University of Amsterdam

Master Thesis

Text detection and recognition in natural scenes by combining multiple color channels with selective search

Author:
Ioannis Zervos

Supervisor:
Prof. Dr. Theo Gevers
Sezer Karaoglu

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science in Artificial Intelligence

in the

University of Amsterdam

December 2014

“ All knowledge which ends in words will die as quickly as it came to life, with the exception of the written word: which is its mechanical part. ”

Leonardo da Vinci

UNIVERSITY OF AMSTERDAM

Abstract

Faculty of Science
University of Amsterdam

Master of Science in Artificial Intelligence

Text detection and recognition in natural scenes by combining multiple color channels with selective search

by Ioannis Zervos

Text in natural scenes exist in almost every phase of our daily life. From the facade of the buildings in our city to the cover of a book in our library. With automated text detection, someone can avoid the complexity of the environment and focus on the important information that this text provides. We propose an alternative to the existing text detection algorithms. Instead of the widely used text detection approach with a sliding window of multiple scales and the classification of every image patch as text or no-text, we aim to detect the character locations from the beginning and then combine them to detect text. In that way we reduce the search area to specific locations in the image. We also use different color channels, as the variety of environmental conditions in the wild may affect the color, texture and/or shape. We apply selective search to guide our detection algorithm, which is a combination of exhaustive search and segmentation. In parallel, we train a character recognition classifier in a simple way, that achieves remarkable recognition results. Finally, we propose a word segmentation method, which doesn't require any training and is able to segment the word into characters, where we can apply the character recognition classifier.

Acknowledgements

I would like to thank my supervisors Prof. Dr. Theo Gevers and Sezer Karaoglu for their aspiring guidance, constructive criticism and advice during the project work.
I would also like to thank my family for their constant support and faith.
Finally, I would like to give special thanks to all the people that I have met during my studies: the professors, the teacher assistants, my colleagues, my teammates and my friends.

Contents

List of Figures

List of Tables

Dedicated to my family

Chapter 1

Introduction

1.1 Problem Statement

Text detection in natural scenes is a challenging task and more complicated than text extraction in document text images, where there is a clear distinction between background and foreground and each character is separated from the context. In natural scenes, text can be appear in numerous states; dark text in light background and vice versa, with wide variety of fonts, even for characters of the same word, part of words can be overlapped by object of the environment and as a result the detection of these parts can be impossible. Other factors, like camera settings, may cause blurry images or perspective distortions. A major factor that makes the text detection and recognition in natural scenes difficult, are the illumination conditions. The light of the environment may create reflections on the text surfaces, object of the environment may cast shadows on the text surface, and also the intensity of the objects depends on the light source. The affect of the illumination conditions on text detection has been reported by Tremeau et al. in [1] and by Tremeau et al. in [2].
A lot of the text detection approaches make use of sliding window. They extract subimages from the original image and they evaluate them as text or no-text. They also repeat this with sliding windows of different scales. This is estimated to need 10^6 classifier evaluation per image on average[3].

1.2 Applications

An efficient text detection and text recognition approach has many applications even in our daily life. Can be a useful component of navigation devices when it successfully recognizes the text on the street signs[4, 5], can be a necessary tool for the blind or the

visually impaired people [6], when a URL is detected can be combined with a browser to navigate to this website. A more general use can be achieved with combination with information retrieval systems; the user can retrieve additional information about an entity like a physical person, a city, an event etc. For example, when a product is detected (figure 1.1, screenshot from promotion video[1]), information like price, or ingredients and expiration date for consumables[7], or specifications and size for devices, etc. can be available to the user.

