not a clear separation between training and testing data, and in the testing phase the system trained on the synthetic data has already "seen" all the words in the testing set, although with a different appearance.

Each word from the aforementioned dictionary is then rendered 100 times into a synthetic word image, each time randomly applying different font, border rendering,

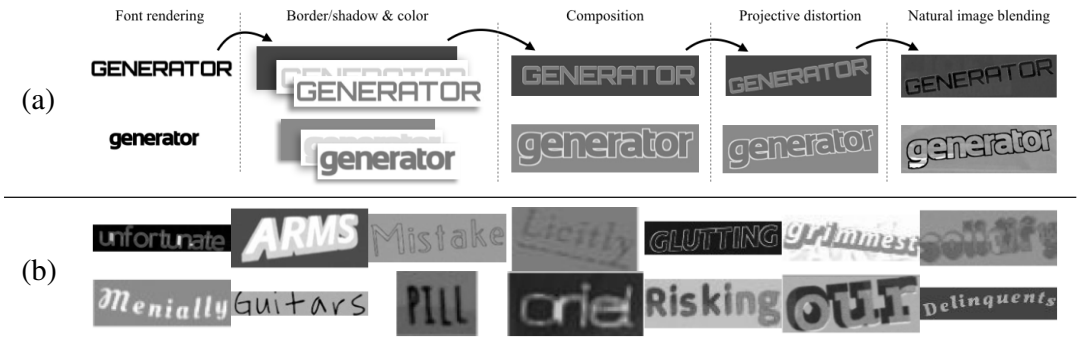


Figure 3.7 The rendering process of the MJSynth dataset [45] (a). Samples from the MJSynth dataset (b). Images taken from [45]

base colouring, perspective distortion, natural data blending (using a random patch from the SVT and ICDAR 2003 datasets) and noise (see Figure 3.7a), thus generating 9 million word images. The redendered data were then randomly divided into 7.2M training, 900k validation and 900k testing images.

3.8 SynthText Dataset

The *SynthText dataset [7]* created by Gupta et al. [41] contains 800 000 synthetically created images with text. The background image is randomly taken from a set of 8 000 background images downloaded from Google Image Search, and text (word or up to 3 text lines) extracted from the Newsgroup20 dataset is rendered into the image, respecting local texture and geometry cues (see Figure 3.8). The dataset therefore contain highly realistic scene text images with full annotations, although the placement of the text does not respect priors of the real word - for example, text does not typically appear on horse hair (Figure 3.8, bottom right).

Figure 3.8 Sample images from the SynthText dataset, the ground truth depicted by red rectangles. Images taken from [41]

3.9 IIIT Datasets

The *IIIT 5k Word (IIIT5k) dataset* [8, 82] contains 5 000 (2 000 training and 3 000 testing) word images cropped from images with text found on the Internet. Each word image has a case-insensitive transcription and a lexicon of 50 (*IIIT5k-50*) or 1 000 (*IIIT5k-1k*) words.

The *IIIT Scene Text Retrieval dataset* [83] consists of 10 000 images downloaded from Flickr. There are 50 text query words, and each word is associated with a list of 10 to 50 images, which contain the word. There are also many distractor images with no text at all. Note that this dataset does not contain any text localization information, so it can only be applied for text retrieval tasks.

3.10 MSRA-TD500 Dataset

The *MSRA-TD500 dataset* [141] consists of 500 scene text images, split into 300 training and 200 testing samples. The dataset contains English as well as Chinese text in different orientations and text is annotated on a text-line level, where each text line is associated with a rotated rectangular bounding box.

3.11 KAIST Dataset

The *KAIST dataset* [65] contains 3 000 images of indoor and outdoor scenes with text. Words and characters are annotated by a bounding box, and character segmentations are provided as well. The dataset contains text in English and Korean, however not all text instances are annotated.

3.12 NEOCR Dataset

The *Natural Environment OCR dataset (NEOCR)* was introduced by Nagy et al. [87] and it contains 659 real world images with 5238 text line annotations. Each text line is annotated by a quadrilateral, its Unicode content and several additional attributes such as language, type face, occlusion level or noise level.

4 Text Recognition by Extremal Regions

4.1 Character Detection

4.1.1 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 our method we consider 4-connected pixels, 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 $\mathcal{D}$

$$\forall p_i, p_j \in \mathcal{R}\ \exists p_i, q_1, q_2, \ldots, q_n, p_j : p_i A q_1, q_1 A q_2, \ldots, q_n A p_j \tag{4.1}$$

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

$$\partial\mathcal{R} = \{p \in \mathcal{D} \setminus \mathcal{R} : \exists q \in \mathcal{R} : pAq\} \tag{4.2}$$

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

$$\forall p \in \mathcal{R}, q \in \partial\mathcal{R} : \mathbf{C}(q) > \theta \geq \mathbf{C}(p) \tag{4.3}$$

where θ denotes threshold of the Extremal Region (see Figure 4.2).

We consider RGB and HSI color spaces [21] 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 4.1):

$$\mathbf{C}_\nabla(p) = \max_{q \in \mathcal{D} : pAq} \{|\mathbf{C}_\mathrm{I}(p) - \mathbf{C}_\mathrm{I}(q)|\} \tag{4.4}$$

In real-world images there are certain instances where characters are formed of smaller elements (see Figure 4.3 left) or a single element consists of multiple joint characters (see Figure 4.3 right). By pre-processing the image in a Gaussian pyramid, in each

(a) (b) (c)

Figure 4.1 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)

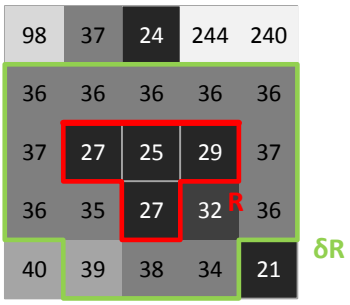


Figure 4.2 *Extremal Region* $\mathcal{R}$ is a region whose outer boundary pixels $\partial\mathcal{R}$ have strictly higher values than pixels of the region itself. θ denotes threshold of the Extremal Region

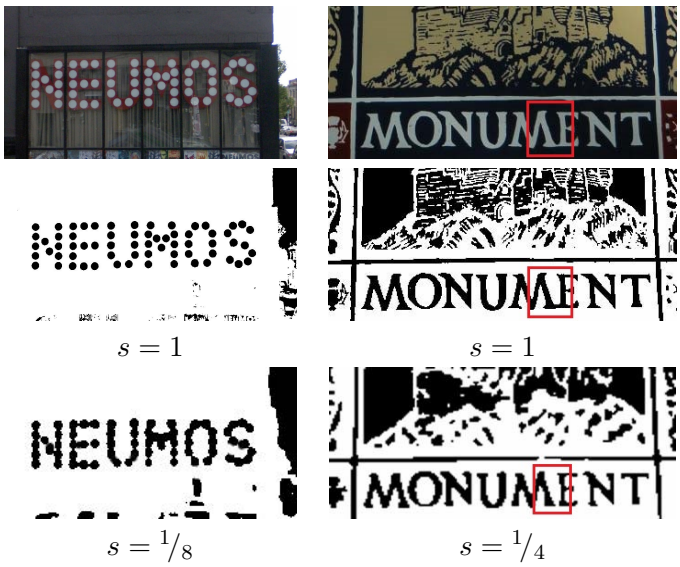


Figure 4.3 Processing with a Gaussian pyramid (the pyramid scale denoted by s). Characters formed of multiple small regions merge together into a single region (left column). A single region which corresponds to characters "ME" is broken into two regions and serifs are eliminated (right column)

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.

4.1.2 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* (see Figure 4.4) 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 all pixels in the image). As proposed by

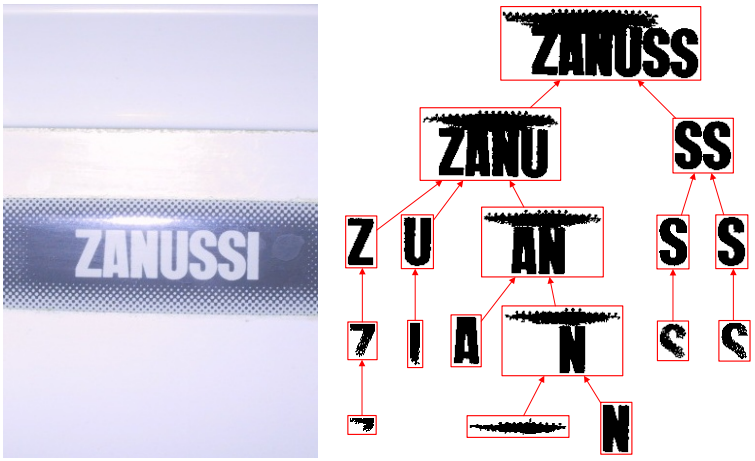


Figure 4.4 Extremal Region (ER) lattice induced by the inclusion relation.

Zimmerman and Matas [81], 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,

$$r = \left(\bigcup u \in R_{\theta-1}\right) \cup \left(\bigcup p \in \mathcal{D} : \mathbf{C}(p) = \theta\right) \tag{4.5}$$

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,

$$\phi(r) = \left(\oplus\, \phi(u)\right) \oplus \left(\oplus\, \psi(p)\right) \tag{4.6}$$

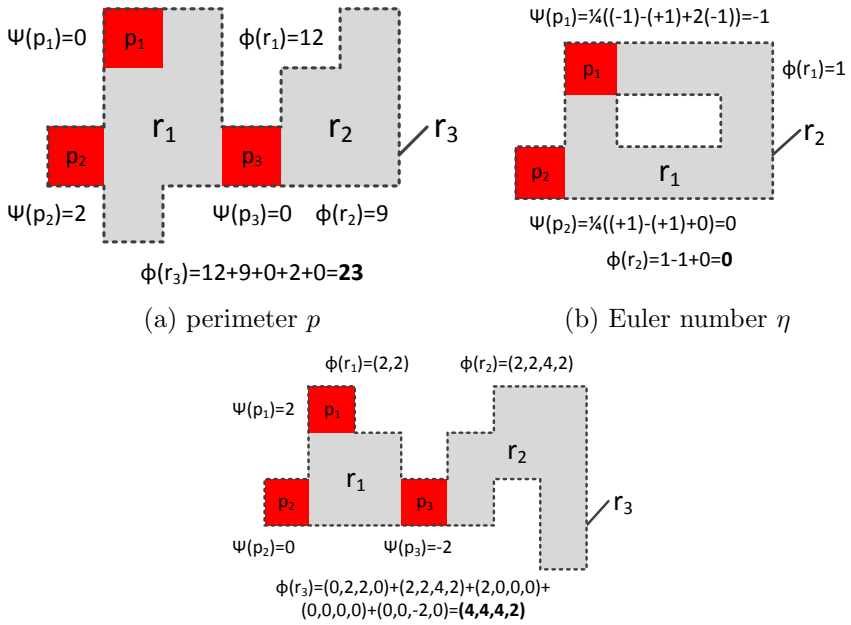
where $\oplus$ denotes an operation that combines descriptors of the regions (pixels) and $\psi(p)$ denotes an initialization function that computes the descriptor for given pixel p. We refer to such descriptors where $\psi(p)$ and $\oplus$ exist as *incrementally computable* (see Figure 4.5).

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 method 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



(a) perimeter p (b) Euler number η

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

Figure 4.5 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

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 4.5a). The initialization function $\psi(p)$ determines a change of the perimeter length by the pixel p at the threshold where it is added

$$\psi(p) = 4 - 2|\{q : qAp \wedge \mathbf{C}(q) \leq \mathbf{C}(p)\}| \tag{4.7}$$

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 [106] calculates the Euler number by counting 2×2 pixel patterns called quads. Consider the following patterns of a binary image:

$$Q_1 = \left\{ \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}, \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}, \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix}, \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}, \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix}, \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix}, \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}, \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix} \right\} \tag{4.8}$$

$$Q_2 = \left\{ \begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}, \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix}, \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}, \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}, \begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix} \right\} \tag{4.9}$$

$$Q_3 = \left\{ \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}, \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \right\} \tag{4.10}$$

Euler number is then calculated as

$$\eta = \frac{1}{4}(C_1 - C_2 + 2C_3)) \tag{4.11}$$

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 4.5b),

$$\psi(p) = \frac{1}{4}\left(\Delta C_1 - \Delta C_2 + 2\Delta C_3\right)) \tag{4.12}$$

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 4.5c and 4.8). 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.

4.1.3 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 4.6).

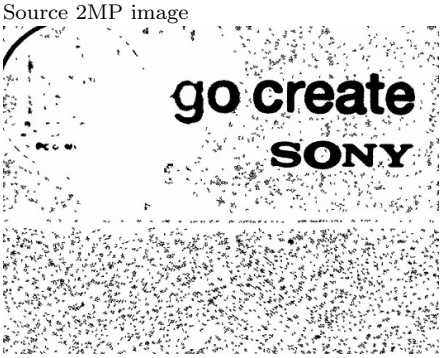
In the first stage, a threshold is increased step by step from 0 to 255, incrementally computable descriptors (see Section 4.1.2) are computed in $O(1)$ for each ER r and the descriptors are used as features for a classifier which estimates the class-conditional probability $p(\text{character}|r)$. The value of $p(\text{character}|r)$ is tracked using the inclusion relation of ER across all thresholds (see Figure 4.7) 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 $\Delta_{\min}$).

A Real AdaBoost [116] classifier with decision trees was used with the following features (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 $(\hat{c} = \text{median}\,\{\mathbf{c}_{\frac{1}{6}w}, \mathbf{c}_{\frac{3}{6}w}, \mathbf{c}_{\frac{5}{6}w}\})$ which estimates number of character strokes in horizontal projection - see Figure 4.8. 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 [98]. 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 4.9).

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 our method, an SVM [24] classifier with the RBF kernel [86] was used, the parameter values σ and C were found by cross-validation on the training set. The

Source 2MP image

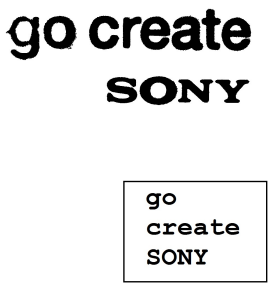
Intensity channel extracted



ERs selected in $O(N)$ by the first stage of the sequential classifier

ERs selected by the second stage of the classifier

Text lines formed



Only ERs in text lines selected and labelled by a character classifier

	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

Figure 4.6 Typical number of regions and timings in each stage (character detection in a single channel only) on a standard 2GHz PC

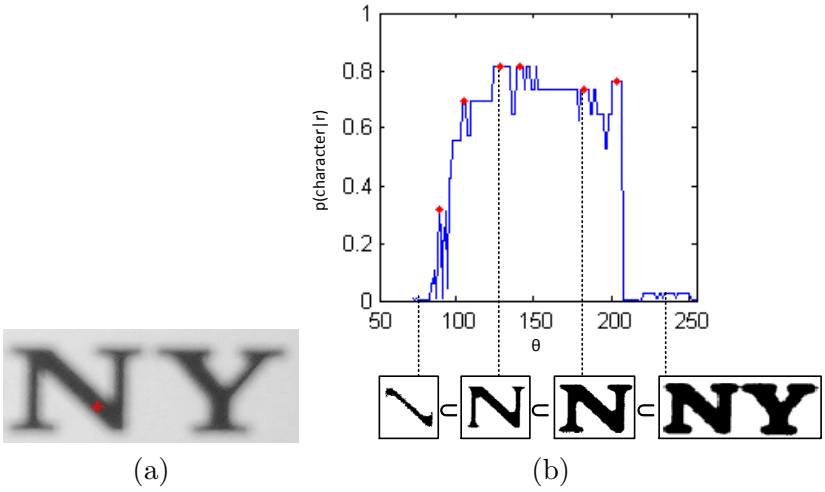


Figure 4.7 In the first stage of the sequential classification the probability $p(\text{character}|r)$ of each ER is estimated using incrementally computable descriptors that exploit the inclusion relation of ERs. (a) A source image cut-out and the initial seed of the ER inclusion sequence (marked with a red cross). (b) The value of $p(\text{character}|r)$ in the inclusion sequence, ERs passed to the second stage marked red

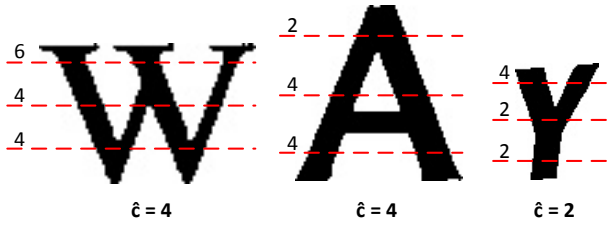


Figure 4.8 The horizontal crossings feature used in the 1st stage of ER classification

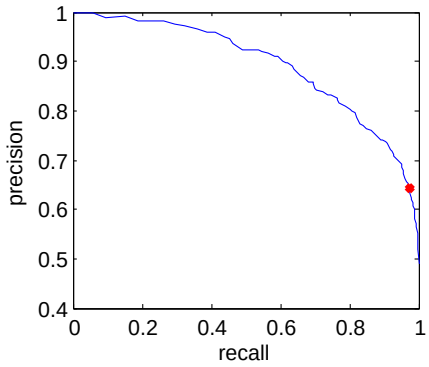


Figure 4.9 The precision-recall curve of the first stage of the sequential classifier obtained by cross-validation. The configuration used in the experiments marked red (recall 95.6%, precision 67.3%)

(a) (b) (c)

Figure 4.10 The features used by the sequential character classifier allow detection of characters in various scripts - Armenian (a), Russian (b) and Kannada (c). Note that the training set contains only Latin characters.

$\kappa = 0$ $\kappa = 5$ $\kappa = 6$ $\kappa = 14$ $\kappa = 15$ $\kappa = 93$

(a) (b)

Figure 4.11 The number of boundary inflexion points κ. (a) Characters. (b) Non-textual content

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 4.11). 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. They also enable detecting characters of different scripts, even though the training set only contains Latin alphabet (see Figure 4.10). The features are also 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 4.3.2).

4.2 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 R
Result: a set of triplets T
T ⟵ ∅;
for r¹ ∈ R do
    for r² ∈ N(r¹) do
        if v({r¹, r²}) = 0 then
            continue;
        end
        for r³ ∈ N(r²) do
            if v({r², r³}) = 0 then
                continue;
            end
            t ⟵ {r¹, r², r³};
            if v′(t) = 1 then
                T ⟵ T ∪ 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 [116]; 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.

In the second step (see Algorithm 2), each triplet is first turned into a text line (of a

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`
  $d \longleftarrow d_{\max}$;
  for $l \in \mathbf{L}$ **do**
    for $l' \in \mathbf{L} \setminus l$ **do**
      if $dist(l, l') < d$ **then**
        $d \longleftarrow \mathrm{dist}(l, l')$;
        $m \longleftarrow l, m' \longleftarrow l'$;
      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

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 $\mathrm{dist}(l, l')$ 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 $\mathrm{dist}(l, l')$ 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

$$\begin{aligned} \mathrm{dist}(l, l') &= \frac{\max(|b_l(\chi) - b_{l'}(\chi)|, |b_l(\chi') - b_{l'}(\chi')|)}{\min(\overline{h}_l, \overline{h}_{l'})} \\ \chi &= \min(\underline{x}_l, \underline{x}_{l'}) \\ \chi' &= \min(\overline{x}_l, \overline{x}_{l'}) \end{aligned} \tag{4.13}$$

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

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 we assume that a character can be present in one word only (see Section 4.2.2). 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.

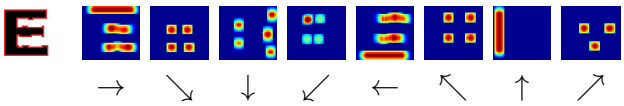


Figure 4.12 Character recognition features: Input character (left). Features of the chain-code bitmap for each direction (right).

Figure 4.13 Random samples from the training set. The set contains 5580 characters from 90 fonts with no distortions, blurring or rotations

4.2.1 Character Recognition

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

$$\overline{\mathbf{R}} = \bigcup_{l \in \mathbf{L}} \{r^1, \ldots, r^n \in l\} \tag{4.14}$$

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

$$\hat{L}(r) = \left\{ l(t) : t \in \mathcal{N}_K(f(r)) \mid \|f(t) - f(r)\| \leq \bar{d}(l(t)) \right\} \tag{4.15}$$

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 4.2.2).

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 [73] 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 4.12). After Gaussian blurring ($\sigma = 1.1$) each bitmap is sub-sampled 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 (see Figure 4.14), 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 [85] 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

Figure 4.14 The method can be also easily extended to incorporate additional characters or even whole scripts

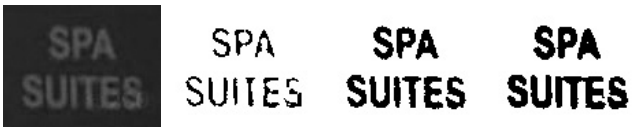


Figure 4.15 Character boundaries are often fuzzy and it is not possible to locally determine the threshold value unambiguously. Note the binarization of letters "ITE" in the word "SUITES" - as the threshold Θ is increased their appearance goes from "IIE" through "ITE" to "m"

used the value $\beta = 2.5$, which yields the best performance on the training subset of the ICDAR dataset (see Section 4.3).

4.2.2 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 4.15), 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 $\left(r_1 \mathcal{P} r_2\right)$ 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

$$G = (V, E) \tag{4.16}$$

$$V = \{r_l \in \mathcal{R} \times \mathcal{A} : l \in \hat{L}(r)\} \tag{4.17}$$

$$E = \{(r^1_{l_1}, r^2_{l_2}) : r^1 \mathcal{P} r^2\} \tag{4.18}$$

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

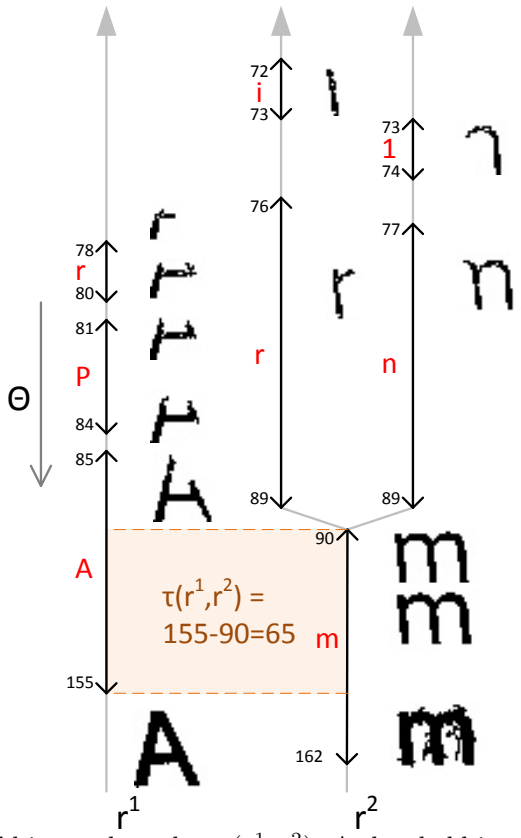


Figure 4.16 Threshold interval overlap $\tau(r^1, r^2)$. A threshold interval is an interval of thresholds during which the region has not changed its OCR label (red). Note that as the threshold is increased the region grows or merges with other regions and the label changes

predecessor of r^2 if the centroid of r^1 comes before the centroid of r^2 in the text line direction, the 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

$$s(r_l) = \alpha_1 \psi(r) + \alpha_2 \omega(r_l) \tag{4.19}$$

$$s(r^1_{l_1}, r^2_{l_2}) = \alpha_3 \tau(r^1, r^2) + \alpha_4 \lambda(l_1, l_2) \tag{4.20}$$

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 4.2.1). 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 $d_{\min}$

$$d_{\min}(r) = \min_{t' \in \mathcal{N}_K(f(r))} d(t', r) \tag{4.21}$$

$$\omega(r_l) \approx \frac{1}{K} \sum_{t \in \mathcal{N}_K(f(r)) : l(t) = l} \frac{d_{\min}(r)}{d(t, r)} \tag{4.22}$$

$$d(x, y) = \|f(x) - f(y)\| \tag{4.23}$$

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 4.16).

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 (see Figure 4.17), 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.

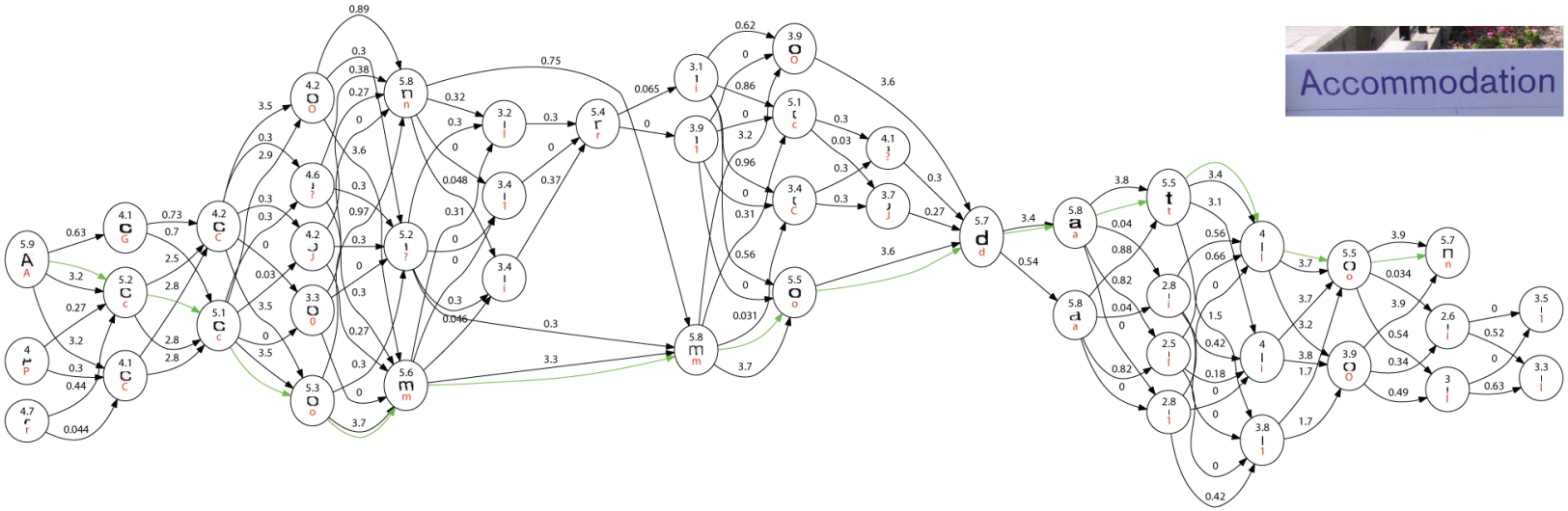


Figure 4.17 The final segmentation sequence is found as an optimal path in a directed graph where nodes correspond to labeled regions and edges represent a "is-a-predecessor" relation. Each node r_c represents a region (segmentation) r with a label c (red). Node (edge) score $s(r_c)$ resp. $s(r^1_{c_1}, r^2_{c_2})$ denoted above the node resp. edge. Optimal path denoted in green

4.3 Experiments

4.3.1 Character Detector

An experimental validation shows that 85.6% characters in the ICDAR 2013 dataset [53] 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 4.1). 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.

4.3.2 ICDAR 2013 Dataset

The proposed method was evaluated on the ICDAR 2013 Robust Reading competition dataset [53] (which is the same dataset as in the 2011 competition), 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 [139], the method reaches the recall of 71.3%, precision of 82.1% and the f-measure of 76.3% in text localization (see Figure 4.19 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. [144] (68%). The overall f-measure (76%) outperforms all published methods (see Table 4.2), 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 4.20). 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 4.2), where the word consists of connected letters (even if the line is formed, such region consisting of

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

Table 4.1 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

method	recall	precision	f-measure
proposed method	**72.4**	81.8	**77.1**
Yin et al. [144]	68.3	**86.3**	76.2
TexStar (ICDAR'13 winner) [53]	66.4	88.5	75.9
Kim (ICDAR'11 winner) [118]	62.5	83.0	71.3

Table 4.2 Text localization results on the ICDAR 2013 dataset

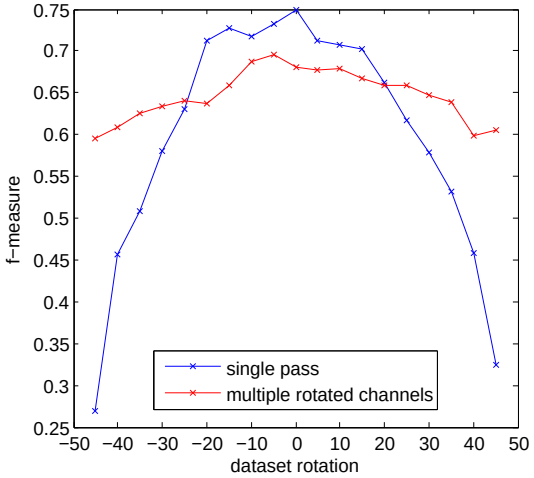


Figure 4.18 Text localization performance on the rotated ICDAR 2013 dataset

method	recall	precision	f-measure
proposed method	**45.4**	**44.8**	**45.2**
Weinman et al. [136]	41.1	36.5	33.7

Table 4.3 End-to-end text recognition results on the ICDAR 2013 dataset (case-sensitive).