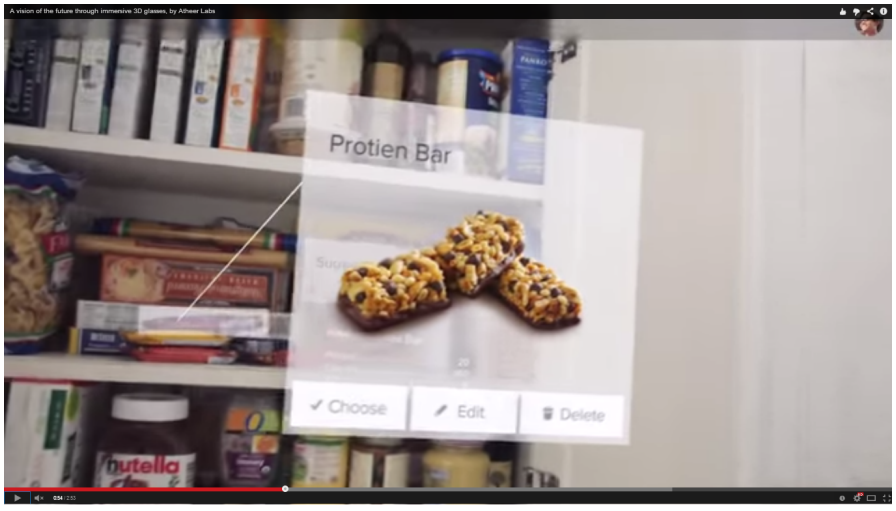

FIGURE 1.1: Application of text detection in natural scenes

1.3 Contributions

The most common approach for text detection in natural scenes is the scanning of the input image with a sliding window of different scales and the evaluation of these patches as text or no-text. This approach requires on average 10^6 classifier evaluations per image[3]. This can be avoided with guided text detection. We achieve this with the extraction of different color channels followed by two different text detection algorithms, in order to detect the character areas and then the connection of these characters into words. By doing this, we achieve not only to reduce the search space substantially, but also to take advantage of the properties of each color channel, since each one is invariant to different illumination conditions. During this process, we also evaluate the contribution of each color channel and each text detection algorithm that we use in the text detection task. In this thesis, we also present an approach for word recognition. We propose a simple word segmentation algorithm and we combine it with a pre-trained character recognition classifier. The result is refined with a spellchecker function.

[1] https://www.youtube.com/watch?v=T0onzbGNJIQ

1.4 Thesis Structure

Our thesis is organized as follows. We refer to the related work for the tasks that we try to tackle in section 2.1, and in section 2.2 we provide the necessary background for our research. In Chapter 3, we present our proposed method and the assumptions on which we based our approach. In Chapter 4, we describe the datasets and the experimental setup that we used and we provide the results of the experiments. In Chapter 5, we conclude and we suggest possible expansions of our research.

Chapter 2

Related Work and Background

2.1 Related work

In this chapter, we present and analyze the related work that has been done for each one of the tasks, that are also part of this thesis. We begin with the related work that has been done on text detection. Then, we continue with character recognition and word recognition. At the second part, we provide the necessary background of our research.

2.1.1 Text detection

Text detection is task of locating candidate word or text line areas for the given input image. There are a lot of different researches for this problem. Based on their approach, text detection algorithms can be split into different categories. As a form of object detection, text detection is often achieved with sliding window approach of multiple scales, where each patch is evaluated as text or no text [8]. In [9], Jaderberg et al. are using sliding window approach in combination with convolutional neural netowrks to distinguish between text foreground and background. This results in text line detection, on which they apply Otsu thresholding to extract the word areas. Sliding window of multiple scales is also used by Yao et al. in [10], when they try to detect character primitives, which they call strokelets. But this demands on average 10^6 classifier evaluations[3], besides the cost of training the classifier. In [11], Epshtein et al. calculate for each pixel of the input image the width of the most probable stroke that this pixel belongs to and then group them into characters. The text detection approaches in [12] and [13] are based on edge detection and texture detection respectively, but often fail because of the complex background. There also have been proposed MSER-based

text detection algorithms [14, 15]. Finally, neural networks have been used to train text detectors[16–18].

2.1.2 Character recognition

The Optical Character Recognition for scanned documents has achieved almost perfect results. On the other hand, the character recognition in natural scenes is really challenging, due to unconstraint conditions, like illuminations, variation of fonts, colors, textures etc. Other researches have tried to cope with these difficulties in numerous ways. In [19], Wang et al. are classifying the character images, by performing normalized cross correlation on the resized Histogram of Oriented Gradients of the input image. In [20], Tian et al. are using co-occurence of histogram of oriented gradients (HOG) for character recognition, which also capture the spatial relationship among the HOG fearures. In [21], Campos et al. present the results of using various features on nearest neighbor and SVM classification approaches. Neural networks have been used for character recognition as well[9, 22].