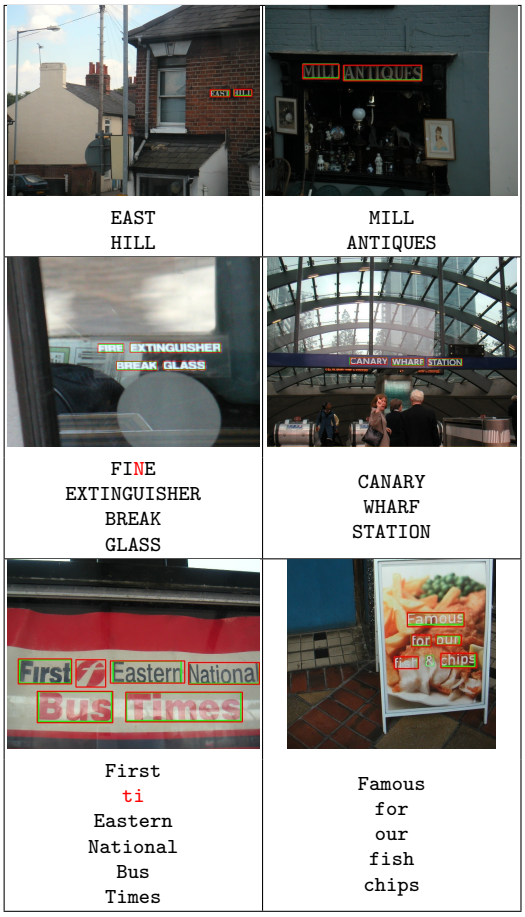


Figure 4.19 Text localization and recognition results on the ICDAR 2013 dataset. Ground truth marked green, method output marked red

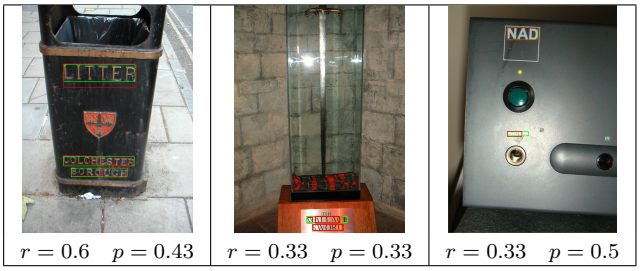


Figure 4.20 The main cause for the lower precision in text localization in some images are missed characters, which then result in a word being only partially detected or detected as multiple words. Such partial/multiple detections are heavily penalized by the evaluation protocol (overall image recall r and precision p denoted below each image)

Figure 4.21 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).

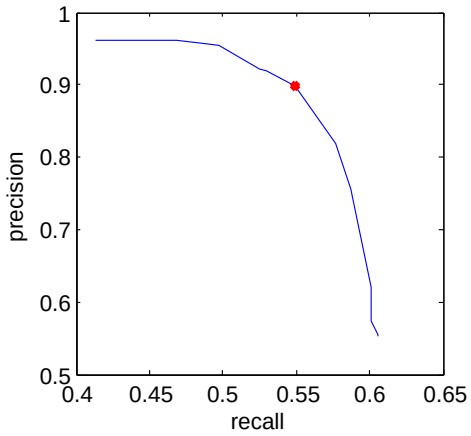


Figure 4.22 Precision and recall of end-to-end word detection and recognition on the Street View Text dataset. The configuration with the highest f-measure (68.1%) marked red

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 4.21.

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 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. [136] (see Table 4.3), mostly benefiting from a superior text localization phase. The dataset was also exploited in the ICDAR 2015 Robust Reading competition (see Section 4.3.4, Table 4.7).

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 4.18). 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 4.18 - red).

method	f-measure
proposed method	**68.1**
proposed method (general language model)	64.7
T. Wang et al. [134]	46.0
K. Wang et al. [132]	38.0

Table 4.4 End-to-end word detection and recognition results on the Street View dataset

Figure 4.23 Text localization and recognition results on the SVT dataset. Ground truth marked green, method output marked red

4.3.3 Street View Text Dataset

The Street View Text (SVT) dataset [132] 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 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 [134, 132] 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 4.2.2) 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

(a) (b) (c)

Figure 4.24 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)

Figure 4.25 "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)

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 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 4.22).

Using the same evaluation protocol as [134], 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 4.4). The method is able to cope with low-contrast and noisy text and high variety of different fonts (see Figure 4.23 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 4.25 - left column) or by incorrect ground truth where confuser words are actually present in the image (see Figure 4.25 - right column). The method fails to detect text where letters 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 4.24).

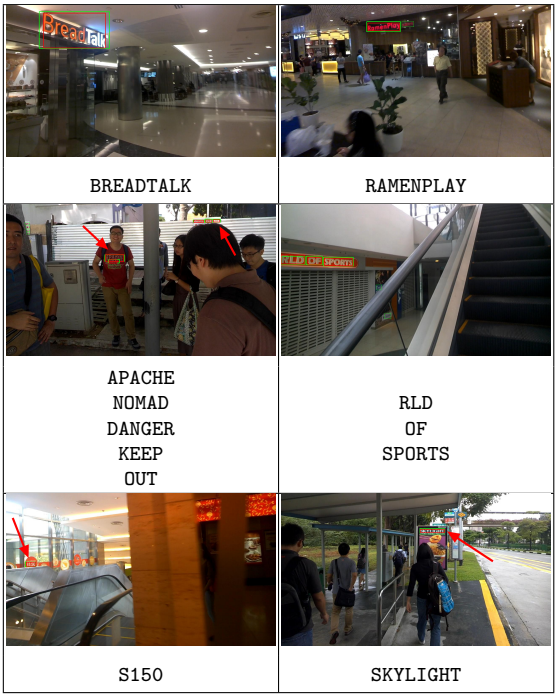


Figure 4.26 Text localization and recognition results on the ICDAR 2015 Incidental scene text dataset. Best viewed zoomed in color.

If general character transition probabilities $\lambda(c_1, c_2)$ for English (i.e. the same ones as in the Section 4.3.2) 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.

4.3.4 ICDAR 2015 Competition

The proposed method was used as the baseline method[1] in the ICDAR 2015 Robust Reading Competition [52] (see Figure 4.26). 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 4.20 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 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 4.3.2), 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.

[1] 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

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 [144, 143]	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 4.5 ICDAR 2015 Robust Reading competition [52] - End-to-end Incidental text recognition

Method	MOTP	MOTA	ATA
proposed method	**69.5**	**59.8**	**41.8**
Stradvision-1	69.2	56.5	28.5
USTB-TexVideo [144, 143]	65.1	45.8	19.8
Deep2Text-I [144, 143]	62.1	35.4	18.6
USTB-TexVideo-II-2 [144, 143]	63.5	50.5	17.8
USTB-TexVideo-II-1 [144, 143]	60.5	21.2	13.8

Table 4.6 ICDAR 2015 Robust Reading competition [52] - End-to-end Video text recognition

Method	Strong			Weak			Generic		
	p	*r*	*f*	*p*	*r*	*f*	*p*	*r*	*f*
VGGMaxBBNet [47]	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 [144, 143]	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 [144, 143]	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

Table 4.7 ICDAR 2015 Robust Reading competition [52] - End-to-end Focused text recognition

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 4.5). 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.

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 [131]) outperformed all participants (see Table 4.6) in all three metrics [54]: 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. [47] trained on significantly more data and the deep-network based StradVision method, which is not published (see Table 4.7).

5 Efficient Unconstrained Scene Text Detector

5.1 FASText Keypoint Detector

The FASText keypoint detector is, as the name suggests, based on the well-known FAST corner detector by Rosten and Drummond [111, 112]. The standard FAST detects certain letters by firing on character 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 doesn't 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 and repeated detections (see Section 5.5.1), which unnecessarily slows down the processing.

Considering we are only interested in detecting character strokes, the FAST detector is modified to introduce two novel keypoint classes: the *Stroke Ending Keypoint (SEK)* matches a stroke ending, whilst the *Stroke Bend Keypoint (SBK)* matches a curved segment of a stroke (see Figure 5.2).

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

$$L(p,x) = \begin{cases} d, & I_x \leq I_p - m & (darker) \\ s, & I_p - m < I_x < I_p + m & (similar) \\ b, & I_x \geq I_p + m & (brighter) \end{cases} \tag{5.1}$$

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

The pixel p is a **Stroke Ending Keypoint** (SEK) if there exists two contiguous partitionings 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 $|P_b| = 12 - |P_s|$), where P_l denotes a contiguous partitioning of the pixels x with the label l. 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

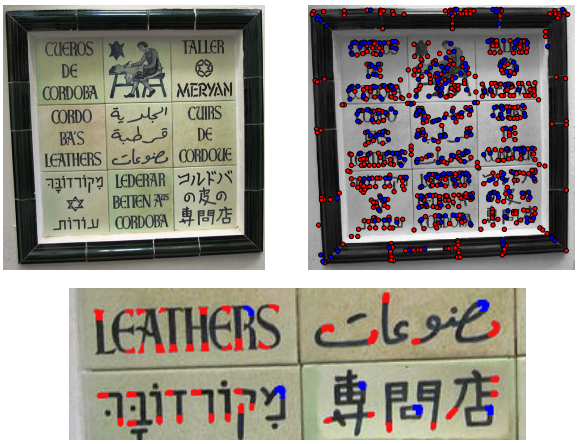


Figure 5.1 The FASText detector output. *Stroke End Keypoints* (SEK) marked red, *Stroke Bend Keypoints* (SBK) marked blue.

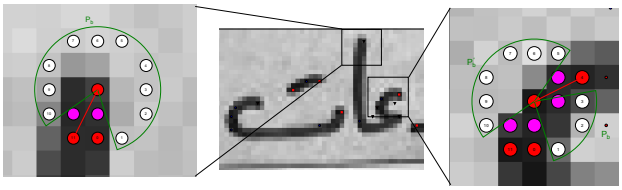


Figure 5.2 The *Stroke Ending Keypoint* (left) and the *Stroke Bend Keypoint* (right). Pixels of the P_s partitioning marked red, pixels of the P_b partitioning marked white, inner pixels P_i for the connectivity test in purple.

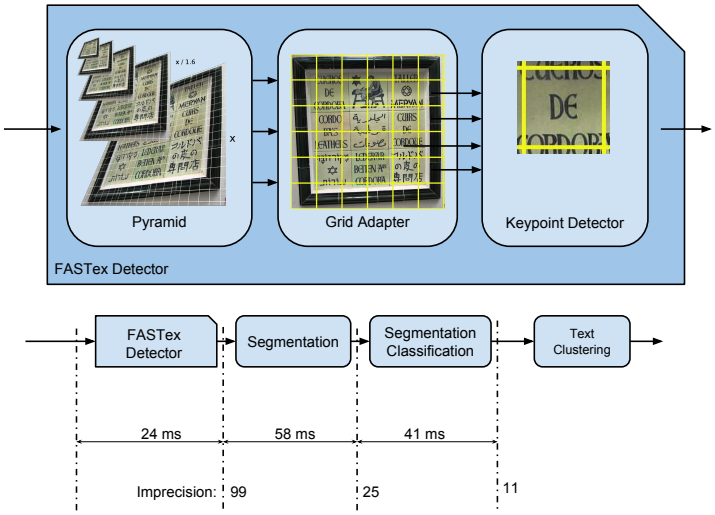


Figure 5.3 The proposed pipeline. The average processing time and the imprecision (number of false and repeated detections) for a 1MPx image denoted below each stage.

positive or negative, depending whether the intensity of the stroke is higher or lower than the background.

Using the same notation, the pixel p is a **Stroke Bend Keypoint** (SBK) if there exists 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 $|P_d| > 6, |P'_d| = 12 - |P_d| - |P_s| - |P'_s|$ (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 very 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 simple rule, which examines the opposite pixels and tests that all opposite pixels are brighter than $I_p + t$ or darker than $I_p - t$. 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 5.2). 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.

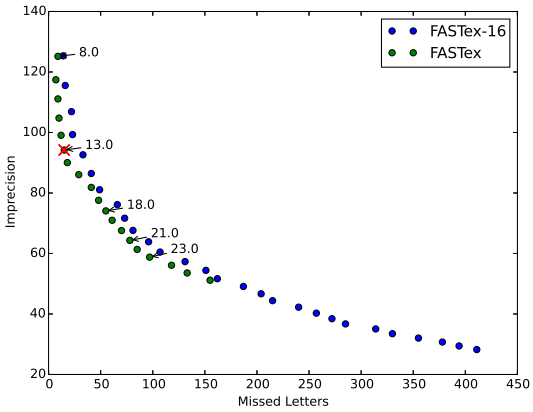


Figure 5.4 The margin parameter m controls the trade-off between imprecision (the number of false and repeated detections) and the number of missed characters. The value used in experiments marked by the red cross, the detector with the circle size of 16 pixels denoted FASText-16.

In order to eliminate FASText keypoints which lie close to each other, a simple non-maximum suppression is performed on a 3×3 neighborhood and 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 optimal values of the detector parameters were found experimentally using the ICDAR 2013 Training dataset [53], which contains 4784 characters in 229 images. A character is considered as 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 a precision-like metric designed to cater for repeated detections of the same character and 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, but it does not suggest what ratio of ground truth characters is actually detected.

The circle size of 12 pixels is the first detector parameter. Its value is lower than the original FAST [111] value of 16 pixels to allow detection of characters (strokes) which are close to each other (see Figure 5.4). Let us note that a circle size smaller than 12 pixels is not possible because of the connectivity test.

The second detector parameter is the margin m, which controls the trade-off between imprecision (the number of false and repeated detections) and the number of missed characters (see Figure 5.4). The optimal margin value was found to be $m = 13$.

Because 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 is the third parameter of the detector and its optimal value was experimentally found to be 1.6 (see Figure 5.5).

The last parameter of the detector is the maximum number of keypoints per image.

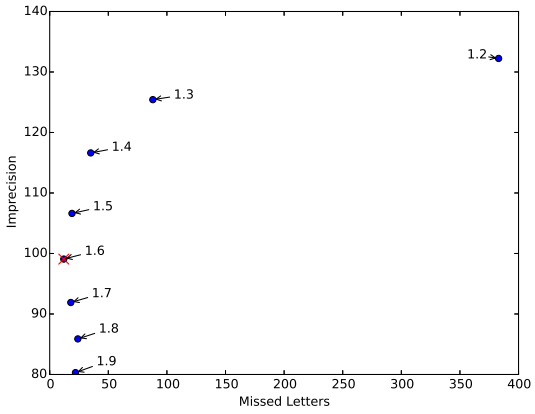


Figure 5.5 The scaling parameter f controls the trade-off between imprecision and the number of missed characters. The value used in experiments marked by the red cross.

An input image is partitioned into uniformly-sized cells (see Figure 5.3) 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 [113].

5.2 Keypoint Segmentation

As successfully demonstrated by the methods based on MSERs [92, 144], 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 below the intensity of the darkest pixel in P_d

$$\theta_p = \min(I_x) - 1 \,|\, x \in P_d \tag{5.2}$$

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

$$\theta_p = \max(I_x) + 1 \,|\, x \in P_b \tag{5.3}$$

The threshold value θ_p is then effectively exploited by a standard flood-fill algorithm [126] to generate a stroke for each FASText keypoint.

5.3 Segmentation Classification

In order to reduce the still relatively high false detection rate of the FASText detector (the average imprecision is 25 segmentations 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.

The classification uses the concept of *text fragments*, where a text fragment can be a single character, a group of characters, a whole word or a part of a character. This allows the classifier to discriminate between text and clutter, regardless of whether a character is only partially detected, or whether a group of characters is detected as one region. As a result, the common assumption of region-based methods (see Section 2.1.2) that one region corresponds to one character is dropped, allowing for a higher recall.

5.3.1 Character Strokes Area

The *Character Strokes 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. In order to estimate the "strokeness" of a region we introduce a novel feature based on *Stroke Support Pixels (SSPs)* which exploits the observation that one can draw any character by taking a brush with a diameter of the stroke width and drawing through certain points of the character (see Figure 5.7) - we refer to such points as *stroke support pixels (SSPs)*. The SSPs have the property that they are in the middle of a character stroke, which we refer to as the *stroke axis*, the distance to the region boundary is half of the stroke width, but unlike skeletons they do not necessary form a connected graph.

Since the area (i.e. the number of pixels) of an ideal stroke is the product of the stroke width and the length of the stroke, the "strokeness" can be estimated by the *Character Strokes Area* (CSA) feature ς which compares the actual area of a region A with the ideal stroke area calculated from the SSPs A_s

$$\varsigma = \frac{A_s}{A} \tag{5.4}$$

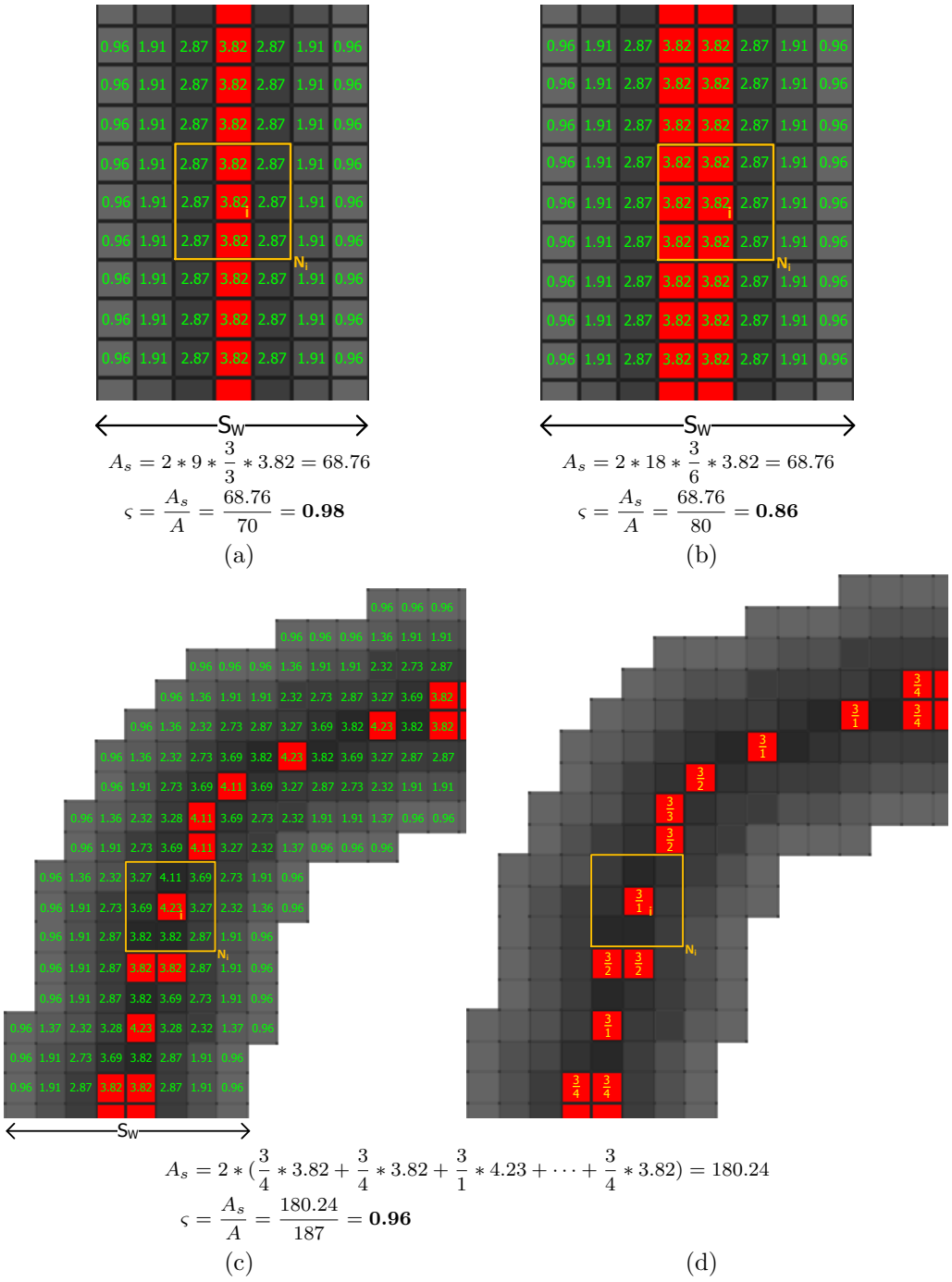
The feature estimates the proportion of region pixels which are part of a character stroke and therefore it allows to efficiently differentiate between text regions (regardless of how many characters they represent - see Figure 5.8) and the background clutter. The feature is efficiently computed from a region distance map, it is invariant to scaling and rotation and it is more robust to noise than methods which aim to estimate a single stroke width value [29] as small pixel changes do not cause unproportional changes to the estimate.

In order to estimate the character strokes area, a distance transform map is calculated for the region binary mask and only pixels corresponding to local distance maxima are considered (see Figure 5.6). These are the *Stroke Support Pixels (SSPs)*, because the pixels determine the position of a latent character stroke axis. In order to estimate the area of the character strokes $\bar{A}_s$, one could simply sum the distances associated with the SSPs

$$\bar{A}_s = 2 \sum_{i \in \mathbf{S}} d_i \tag{5.5}$$

where $\mathbf{S}$ are the SSPs and d_i is the distance of the pixel i from the boundary.

Such an estimate is correct for an straight stroke of an odd width, however it becomes inaccurate for strokes of an even width (because there are two support pixels for a unitary stroke length) or when the support pixels are not connected to each other as a result of stroke curvature, noise at the region boundary or changing stroke width (see Figure 5.6). We therefore propose to compensate the estimate by introducing a weight w_i for each SSP, which ensures normalization to a unitary stroke length by counting



$$A_s = 2 * 9 * \frac{3}{3} * 3.82 = 68.76$$

$$\varsigma = \frac{A_s}{A} = \frac{68.76}{70} = \mathbf{0.98}$$

(a)

$$A_s = 2 * 18 * \frac{3}{6} * 3.82 = 68.76$$

$$\varsigma = \frac{A_s}{A} = \frac{68.76}{80} = \mathbf{0.86}$$

(b)

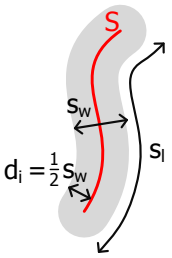
$$A_s = 2 * (\frac{3}{4} * 3.82 + \frac{3}{4} * 3.82 + \frac{3}{1} * 4.23 + \cdots + \frac{3}{4} * 3.82) = 180.24$$

$$\varsigma = \frac{A_s}{A} = \frac{180.24}{187} = \mathbf{0.96}$$

(c)

(d)

Figure 5.6 *Character Strokes Area* (CSA) ς calulation for a straight stroke of an odd (a) and even (b) width and for a curved stroke - distance map d_i (c) and Stroke Support Pixel weights w_i (d). Stroke Support Pixels (SSPs) denoted red



$$A = s_w * s_l \doteq 2 \sum_{i \in \mathbf{S}} d_i$$

Figure 5.7 Area A of an ideal stroke is a product of the stroke width s_w and the length of the stroke s_l. This is approximated by summing double the distances d_i of *Stroke Support Pixels (SSPs)* along the stroke axis $\mathbf{S}$

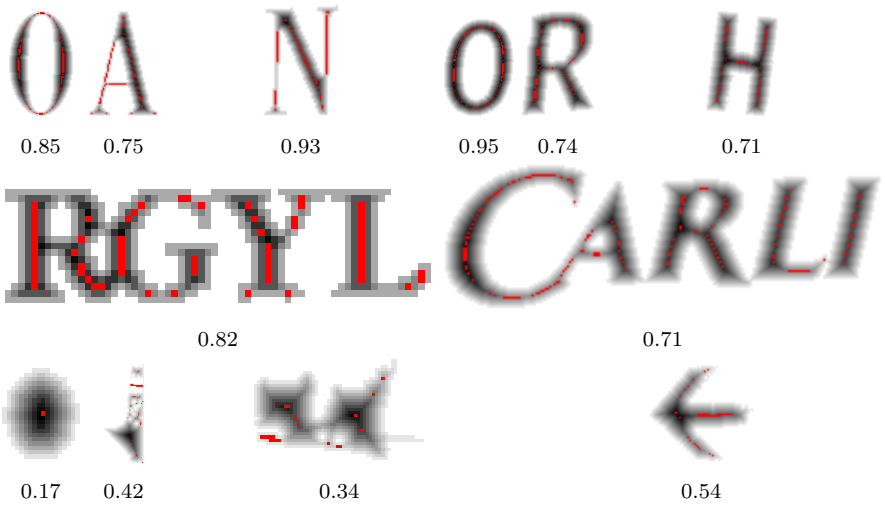


Figure 5.8 Examples of the *Character Strokes Area* (CSA) ς values for character (top row), multi-character (middle row) and background (bottom row) connected components. Distance map denoted by pixel intensity, *Stroke Support Pixels (SSPs)* denoted red

the number of pixels in a 3×3 neighborhood

$$A_s = 2 \sum_{i \in \mathbf{S}} w_i d_i \quad w_i = \frac{3}{|\mathcal{N}_i|} \tag{5.6}$$

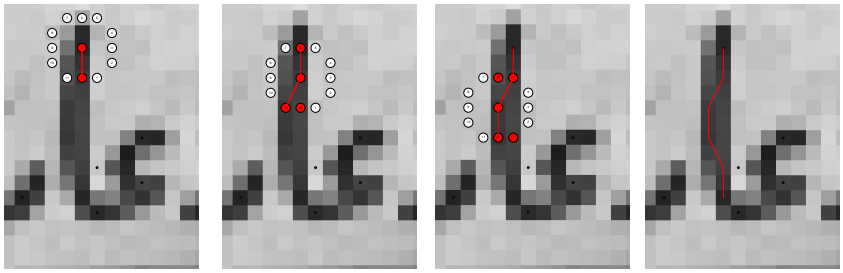
where $|\mathcal{N}_i|$ denotes the number of SSPs within the 3×3 neighborhood of the pixel of i (including the pixel i itself). The numerator value is given by the observation that for a straight stroke, there are 3 support pixels in the 3×3 neighborhood (see Figure 5.6a).

5.3.2 Approximate Character Strokes Area

The main drawback of the CSA feature introduced in Section 5.3.1 is its computational complexity, given by the need to calculate a distance map and employ an iterative non-maxima suppression over the whole region. We therefore also propose an approximate, but significantly faster calculation of the Character Strokes Area feature.

Given a segmentation r, a set of *Stroke Straight Keypoints* (SSK) is found for each Stroke Ending Keypoint (SEK) p which intersects with the segmentation r using the following iterative algorithm (see Figure 5.9):

1. Take the Stroke Ending Keypoint p as the starting point



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

Figure 5.9 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).

	t / region [ms]
Full CSA (Section 5.3.1)	0.895
Approximated CSA (Section 5.3.2)	0.015

Table 5.1 Character Strokes Area calculation time comparison.

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 $|P_b| > 3, |P'_b| = 12 - |P'_b| - |P_s| - |P'_s|$)

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

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

$$A_s(r) = \sum_{p \in \mathrm{SSK}_r} 3|P_s|_p + \sum_{p \in \mathrm{SSB}_r} 3\Big(|P_s|_p + |P_s|'_p\Big) \tag{5.7}$$

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

In the end, four rotation- and scale-invariant features are employed by a Gentle AdaBoost classifier [34] to classify FASText segmentations as either a text fragment (typically a character) or a background clutter: compactness, convex hull area ratio, holes area ratio (all calculated as part of the segmentation process) and the Character Strokes Area (CSA).

The proposed approximate algorithm is almost 60 times faster (see Table 5.1), yet the impact on the classification accuracy is negligible (see Figure 5.10).