2.1.3 Word recognition

Word recognition is the task of efficiently recognizing a word, given a word image from natural scene. In [23], Shi et al. based their text recognition on a part-based tree-structures character detection. In [24] Mishra et al. are proposing one more sliding window approach. They scan the word image with a sliding window and they use a character classifier to give a character prediction per window. Then, they use a probabilistic model to give the final word proposal. Probabilistic model is also used in [25], where Novikova et al. are combining local likelihood and pairiwise positional consistency priors with higher order priors. In other words, they are relating the characters with their attributes, like color and font. Higher order language priors are also used in [26], by Mishra et al. In word recognition there is one category called holistic, where the recognition of whole word as a unit is achieved. In that category belongs the work of Almazan at al.[27], where they use the introduced Pyramidal Histograms of Characters along with Fisher vectors. In [28], Gordo is also using Fisher vectors in combination with densely extracted low-descriptors and spatial pyramids for holistic word recognition. In [29], Goel et al. are proposing one more holistic word recognition approach, where first they represent natural scene text images with gradient-based features and then they match these features with text using the introduced weighted Dynamic Time Warping (wDTW).

2.2 Background

In this section we present the background, on which we based our research.

2.2.1 Color Channels

For the purpose of our research we use the following color channels: RGB, O1, O2, O3, HSV, Lab and rgb (Figure 2.1). We concluded to these color channels, as we think that are sufficiently informative, but our research can be expanded to more color channels. We give a short description for each color channel and then we analyze their invariance to different illumination conditions.

FIGURE 2.1: All color channels
1st column: RGB,
2nd column: O1, O2, O3,
3rd column: Lab,
4th column: HSV,
5th column: rgb

2.2.1.1 RGB

RGB color model stands for **R**ed, **G**reen and **B**lue. According to this model, red, green and blue colors are combined to produce a broad array of colors and is based on human perception of colors. The range of values are [0,255].

2.2.1.2 Opponent color channels

Opponent color spaces are computed from the LMS color space (named after **L**ong, **M**edium and **S**hort wavelengths and is the response of the three types of cones of the human eye), after applying the transformation introduced by Wandell[30](equations 2.1, 2.2, 2.3). O1 is a luminance component, O2 is the green-red channel (G-R) and O3 is the blue-yellow channel (B-Y).

$$O1 = \frac{R - G}{\sqrt{2}} \tag{2.1}$$

$$O2 = \frac{R + G - 2 * B}{\sqrt{6}} \tag{2.2}$$

$$O3 = \frac{R + G + B}{\sqrt{3}} \tag{2.3}$$

2.2.1.3 Lab

L stands for **L**ightness, while **a** and **b** are color opponent dimensions. the a is the color balance between green and magenta, and the b is the color balance between blue and yellow.

2.2.1.4 HSV

The HSV color channel converts an image into **H**ue, **S**aturation and **V**alue components. Hue is the color of the image (equation 2.4), Saturation is pureness of the hue color (equation 2.5) and Value is the strength of the cue(equation 2.6).

$$
\begin{aligned}
R' &= \frac{R}{255} \\
G' &= \frac{G}{255} \\
B' &= \frac{B}{255} \\
C_{max} &= max(R', G', B') \\
C_{min} &= min(R', G', B') \\
\Delta &= C_{max} - C_{min}
\end{aligned}
$$

$$
H = \begin{cases}
60° \times (\frac{G' - B'}{\Delta} \bmod 6) & , C_{max} = R' \\
60° \times (\frac{B' - R'}{\Delta} + 2) & , C_{max} = G' \\
60° \times (\frac{R' - G'}{\Delta} + 4) & , C_{max} = B'
\end{cases} \tag{2.4}
$$

$$
S = \begin{cases}
0 & , \Delta = 0 \\
\frac{\Delta}{Cmax} & , \Delta <> 0
\end{cases} \tag{2.5}
$$

$$
V = C_{max} \tag{2.6}
$$

2.2.1.5 rgb

Similar to RGB, with the difference that normalization removes all illumination effects from the image while preserving chroma. The benefit of this channel is that removes shadows and lighting changes on color pixels. The formulas to calculate this values are using the RGB values, as you can see in equations 2.7, 2.8, 2.9.

$$
r = \frac{R}{(R + G + B)} \tag{2.7}
$$

$$
g = \frac{G}{(R + G + B)} \tag{2.8}
$$

$$
b = \frac{B}{(R + G + B)} \tag{2.9}
$$

The reason why we choose these color channels, is that each one behaves differently under different illumination conditions and has different invariances concerning light intensity, shadows and highlights, as it has been proven by van de Sande et al.[31] end by Uijlings et al.[32]. These invariances are displayed in table 2.1.