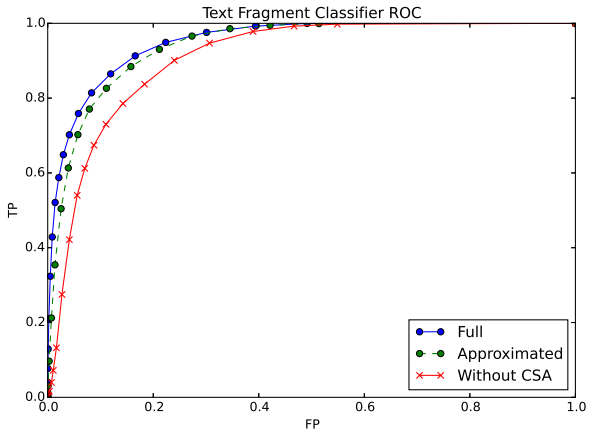


Figure 5.10 The impact of the Character Strokes Area (CSA) feature in the segmentation classification. The full CSA feature (blue), the approximated CSA feature (green), a classifier without the CSA feature (red)

5.4 Text Clustering

In this stage, the unordered set of FASText segmentations 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 segmentations as $\mathcal{R}$. For each segmentation $r_i \in \mathcal{R}$, all segmentations $r_j \in \mathcal{R}$ are found, such that r_i and r_j are neighbors. The segmentation r_i is a *neighbor* of r_j (denoted $N(r_i, r_j)$, if they are sufficiently close to each other (the distance is measured as the distance of their centroids - see Figure 5.11a) and they have a comparable scale

$$N(r_i, r_j) = \begin{cases} 1, & \|c(r_i) - c(r_j)\| < \alpha\sqrt{\max(A(r_i), A(r_j))} \\ & \max\left(\frac{A(r_i)}{A(r_j)}, \frac{A(r_j)}{A(r_i)}\right) < \beta \\ 0, & otherwise \end{cases} \tag{5.8}$$

where $c(r)$ is the centroid of the segmentation r and $A(r)$ is the convex hull area of the segmentation r. The parameter values $\alpha = 4$ and $\beta = 10$ were chosen experimentally and they provide sufficient tolerance for a vast majority of typographical models. Since the segmentation 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 segmentations just in the closest cells. Alternatively, one could use a standard (approximate) nearest-neighbor algorithm.

Each pair of neighboring segmentations then casts a vote for their text direction, where the text direction is given by the line which passes through the two centroids of the neighboring pair (see Figure 5.11b). Drawing inspiration from the well-known Hough transform [11], 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 5.11c).

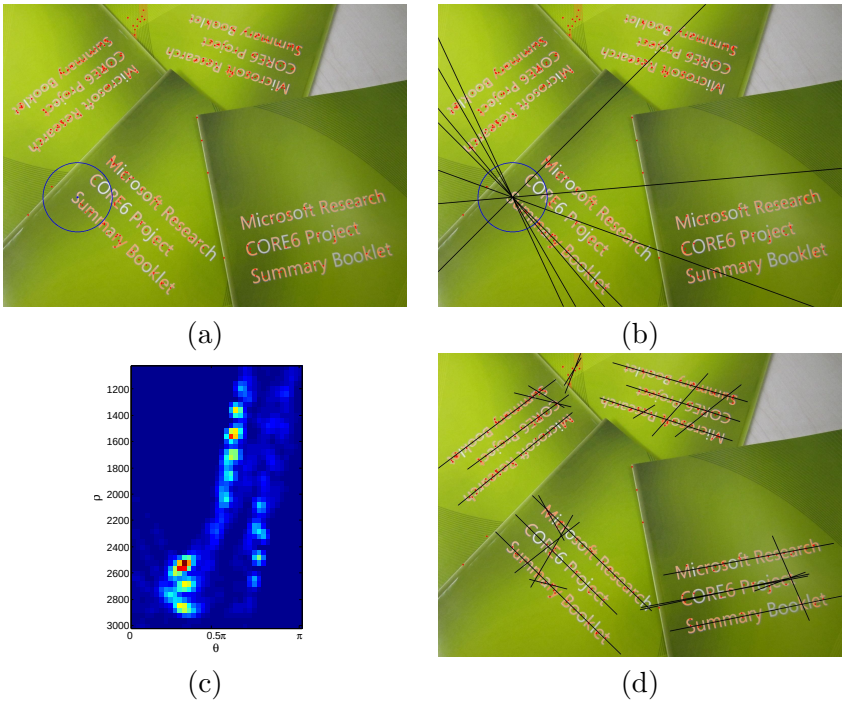


Figure 5.11 Segmentations are clustered to form lines of text. A selected segmentation r and the radius of neighbor search (a). All neighboring pairs of the segmentation r and their corresponding text directions (b). Quantized text direction votes in the (ρ, θ) parameter space (c). Final text line clusters (d).

Each local maxima with its parameters (ρ, θ) then unambiguously induces a text cluster by simply taking all segmentations 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 segmentation can lie on multiple lines with different parameters (ρ, θ), the local maxima are processed in a decreasing order of number of their votes and each segmentation 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 5.11d).

5.5 Experiments

5.5.1 Character Detection

In the first experiment, the character detection ability of the FASText keypoint detector is compared with the existing detectors. The evaluation uses the standard ICDAR 2013 Test dataset (which is commonly used for text localization evaluation - see Section 5.5.2) containing 6393 characters in 255 images annotated on the pixel level, i.e. ground truth character segmentations are provided.

In Table 5.2, keypoints detected by the FASText and the FAST detector [111] are compared by processing all images in the dataset and calculating keypoint statistics for each detector (see Figure 5.13 for a visual comparison on a sample image). The number of total detections is almost identical for both detectors, but the proposed FASText detector detects 3 times more characters (only 111 characters are missed against 328 missed characters by the FAST detector). A character is considered as

Figure 5.12 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.

Figure 5.13 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.

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

Table 5.2 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.

	$\|D\|$	$\frac{\|D\|}{\|GT\|}$	$\|FN\|$	t [ms]
MSER [80]	401972	69.4	1128	328.09
ER detector (Section 4.1)	636729	109.9	657	1068.41
FASText	215325	37.2	857	**82.01**
FASText+AdaBoost	**62394**	**10.8**	1240	122.10

Table 5.3 Detected segmentations in the ICDAR 2013 dataset. The number of segmentations $|D|$, imprecision $\frac{|D|}{|GT|}$, the number of characters without a valid segmentation $|FN|$ and the average time per image t.

missed by a detector, if there is no keypoint whose position coincides with any pixel in the character ground truth segmentation. The FASText detector is also 20% faster than the FAST detector.

In Table 5.3, segmentations produced by the detectors commonly used in scene text localization are compared with the 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 5.3) are included, whilst the FAST detector is not included as it does not provide segmentations. A character is considered as detected by a detector, if the detector produces a segmentation, whose bounding-box overlap with the character ground truth bounding-box is above 60%. If no such segmentation exists, a character is considered as missed.

The FASText detector produces 2 times less segmentations and still detects 25% more characters than the commonly exploited MSER [80] detector and at the same time it is 4 times faster. The FASText detector with the proposed subsequent classification phase produces 7 times less segmentations and is almost 3 times faster than the MSER detector. The number of missed characters is 10% higher than the MSER detector, where the characters incorrectly rejected by the classifier are typically thin letters such as “i” or “l”.

5.5.2 Text Localization and Recognition

In order to evaluate scene text localization ability of the proposed detector, we have adapted the ER-based pipeline from Section 4 and replaced the initial stages (ER detection, character classification and the text line formation) with the proposed detector. The resulting experimental pipeline therefore consists of the FASText detector with the proposed segmentation classification and the text clustering, followed by the local iterative segmentation refinement and character recognition adapted from the original pipeline.

In the first experiment, the pipeline was evaluated on the most cited ICDAR 2013 Robust Reading Dataset [53], which consists of 1189 words and 6393 letters in 255 images. There are many challenging text instances in the dataset (reflections, text

Figure 5.14 Text localization and recognition examples from the ICDAR 2013 dataset.

method	R	P	F	t_l	t_r	**t**
FASText pipeline	69.3	84.0	76.8	**0.15**	0.4	0.55
ER pipeline (Chapter 4)	72.4	81.8	77.1	?	?	0.8
Zamberletti et al. [147]	70.0	85.6	77.0	0.75	N/A	N/A
Yin et al. [144]	68.3	86.3	76.2	0.43	N/A	N/A
ICDAR 2013 winner [53]	66.4	88.5	75.9	?	?	?

Table 5.4 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 **t** (in seconds).

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 (see output examples in the Figure 5.14).

Using the same evaluation protocol as the latest ICDAR 2013 Robust Reading competition [53], the text localization accuracy compares favorably to the state-of-the-art methods (see Table 5.4), whilst the proposed method is significantly faster - in text localization (t_l), the proposed pipeline is 3 times faster than the best method.

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 5.12, 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 line segmentations.

6 Single Shot Text Detection and Recognition

In this chapter, we propose a novel end-to-end framework which simultaneously detects and recognizes text in scene images. As the first contribution, we present a model which is trained for both text detection and recognition in a single learning framework, and we show that such joint model outperforms the combination of state-of-the-art localization and state-of-the-art recognition methods [41, 38].

As the second contribution, we show how the state-of-the-art object detection methods [109, 110] can be extended for text detection and recognition, taking into account specifics of text such as the exponential number of classes (given an alphabet $\mathcal{A}$, there are up to $\mathcal{A}^L$ possible classes, where L denotes maximum text length) and the sensitivity to hidden parameters such as text aspect and rotation.

The method achieves state-of-the-art results on the standard ICDAR 2013 [53] and ICDAR 2015 [52] datasets and the pipeline runs end-to-end at 10 frames per second on a NVidia K80 GPU, which is more than 10 times faster than the fastest methods.

Figure 6.1 The proposed method detects and recognizes text in scene images at 10fps on an NVidia K80 GPU. Ground truth in green, model output in red. The image taken from the ICDAR 2013 dataset [53]

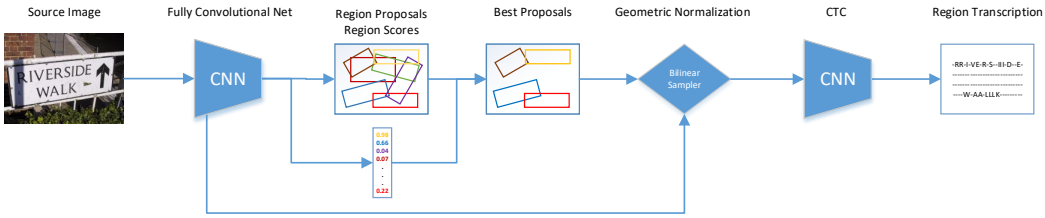


Figure 6.2 Method overview. Text region proposals are generated by a Region Proposal Network [109]. Each region with a sufficient text confidence is then normalized to a variable-width feature tensor by bilinear sampling. Finally, each region is associated with a sequence of characters or rejected as not text.

6.1 Fully Convolutional Network

The proposed model localizes text regions in a given scene image and provides text transcription as a sequence of characters for all regions with text (see Figure 6.2). The model is jointly optimized for both text localization and recognition in an end-to-end training framework.

We adapt the YOLOv2 architecture [109] for its accuracy and significantly lower complexity than the standard VGG-16 architecture [120, 50], as the full VGG-16 architecture requires 30 billion operations just to process a 224×224 (0.05 Mpx) image [109]. Using YOLOv2 architecture allows us to process images with higher resolution, which is a crucial ability for text recognition - processing at higher resolution is required because a 1Mpx scene image may contain text which is 10 pixels high [52], so scaling down the source image would make the text unreadable.

The proposed method uses the first 18 convolutional and 5 max pool layers from the YOLOv2 architecture, which is based on 3×3 convolutional filters, doubling the number of channels after every pooling step and adding 1×1 filters to compress the representations between the 3×3 filters [109]. We remove the fully-connected layers to make the network fully convolutional, so our model final layer has the dimension of $\frac{W}{32} \times \frac{H}{32} \times 1024$, where W a H denote source image width and height [109].

6.2 Region Proposals

Similarly to Faster R-CNN [110] and YOLOv2 [109], we use a Region Proposal Network (RPN) to generate region proposals, but we add rotation r_θ which is crucial for a successful text recognition. At each position of the last convolutional layer, the model predicts k rotated bounding boxes, where for each bounding box r we predict 6 features - its position r_x, r_y, its dimensions r_w, r_h, its rotation r_θ and its score r_p, which captures the probability that the region contains text.

The bounding box position and dimension is encoded with respect to predefined anchor boxes using the logistic activation function, so the actual bounding box position (x, y) and dimension (w, h) in the source image is given as

$$x = \sigma(r_x) + c_x \tag{6.1}$$
$$y = \sigma(r_y) + c_y \tag{6.2}$$
$$w = a_w \exp(r_w) \tag{6.3}$$
$$h = a_h \exp(r_h) \tag{6.4}$$
$$\theta = r_\theta \tag{6.5}$$

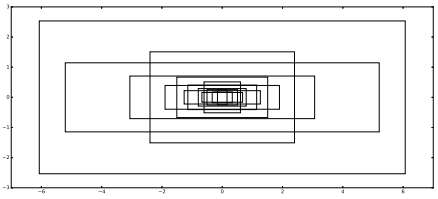


Figure 6.3 Anchor box widths and highs, or equivalently scales and aspects, were obtained by k-means clustering on the training set. Requiring that each ground truth box had intersection-over-union of at least 60% with one anchor box led to $k = 14$ boxes.

where c_x and c_y denote the offset of the cell in the last convolutional layer and a_w and a_h denote the predefined height and width of the anchor box a. The rotation θ of the bounding box is predicted directly by r_θ.

We followed the approach of Redmon *et al.* [109] and found suitable anchor box scales and aspects by k-means clustering on the aggregated training set (see Section 6.5). Requiring the anchor boxes to have at least 60% intersection-over-union with the ground truth led to $k = 14$ different anchor boxes dimensions (see Figure 6.3).

For every image, the RPN produces $\frac{W}{32} \times \frac{H}{32} \times 6k$ boxes (there are 6 estimated parameters for every anchor - x,y,w,h,θ and the text score r_p), which would make subsequent computation too complex and it is therefore necessary to only select a subset.

In the training stage, we use the YOLOv2 approach [109] by taking all positive and negative samples in the source image, where every 20 batches we randomly change the input dimension size into one of $\{352, 416, 480, 544, 608\}$. A positive sample is the region with the highest intersection over union with the ground truth, the other intersecting regions are negatives.

At runtime, we found the best approach is to take all regions with the score r_p above a certain threshold $p_{\min}$ and to postpone the non-maxima suppression after the recognition stage, because regions with very similar r_p scores could produce very different transcriptions, and therefore selecting the region with the highest r_p at this stage would not always correspond to the correct transcription (for example, in some cases a region containing letters "TALY" may have slightly higher score r_p than a region containing the full word "ITALY"). We found the value $p_{\min} = 0.1$ to be a reasonable trade-off between accuracy and speed.