color channels	R	G	B	O1	O2	O3	L	a	b	H	S	V	r	g	b
Light intensity	-	-	-	+	+	-	-	+/-	+/-	+	+	-	+	+	+
Shadows/Shading	-	-	-	-	-	-	-	+/-	+/-	+	+	-	+	+	+
Highlights	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-

TABLE 2.1: Invariances of the color channels

As we can see in this table, the most invariant to photometric changes color channel is the Hue, as it is invariant to light intensity, shadows and highlights. The color channels r, g, b, along with S color channel are invariant to shadows and changes in light intensity, whereas a and b are partially invariant to the same conditions. Finally, the majority of the color channels that we use is affected by these factors, but their ability to detect different areas of the color space is crucial for our research. For example, in figure 2.2 we can see, that some color channels are unable to detect character areas, that are affected by the illumination conditions of the environment. Figure 2.3 illustrates another example of color channel extraction, where shadows affect the text areas.

FIGURE 2.2: Example of color channel extraction

2.2.2 Maximally Stable Extremal Region (MSER)

Maximally Stable Extremal Region (MSER) was first introduced by Matas and al. [33] as a tool for finding correspondences between two images of the same scene under different perspective, in order to be used for stereo matching. This algorithm thresholds the input image on various gray values and connects the detected regions into a component tree with spacial overlap. For each region, the change of size is calculated based on the size of

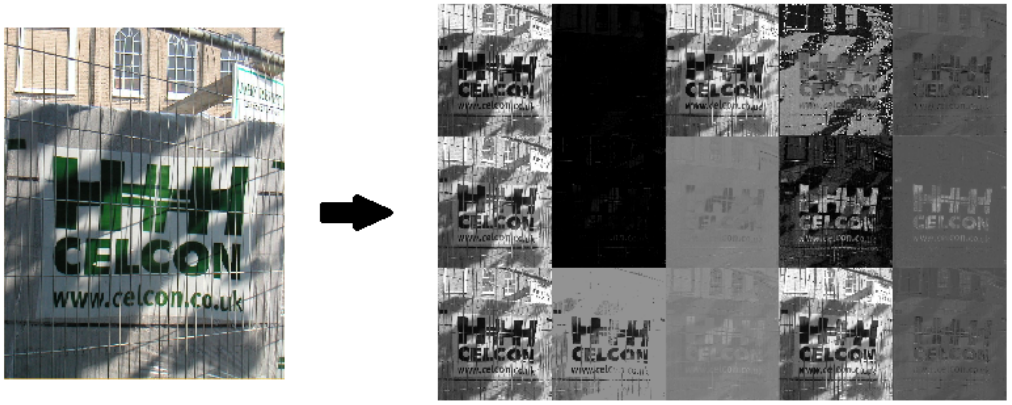

FIGURE 2.3: Example of color channel extraction

the same region in the above and below level. These regions are robust to illumination changes and also their ability of rotation invariance make them ideal for stereo matching. This algorithm is able to detect homogeneous light regions on dark background, as well as homogeneous dark regions on light background, ability necessary for our method, since the text can be found in both cases. We use the MSER implementation from vlfeat [1].

2.2.3 Saliency algorithm

We will also use an alternative for text detection, introduced by Karaoglu et al. [34] (figure 2.4) and is based on the work of Karaoglu et al. in [35]. This algorithm is based on color (figure 2.4 (b)) and curvature saliency cues(figure 2.4 (c)), since text in natural scenes has uniform color, which separates it from the background, as well as its curvature make it distinguishable from the background. The returned regions from these two approaches, are refined based on contextual priors(figure 2.4 (d)). Also noise removal is applied to get the final output (figure 2.4 (e)).

2.2.4 Connected component

According to graph theory[36], a connected component of a undirected graph is a subgraph, where any two vertices are connected to each other by paths. In image analysis in our case, a connected component in a binary image is a set of pixels that form a