Type	Channels	Size/Stride	Dim/Act
input	C	-	$\overline{W} \times 32$
conv	32	3×3	leaky ReLU
conv	32	3×3	leaky ReLU
maxpool		$2 \times 2/2$	$\overline{W}/2 \times 16$
conv	64	3×3	leaky ReLU
BatchNorm			
recurrent conv	64	3×3	leaky ReLU
maxpool		$2 \times 2/2$	$\overline{W}/4 \times 8$
conv	128	3×3	leaky ReLU
BatchNorm			
recurrent conv	128	3×3	leaky ReLU
maxpool		$2 \times 2/2 \times 1$	$\overline{W}/4 \times 4$
conv	256	3×3	leaky ReLU
BatchNorm			
recurrent conv	256	3×3	leaky ReLU
maxpool		$2 \times 2/2 \times 1$	$\overline{W}/4 \times 2$
conv	512	3×2	leaky ReLU
conv	512	5×1	leaky ReLU
conv	$\lvert\hat{\mathcal{A}}\rvert$	7×1	$\overline{W}/4 \times 1$
log softmax			

Table 6.1 Fully-Convolutional Network for Text Recognition

6.3 Bilinear Sampling

Each region detected in the previous stage has a different size and rotation and it is therefore necessary to map the features into a tensor of canonical dimensions, which can be used in recognition.

Faster R-CNN [110] uses the RoI pooling approach of Girshick [35], where a $w \times h \times C$ region is mapped onto a fixed-sized $W' \times H' \times C$ grid ($7 \times 7 \times 1024$ in their implementation), where each cell takes the maximum activation of the $\frac{w}{W} \times \frac{h}{H}$ cells in the underlying feature layer.

In our model, we instead use bilinear sampling [48, 50] to map a $w \times h \times C$ region from the source image into a fixed-height $\frac{wH'}{h} \times H' \times C$ tensor ($H' = 32$). This feature representation has a key advantage over the standard RoI approach as it allows the network to normalize rotation and scale, but at the same to persist the aspect and positioning of individual characters, which is crucial for text recognition accuracy (see Section 6.4).

Given the detected region features $\mathbf{U} \in \mathbb{R}^{w \times h \times C}$, they are mapped into a fixed-height tensor $\mathbf{V} \in \mathbb{R}^{\frac{wH'}{h} \times H' \times C}$ as

$$\mathbf{V}^c_{x',y'} = \sum_{x=1}^{w} \sum_{y=1}^{h} \mathbf{U}^c_{x,y} \kappa(x - \mathcal{T}_x(x')) \kappa(y - \mathcal{T}_y(y')) \tag{6.6}$$

where κ is the bilinear sampling kernel $\kappa(v) = \max(0, 1 - |v|)$ and $\mathcal{T}$ is a point-wise coordinate transformation, which projects co-ordinates x' and y' of the fixed-sized tensor $\mathbf{V}$ to the co-ordinates x and y in the detected region features tensor $\mathbf{U}$.

The transformation allows for shift and scaling in x- and y- axes and rotation and its parameters are taken directly from the region parameters (see Section 6.2).

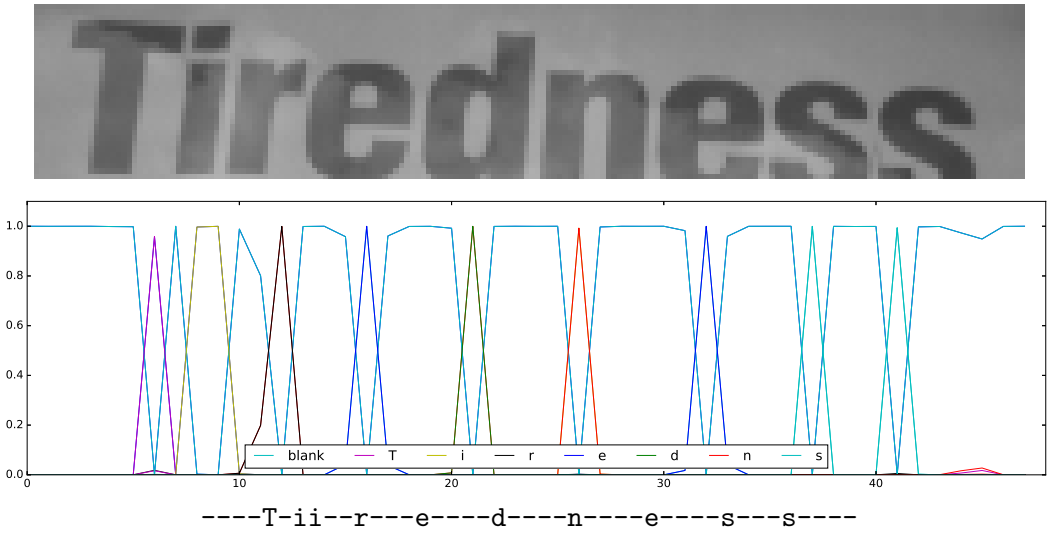


Figure 6.4 Text recognition using Connectionist Temporal Classification. Input $\overline{W} \times 32$ region (top), CTC output $\frac{\overline{W}}{4} \times |\hat{\mathcal{A}}|$ as the most probable class at given column (middle) and the resulting sequence (bottom)

6.4 Text Recognition

Given the normalized region from the source image, each region is associated with a sequence of characters or rejected as not text in the following process.

The main problem one has to address in this step is the fact, that text regions of different sizes have to be mapped to character sequences of different lengths. Traditionally, the issue is solved by resizing the input to a fixed-sized matrix (typically 100×32 [47, 119]) and the input is then classified by either making every possible character sequence (*i.e.* every word) a separate class of its own [47, 41], thus requiring a list of all possible outputs in the training stage, or by having multiple independent classifiers, where each classifier predicts the character at predefined position [45].

Our model exploits a novel fully-convolutional network (see Table 6.1), which takes a variable-width feature tensor $\overline{W} \times H' \times C$ as an input ($\overline{W} = \frac{wH'}{h}$) and outputs a matrix $\frac{\overline{W}}{4} \times |\hat{\mathcal{A}}|$, where $\mathcal{A}$ is the alphabet (*e.g.* all English characters). The matrix height is fixed (it's the number of character classes), but its width grows with the width of the source region and therefore with the length of the expected character sequence.

As a result, a single classifier is used regardless of the position of the character in the word (in contrast to Jaderberg *et al.* [45], where there is an independent classifier for the character "A" as the first character in the word, an independent classifier for the character "A" as the second character in the word, etc). The model also does not require prior knowledge of all words to be detected in the training stage, in contrast to the separate class per character sequence formulation [47].

The model uses Connectionist Temporal Classification (CTC) [40, 119] to transform variable-width feature tensor into a conditional probability distribution over label sequences. The distribution is then used to select the most probable labelling sequence for the text region (see Figure 6.4).

Let $\mathbf{y} = y_1, y_2, \cdots, y_n$ denote the vector of network outputs of length n from an alphabet $\mathcal{A}$ extended with a blank symbol "–".

The probability of a *path* π is then given as

$$\begin{aligned} p(\pi|\mathbf{y}) &= \prod_{i=1}^{n} y^i_{\pi_i}, \quad \pi \in \hat{\mathcal{A}}^n \\ \hat{\mathcal{A}} &= \mathcal{A} \cup \{-\} \end{aligned} \tag{6.7}$$

where $y^i_{\pi_i}$ denotes the output probability of the network predicting the label π_i at the position i (*i.e.* the output of the final softmax layer in Table 6.1).

Let us further define a many-to-one mapping $\mathcal{B} : \hat{\mathcal{A}}^n \mapsto \mathcal{A}^{\leq n}$, where $\hat{\mathcal{A}}^{\leq n}$ is the set of all sequences of shorter or equal in length. The mapping $\mathcal{B}$ removes all blanks and repeated labels, which corresponds to outputting a new label every time the label prediction changes. For example,

$$\begin{aligned} \mathcal{B}(-\mathrm{ww}-\mathrm{al}-\mathrm{k}) &= \mathcal{B}(\mathrm{wwaaa}-\mathrm{l}-\mathrm{k}-) = \mathrm{walk} \\ \mathcal{B}(-\mathrm{f}-\mathrm{oo}-\mathrm{o}--\mathrm{d}) &= \mathcal{B}(\mathrm{ffoo}-\mathrm{ooo}-\mathrm{d}) = \mathrm{food} \end{aligned}$$

The conditional probability of observing the output sequence $\mathbf{w}$ is then given as

$$p(\mathbf{w}|\mathbf{y}) = \sum_{\pi:\, \mathcal{B}(\pi)=\mathbf{w}} p(\pi|\mathbf{y}), \quad \mathbf{w} \in \mathcal{A}^{\leq n} \tag{6.8}$$

In training, an objective function that maximizes the log likelihood of target labellings $p(\mathbf{w}|\mathbf{y})$ is used [40]. In every training step, the probability $p(\mathbf{w}_{\mathrm{gt}}|\mathbf{y})$ of every text region in the mini-batch is efficiently calculated using a forward-backward algorithm similar to HMMs training [107] and the objective function derivatives are used to update network weights, using the standard back-propagation algorithm ($\mathbf{w}_{\mathrm{gt}}$ denotes the ground truth transcription of the text region).

At test time, the classification output $\mathbf{w}^*$ should be given by the most probable path $p(\mathbf{w}|\mathbf{y})$, which unfortunately is not tractable, and therefore we adapt the approximate approach [40] of taking the most probable labelling

$$\mathbf{w}^* \approx \mathcal{B}\big(\operatorname{argmax} p(\pi|\mathbf{y})\big) \tag{6.9}$$

At the end of this process, each text region in the image has an associated content in the form of a character sequence, or it is rejected as not text when all the labels are blank.

The model typically produces many different boxes for a single text area in the image, we therefore suppress overlapping boxes by a standard non-maxima suppression algorithm based on the text recognition confidence, which is the $p(\mathbf{w}^*|\mathbf{y})$ normalized by the text length.

6.5 Training

We pre-train the detection CNN using the SynthText dataset [41] (800,000 synthetic scene images with multiple words per image) for 3 epochs, with weights initialized from ImageNet [109]. The recognition CNN is pre-trained on the Synthetic Word dataset [45] (9 million synthetic cropped word images) for 3 epochs, with weights randomly initialized from the $\mathcal{N}(0, 1)$ distribution.

As the final step, we train both networks simultaneously for 3 epochs on a combined dataset consisting of the SynthText dataset, the Synthetic Word dataset, the ICDAR

Figure 6.5 End-to-end scene text recognition samples from the ICDAR 2013 dataset. Model output in red, ground truth in green. Note that in some cases (*e.g.* top-right) text is correctly recognized even though the bounding IoU with the ground truth is less than 80%, which would be required by the text localization protocol [53]. Best viewed zoomed in color

	end-to-end			word spotting			speed
	strong	weak	generic	strong	weak	generic	fps
Deep2Text [145]	0.81	0.79	0.77	0.85	0.83	0.79	1.0
FASText (Chapter 5)	0.77	0.63	0.54	0.85	0.66	0.57	1.0
StradVision [52]	0.81	0.79	0.67	0.84	0.83	0.70	?
Jaderberg *et al.* [47]	0.86	-	-	0.90	0.76	-	*0.3
Gupta *et al.* [41]	-	-	-	-	0.85	-	*0.4
proposed method	**0.89**	**0.86**	**0.77**	**0.92**	**0.89**	**0.81**	***10.0**

Table 6.2 ICDAR 2013 dataset - End-to-end scene text recognition accuracy (f-measure), depending on the lexicon size and whether digits are excluded from the evaluation (denoted as *word spotting*). Speed reports for methods using a GPU marked with an asterisk

	end-to-end			word spotting			speed
	strong	weak	generic	strong	weak	generic	fps
FASText (Chapter 5)	0.35	0.20	0.16	0.37	0.21	0.16	1.0
Stradvision [52]	0.44	-	-	0.46	-	-	?
TextProposals [38, 47]	0.53	0.50	0.47	0.56	0.52	0.50	0.2
proposed method	**0.54**	**0.51**	**0.47**	**0.58**	**0.53**	**0.51**	***9.0**

Table 6.3 ICDAR 2015 dataset - End-to-end scene text recognition accuracy (f-measure). Speed reports for methods using a GPU marked with an asterisk

2013 Training dataset [53] (229 scene images captured by a professional camera) and the ICDAR 2015 Training dataset [52] (1000 scene images captured by Google Glass). For every image, we randomly crop up to 30% of its width and height. We use standard Stochastic Gradient Descent with momentum 0.9 and learning rate 10^{-3}, divided by 10 after each epoch. One mini-batch takes about 500ms on a NVidia K80 GPU.

Note that in preparation of this paper, we unfortunately did not have enough time and available computing resources to let the training process run for more epochs and to experiment with the traditional deep learning tricks; despite this, the model still performs well and compares favorably to the state of the art.

6.6 Experiments

We trained our model once and then evaluated its accuracy on two standard datasets. We follow the standard ICDAR Robust Reading Competition protocol [53, 52] for the End to End task, where the objective is to localize and recognize all words in the image in a single step.

In the ICDAR evaluation schema, each image in the test set is associated with a list of words (lexicon), which contains the words that the method should localize and recognize, as well as an increasing number of random "distractor" words. There are three sizes of lists provided with each image, depending how heavily contextualized their content is to the specific image:

- *strongly contextualized* - 100 words specific to each image, contains all words in the image and the remaining words are "distractors"
- *weakly contextualized* - all words in the testing set, same list for every image
- *generic* - all words in the testing set plus 90k English words

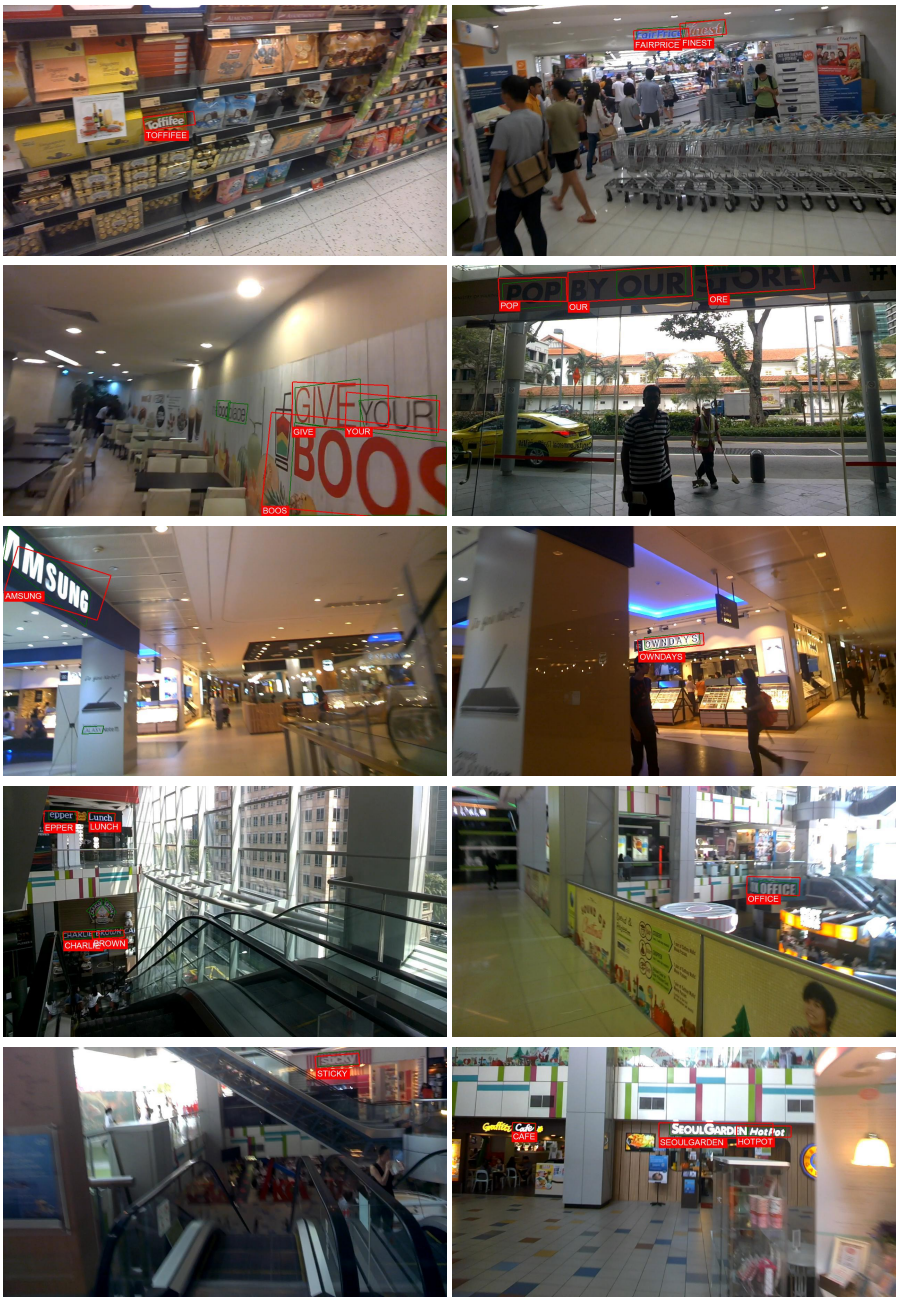


Figure 6.6 End-to-end scene text recognition samples from the ICDAR 2015 dataset. Model output in red, ground truth in green. Best viewed zoomed in color

Figure 6.7 *All the images* of the ICDAR 2013 Testing set where the proposed method fails to correctly recognize any text (*i.e.* images with 0% recall)

A word is considered as correctly recognized, when its Intersection-over-Union (IoU) with the ground truth is above 0.5 and the transcription is identical, using case-insensitive comparison [52].

6.6.1 ICDAR 2013 dataset

The ICDAR 2013 Dataset [53] is the most-frequently cited dataset for scene text evaluation. It consists of 255 testing images with 716 annotated words, the images were taken by a professional camera so text is typically horizontal and the camera is almost always aimed at it. The dataset is sometimes referred to as the *Focused Scene Text* dataset.

The proposed model achieves state-of-the-art text recognition accuracy (see Table 6.2) for all 3 lexicon sizes. In the *end-to-end* set up, where all lexicon words plus all digits in an image should be recognized, the maximal f-measure it achieves is 0.89/0.86/0.77 for strongly, weakly and generally contextualized lexicons respectively. Each image is first resized to 544×544 pixels, the average processing time is 100ms per image on a NVidia K80 GPU for the whole pipeline.

While training on the same training data, our model outperforms the combination of the state-of-the-art localization method of Gupta *et al.* [41] with the state-of-the-art recognition method of Jaderberg *et al.* [47] by at least 3 per cent points on every measure, thus demonstrating the advantage of the joint training for the end-to-end task of our model. It is also more than 20 times faster than the method of Gupta *et al.* [41].

Let us further note that our model would not be considered as a state-of-the-art text localization method according to the text localization evaluation protocol, because at least a 80% intersection-over-union with bounding boxes created by human annotators is required. Our method in contrast does not always achieve the required 80% overlap, but it is still mostly able to recognize the text correctly even when the overlap is lower (see Figure 6.5).

Figure 6.8 Main failure modes on the ICDAR 2015 dataset. Blurred and noisy text (top), vertical text (top) and small text (bottom). Best viewed zoomed in color

We argue that evaluating methods purely on text localization accuracy without subsequent recognition is not very informative, because the text localization "accuracy" only aims to fit the way human annotators create bounding boxes around text, but it does not give any estimates on how well a text recognition phase would read text post a successful localization, which should be the prime objective of the text localization metrics.

The main limitation of the proposed model are single characters or short snippets of digits and characters (see Figure 6.7), which may be partially caused by the fact that such examples are not very frequent in the training set.

6.6.2 ICDAR 2015 dataset

The *ICDAR 2015 dataset* was introduced in the ICDAR 2015 Robust Reading Competition [52]. The images were collected by people who were wearing Google Glass devices and walking in Singapore, and then subsequently all images with text were selected and annotated. The images in the dataset were taken "not having text in mind", therefore text is much smaller and the images contain a high variability of text fonts and sizes. They also include many realistic effects - *e.g.* occlusion, perspective distortion, blur or noise, so as a result the dataset is significantly more challenging than the ICDAR 2013 dataset (Section 6.6.1), which contains typically large horizontal text.

The proposed model achieves state-of-the-art end-to-end text recognition accuracy (see Table 6.3 and Figure 6.6) for all 3 lexicon sizes. In our experiments, the average processing time was 110ms per image on a NVidia K80 GPU (the image is first resized to 608×608 pixels), which makes the proposed model 45 times faster than currently the best published method of Gomez *et al.* [38]

The main failure mode of the proposed method is blurry or noisy text (see Figure 6.8), which are effects not present in the training set (Section 6.5). The method also often fails to detect small text (less than 15 pixels high), which again is due to the lack of such samples in the training stage.

7 Results

7.1 Applications

7.1.1 TextSpotter

The methods presented in Chapter 4 and Chapter 5 were implemented into a software package TextSpotter. The package is implemented in C++ using STL libraries, which makes the code easily portable to Windows, Linux and mobile devices (see Section 7.1.2). The package uses OpenCV library [16] for low-level image processing and classification tasks.

For demonstration purposes, an online demonstration website was set up[1], so that anyone can test methods' capabilities using data of their own choice (see Figure 7.1).

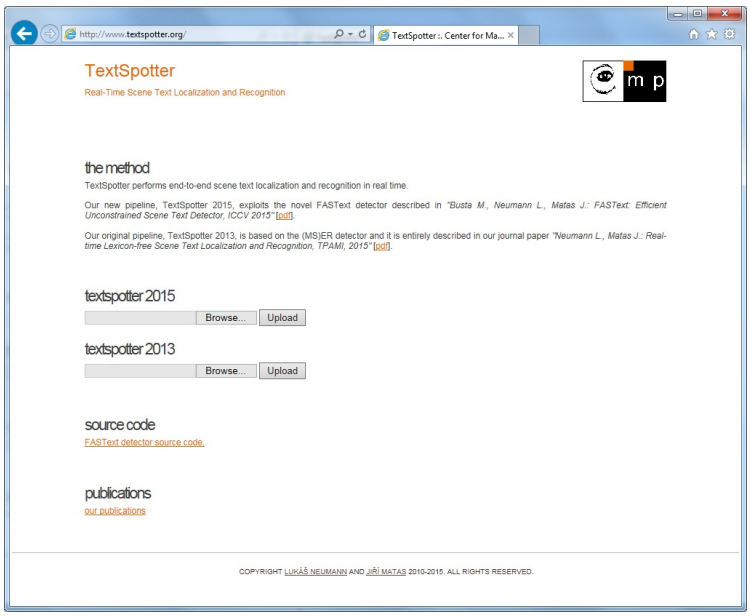


Figure 7.1 Text Spotter on-line demo

On the website, the pipeline based on the ER detection (Chapter 4) is denoted as *TextSpotter 2013*, and the FASText pipeline (Chapter 5) is denoted as *TextSpotter 2015*.

7.1.2 Mobile Application for Translation

Text Lens is a mobile application for Android devices which uses mobile phone camera to capture images and then detects and recognizes text using the scene text recognition method described in Chapter 4. Text Lens runs fully on the mobile device without any need for an Internet connection (the only time Internet connection is required is when offline dictionary is not sufficient and the user requires complex translation by an online service).

[1] http://www.textspotter.org/

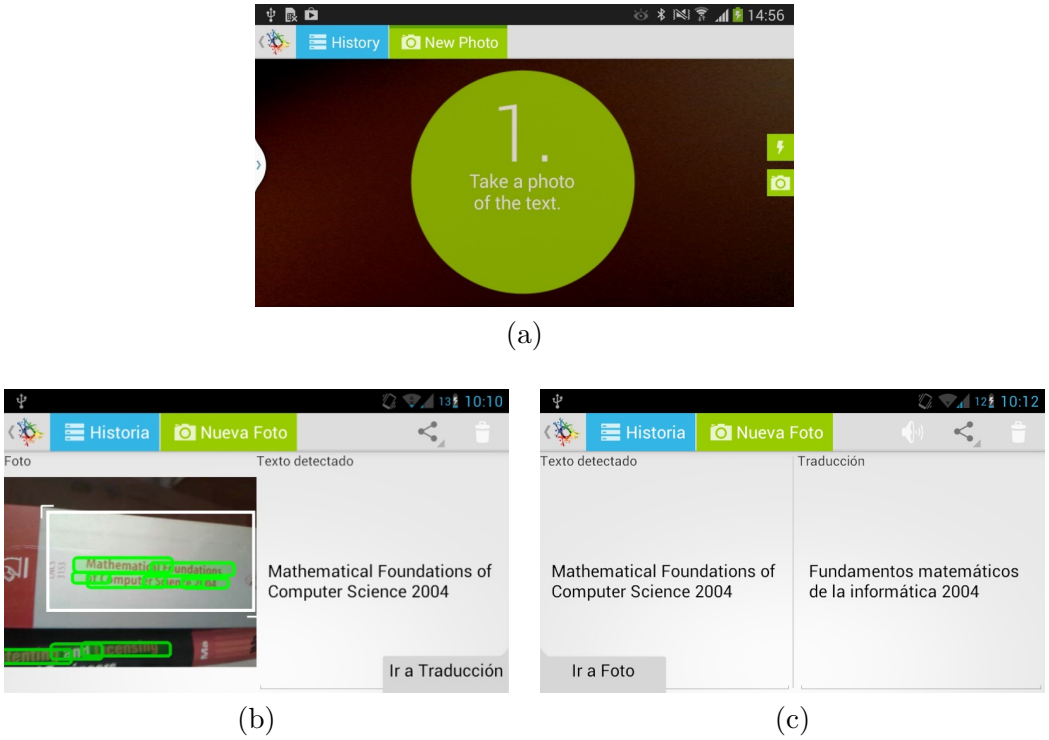


Figure 7.2 The mobile application for automated text translation. Initial screen (a), text detection and recognition using our method (b) and its translation from English to Spanish (c)

When launched, a user is presented with a video stream from the mobile phone camera (in the same way that a standard photo-taking application works). When the user touches the screen, a photo is taken and it is sent to the scene text recognition algorithm. The algorithm detects all text areas present in the image, highlights them with a green rectangle and outputs their textual content, which then a user can chose to translate (see Figure 7.2).

This is significantly faster than typing in the text and it is especially useful in situations when a user is not familiar with the alphabet of the text which he is trying to translate (for example a European tourist in China trying to translate Chinese text).

7.2 Available Code

7.2.1 FASText Detector

We have released the source code for the keypoint detector and the text clustering algorithm presented in Chapter 5. The code is publicly available at GitHub [2], including the trained classification model described in Section 5.3.

7.2.2 ER Detector

The Extremal Region (ER) detection and classification using incrementally computed descriptors presented in Section 4.1.2 was re-implemented by Lluis Gomez into the OpenCV 3.0 library [9, 16, 36], the most popular computer vision library, solely based

[2] https://github.com/MichalBusta/FASText

on the method's description [91]. Although the code is completely independent from the one used in this thesis, the reported results [37] are comparable to the results presented in this thesis. Note that the reported results are not completely identical, because different training data and a different classifier was used.

8 Conclusion

In this thesis, the problem of *Scene Text Localization and Recognition* was studied. Three different methods were proposed in the course of the research, each one advancing the state of the art and improving the accuracy.

The first method (see Chapter 4) detects individual characters as Extremal Regions (ER), where the probability of each ER being a character is estimated using novel features with $O(1)$ 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. Each character is recognized individually by an OCR classifier trained on synthetic fonts and the most probable character sequence is selected by dynamic programming in the very last stage of the processing, when context of each character in a text line is known.

To our knowledge, the method was the first published method [88] to address the complete problem of scene text localization and recognition as a whole - all previous work in the literature focused solely on different subproblems, such as only detecting positions of text or only recognizing text previously found by a human annotator. The method's low complexity also allowed for real-time image processing, as the first method in the literature [91]. The efficiency and the ability to automatically process an image or a video in real time and to output text in a standard digital format allowed creation of many practical applications, such as a mobile translation tool (see Chapter 7). The ER detector itself became the de-facto standard component of scene text localization methods - in the ICDAR 2015 Robust Reading Competition, 20 out of 22 participating methods including the winner use the ER detector [52].

Observing that characters in virtually any script consist of strokes, a novel FASText detector was proposed (see Chapter 5). The FASText detector finds text fragments (characters, parts of characters or character groups) irrespective to text orientation, scale or script and it is significantly faster and produces significantly less false detections than the commonly used ER detector. Additionally, an efficient text clustering algorithm based on text direction voting is proposed, which as well as the previous stages is scale- and rotation- invariant and supports a wide variety of scripts and fonts.

The third method exploits a deep-learning model (Chapter 6), which is trained for both text detection and recognition in a single trainable pipeline. The method localizes and recognizes text in an image in a single feed-forward pass, it is trained purely on synthetic data so it does not require obtaining expensive human annotations for training and it achieves state-of-the-art accuracy in the end-to-end text recognition on two standard datasets, whilst being an order of magnitude faster than the previous methods - the whole pipeline runs at 10 frames per second on a NVidia K80 GPU. We also show the advantage of the joint training for the end-to-end task, by outperforming the ad-hoc combination of the state-of-the-art localization and state-of-the-art recognition methods, while exploiting the same training data.

The research has been supported by a prestigious Google PhD Fellowship, the conference papers describing the methods have been awarded the **ICDAR 2013 Best Student Paper Award** [93] and the **ICDAR 2015 Best Paper Award** [94]. The presented work currently has over 1200 citations in Google Scholar and 500 citations in the Web of Science.

Bibliography

[1] http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/. 24

[2] http://rrc.cvc.uab.es/. 25, 26

[3] http://vision.ucsd.edu/~kai/svt/. 27

[4] https://vision.cornell.edu/se3/coco-text/. 28

[5] http://www.robots.ox.ac.uk/~vgg/data/text/. 28

[6] http://hunspell.github.io//. 28

[7] http://www.robots.ox.ac.uk/~vgg/data/scenetext/. 29

[8] http://cvit.iiit.ac.in/projects/SceneTextUnderstanding/. 30

[9] http://docs.opencv.org/3.0-beta/modules/text/doc/erfilter.html. 82

[10] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2552–2566, 2014. 19, 20

[11] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. 64

[12] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002. 18

[13] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 26–33. IEEE, 2005. 18

[14] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. PhotoOCR: reading text in uncontrolled conditions. ICCV, 2013. 19

[15] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007. 19, 21, 22

[16] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008. 81, 82

[17] M. Bušta, L. Neumann, and J. Matas. Fastext: Efficient unconstrained scene text detector. In *2015 IEEE International Conference on Computer Vision (ICCV 2015)*, pages 1206–1214, California, US, December 2015. IEEE. 6

[18] M. Busta, L. Neumann, and J. Matas. Deep TextSpotter: An end-to-end trainable scene text localization and recognition framework. in review. 6

[19] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986. 12

[20] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. *CVPR*, 2:366–373, 2004. 10, 11

[21] H. Cheng, X. Jiang, Y. Sun, and J. Wang. Color image segmentation: advances and prospects. *Pattern Recognition*, 34(12):2259 – 2281, 2001. 12, 31

[22] A. Clavelli, D. Karatzas, and J. Lladós. A framework for the assessment of text extraction algorithms on complex colour images. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 19–26. ACM, 2010. 14, 15, 16

[23] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 440 –445, sept. 2011. 11

[24] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines*. Cambridge University Press, March 2000. 12, 21, 35

[25] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005. 18, 19

[26] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. *VISAPP*, 05-08 February 2009, 2009. 18, 24

[27] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014. 22

[28] M. Donoser, H. Bischof, and S. Wagner. Using web search engines to improve text recognition. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008. 13

[29] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR 2010*, pages 2963 –2970, 6 2010. 12, 13, 60

[30] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 21

[31] A. Evgeniou and M. Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007. 13

[32] J. Fabrizio, B. Marcotegui, and M. Cord. Text segmentation in natural scenes using toggle-mapping. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 2373–2376. IEEE, 2009. 16, 17

[33] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, C-22(1):67 – 92, jan. 1973. 21

[34] 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. 63

[35] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 73

[36] L. Gomez and D. Karatzas. Multi-script text extraction from natural scenes. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 467–471. IEEE, 2013. 82

[37] L. Gómez and D. Karatzas. Scene text recognition: No country for old men? In *Asian Conference on Computer Vision*, pages 157–168. Springer International Publishing, 2014. 3, 83

[38] L. Gomez-Bigorda and D. Karatzas. Textproposals: A text-specific selective search algorithm for word spotting in the wild. *arXiv preprint arXiv:1604.02619*, 2016. 70, 77, 80

[39] A. Gordo. Supervised mid-level features for word image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2956–2964, 2015. 20

[40] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006. 20, 74, 75

[41] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6, 14, 29, 70, 74, 75, 77, 79

[42] T. He, W. Huang, Y. Qiao, and J. Yao. Text-attentional convolutional neural network for scene text detection. *IEEE Transactions on Image Processing*, 25(6):2529–2541, 2016. 13

[43] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 20

[44] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced mser trees. In *European Conference on Computer Vision*, pages 497–511. Springer, 2014. 13

[45] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *NIPS Deep Learning Workshop 2014*, 2014. 6, 22, 28, 29, 74, 75

[46] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. In *ICLR 2015*, 2015. 23

[47] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016. 22, 28, 54, 55, 74, 77, 79

[48] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 73

[49] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *European conference on computer vision*, pages 512–528. Springer, 2014. 3, 22

[50] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4565–4574, 2016. 71, 73

[51] L. Kang, Y. Li, and D. Doermann. Orientation robust text line detection in natural images. In *CVPR 2014*, pages 4034–4041, 2014. 13

[52] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, B. Andrew, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 robust reading competition. In *ICDAR 2015*, pages 1156–1160. IEEE, 2015. 7, 8, 10, 21, 25, 26, 53, 54, 70, 71, 77, 79, 80, 84

[53] 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. 8, 10, 13, 14, 17, 19, 25, 47, 48, 58, 67, 69, 70, 76, 77, 79

[54] B. Keni and S. Rainer. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008, 2008. 55

[55] E. Kim, S. Lee, and J. Kim. Scene text extraction using focus of mobile camera. In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 166 –170, july 2009. 17

[56] K. Kim, H. Byun, Y. Song, Y. Choi, S. Chi, K. Kim, and Y. Chung. Scene text extraction in natural scene images using hierarchical feature combining and verification. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 679 – 682 Vol.2, aug. 2004. 12

[57] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Higher-order correlation clustering for image segmentation. In *Advances in neural information processing systems*, pages 1530–1538, 2011. 13

[58] D. Kumar and A. Ramakrishnan. OTCYMIST: Otsu-canny minimal spanning tree for born-digital images. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 389–393. IEEE, 2012. 17

[59] J. Lafferty. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289. Morgan Kaufmann, 2001. 12, 19

[60] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005. 18

[61] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 13, 22

[62] 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, 2014. 18

[63] C.-Y. Lee and S. Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 20

[64] J.-J. Lee, P.-H. Lee, S.-W. Lee, A. Yuille, and C. Koch. Adaboost for text detection in natural scene. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 429 –434, sept. 2011. 11

[65] S. Lee, M. S. Cho, K. Jung, and J. H. Kim. Scene text extraction with edge constraint and text collinearity. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3983–3986. IEEE, 2010. 30

[66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966. 18

[67] C. Li, X. Ding, and Y. Wu. Automatic text location in natural scene images. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 1069 –1073, 2001. 12

[68] L. Li and C. L. Tan. Character recognition under severe perspective distortion. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008. 17

[69] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015. 20

[70] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. *arXiv preprint arXiv:1611.06779*, 2016. 14

[71] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 28

[72] X. Lin. Reliable OCR solution for digital content re-mastering. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Dec. 2001. 8

[73] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: investigation of normalization and feature extraction techniques. *Pattern Recognition*, 37(2):265 – 279, 2004. 42

[74] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016. 14

[75] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. IEEE, 1999. 18

[76] S. M. Lucas. Text locating competition results. *Document Analysis and Recognition, International Conference on*, 0:80–85, 2005. 11, 25

[77] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. In *ICDAR 2003*, page 682, 2003. 24

[78] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. *arXiv preprint arXiv:1703.01086*, 2017. 14

[79] C. Mancas-Thillou and B. Gosselin. Color text extraction with selective metric-based clustering. *Computer Vision and Image Understanding*, 107(12):97 – 107, 2007. ¡ce:title¿Special issue on color image processing¡/ce:title¿. 17

[80] 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. 5, 6, 13, 17, 67

[81] J. Matas and K. Zimmermann. A new class of learnable detectors for categorisation. In *Image Analysis*, volume 3540 of *LNCS*, pages 541–550. 2005. 13, 33

[82] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. In *BMVC 2012-23rd British Machine Vision Conference*. BMVA, 2012. 30

[83] A. Mishra, K. Alahari, and C. Jawahar. Image retrieval using textual cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3040–3047, 2013. 30

[84] A. Mishra, K. Alahari, and C. V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2687 –2694, june 2012. 19

[85] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP09*, pages 331–340, 2009. 42

[86] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. on Neural Networks*, 12:181–201, 2001. 35

[87] R. Nagy, A. Dicker, and K. Meyer-Wegener. Neocr: A configurable dataset for natural image text recognition. In *International Workshop on Camera-Based Document Analysis and Recognition*, pages 150–163. Springer, 2011. 30

[88] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *ACCV 2010*, volume IV of *LNCS 6495*, pages 2067–2078, November 2010. 5, 6, 84

[89] L. Neumann and J. Matas. Estimating hidden parameters for text localization and recognition. In *Proc. of 16th CVWW*, pages 29–26, February 2011. 7

[90] L. Neumann and J. Matas. Text localization in real-world images using efficiently pruned exhaustive search. In *ICDAR 2011*, pages 687–691, 2011. 7

[91] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538 –3545, 6 2012. 5, 6, 83, 84

[92] L. Neumann and J. Matas. On combining multiple segmentations in scene text recognition. In *ICDAR 2013*, pages 523–527, California, US, August 2013. IEEE. 6, 59

[93] L. Neumann and J. Matas. Scene text localization and recognition with oriented stroke detection. In *2013 IEEE International Conference on Computer Vision (ICCV 2013)*, pages 97–104, California, US, December 2013. IEEE. 7, 84

[94] L. Neumann and J. Matas. Efficient scene text localization and recognition with local character refinement. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 746–750, California, US, Aug 2015. IEEE. 6, 84

[95] L. Neumann and J. Matas. Real-time lexicon-free scene text localization and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 38(9):1872–1885, Sept 2016. 6

[96] A. Newell and L. Griffin. Multiscale histogram of oriented gradient descriptors for robust character recognition. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1085 –1089, sept. 2011. 18

[97] W. Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, Birkeroed, Denmark, Denmark, 1985. 10, 12

[98] A. Niculescu-Mizil and R. Caruana. Obtaining calibrated probabilities from boosting. In *UAI*, page 413, 2005. 35

[99] 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. 3, 19

[100] J. Ohya, A. Shio, and S. Akamatsu. Recognizing characters in scene images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(2):214 –220, feb 1994. 12

[101] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. 17

[102] C. Pal, C. Sutton, and A. McCallum. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, page V, may 2006. 18

[103] Y.-F. Pan, X. Hou, and C.-L. Liu. A robust system to detect and localize texts in natural scene images. In *Document Analysis Systems, 2008. DAS '08. The Eighth IAPR International Workshop on*, pages 35 –42, sept. 2008. 11

[104] Y.-F. Pan, X. Hou, and C.-L. Liu. Text localization in natural scene images based on conditional random field. In *ICDAR 2009*, pages 6–10. IEEE Computer Society, 2009. 12

[105] Y.-F. Pan, X. Hou, and C.-L. Liu. A hybrid approach to detect and localize texts in natural scene images. *Image Processing, IEEE Transactions on*, 20(3):800 –813, march 2011. 12

[106] W. K. Pratt. *Digital Image Processing: PIKS Inside*. John Wiley & Sons, Inc., New York, NY, USA, 3rd edition, 2001. 34

[107] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 75

[108] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 14

[109] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 70, 71, 72, 75

[110] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 14, 70, 71, 73

[111] 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. 56, 58, 65, 67

[112] 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. 56

[113] 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. 59

[114] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003. 19

[115] M. Sarifuddin. A new perceptually uniform color space with associated color similarity measure for contentbased image and video retrieval. In *in Proc*, pages 3–7, 2005. 17

[116] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999. 10, 35, 39

[117] J. Serra. Toggle mappings. *From pixels to features*, pages 61–72, 1989. 17

[118] A. Shahab, F. Shafait, and A. Dengel. ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In *ICDAR 2011*, pages 1491–1496, 2011. 8, 10, 25, 48

[119] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 20, 74

[120] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 14, 71

[121] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–ECCV 2012*, pages 73–86. Springer, 2012. 19

[122] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 150 – 156 vol. 2, june 2005. 12

[123] H. Takahashi and M. Nakajima. Region graph based text extraction from outdoor images. In *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on*, volume 1, pages 680 – 685 vol.1, july 2005. 12

[124] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. Lim Tan. Text flow: A unified text detection system in natural scene images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4651–4659, 2015. 13

[125] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *European Conference on Computer Vision*, pages 56–72. Springer, 2016. 14

[126] A. Treuenfels. An efficient flood visit algorithm. *C/C++ Users Journal*, 12(8):39–62, 1994. 59

[127] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *European Conference on Computer Vision*, pages 255–271. Springer, 2002. 18

[128] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Computer vision and pattern recognition, 2003. Proceedings. 2003 IEEE computer society conference on*, volume 2, pages II–691. IEEE, 2003. 18

[129] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016. 4, 7, 25, 28

[130] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing System*, 14, 2001. 10

[131] T. Vojíř and J. Matas. The enhanced flock of trackers. In *Registration and Recognition in Images and Videos*, pages 113–136. Springer, 2014. 6, 55

[132] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457 –1464, nov. 2011. 21, 51

[133] K. Wang and S. Belongie. Word spotting in the wild. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *ECCV 2010*, volume 6311 of *Lecture Notes in Computer Science*, pages 591–604. Springer Berlin / Heidelberg, 2010. 21, 27

[134] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308, 2012. 3, 22, 27, 51, 52

[135] J. Weinman, E. Learned-Miller, and A. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(10):1733 –1746, oct. 2009. 18

[136] J. J. Weinman, Z. Butler, D. Knoll, and J. Feild. Toward integrated scene text reading. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(2):375–387, 2014. 48, 50

[137] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988. 20

[138] Wikipedia. Google glass. https://en.wikipedia.org/wiki/Google_Glass, 2016. [Online; accessed 31-December-2016]. 26

[139] C. Wolf and J.-M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *Int. J. Doc. Anal. Recognit.*, 8:280–296, August 2006. 8, 9, 47

[140] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015. 20

[141] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1083 –1090, june 2012. 13, 30

[142] 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. 19

[143] X.-C. Yin, W.-Y. Pei, J. Zhang, and H.-W. Hao. Multi-orientation scene text detection with adaptive clustering. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1930–1937, 2015. 13, 54

[144] 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. 13, 17, 47, 48, 54, 59, 69

[145] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao. Robust text detection in natural scene images. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):970–983, 2014. 77

[146] M. Yokobayashi and T. Wakahara. Segmentation and recognition of characters in scene images using selective binarization in color space and gat correlation. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 167 – 171 Vol. 1, aug.-1 sept. 2005. 17

[147] 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. 69

[148] J. Zhang and R. Kasturi. Character energy and link energy-based text extraction in scene images. In *ACCV 2010*, volume II of *LNCS 6495*, pages 832–844, November 2010. 13

[149] S. Zhu and R. Zanibbi. A text detection system for natural scenes with convolutional feature learning and cascaded classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 11

[150] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014. 22