[1] http://www.vlfeat.org/overview/mser.html

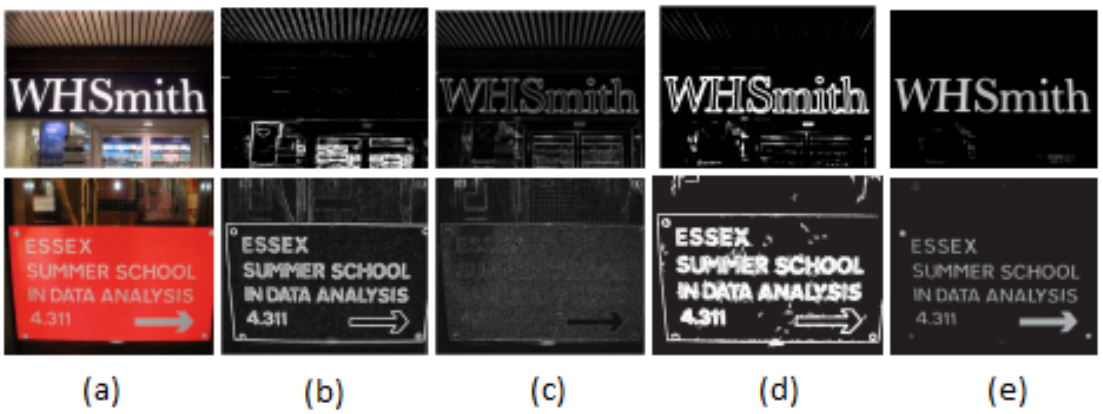

FIGURE 2.4: Saliency-based text detection algorithm
(a)original
(b)color boosting
(c)curvature shape saliency
(d)context guided saliency
(e)final output

connected group [2] (figure 2.5). Each of the connected components has some properties that can be useful. We will make use of the following properties:

- **BoundingBox:** The smallest rectangle that contains the component
- **Centroid:** The coordinates of the component's center of mass

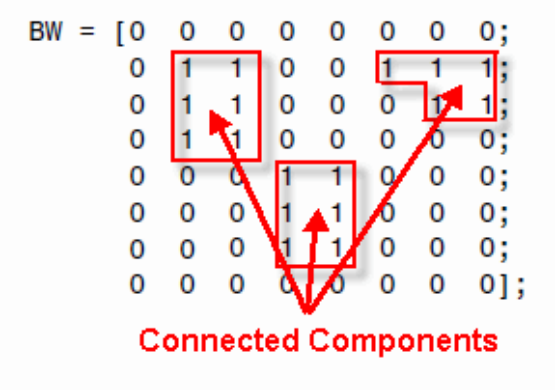

FIGURE 2.5: Connected component

2.2.5 Object bounding box proposal

Selective search[32] has been used successfully in object detection (figure 2.6) and is a combination of exhaustive search and segmentation. Like segmentation, selective search tries to split the whole input image into smaller areas, often using more than one object

[2] http://www.mathworks.nl/help/images/labeling-and-measuring-objects-in-a-binary-image.html

detection approaches, and like exhaustive search it tries to combine parts of an object to detect the whole object. We decide to use a method of object detection for text detection, assuming that words can be detected combining its parts (characters) in the same way that an object can be detected when candidate object parts are combined.

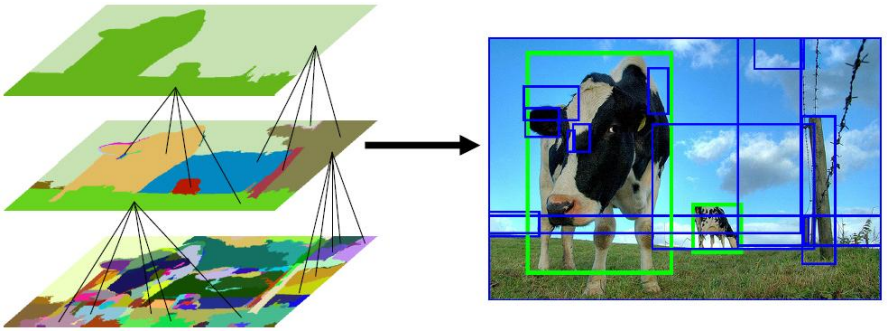

FIGURE 2.6: Selective search

2.2.6 Histogram of oriented gradients features(HoG)

Histogram of oriented Gradients are feature descriptors, widely used in computer vision. While HOG has been firstly used for object detection, and more especially for pedestrian detection[37], it performs really well in recognition tasks. This approach is similar to edge orientation histograms, scale-invariant feature transform descriptors and shape contexts. First, the gradient orientation of the input image is calculated. Then, the image is splitted uniformly into cells and for each cell, every pixel is quantized into histogram bins, where each bin represents an angle range (see figure 2.7). Then, the histograms of each cell is concatenated to create a feature vector.

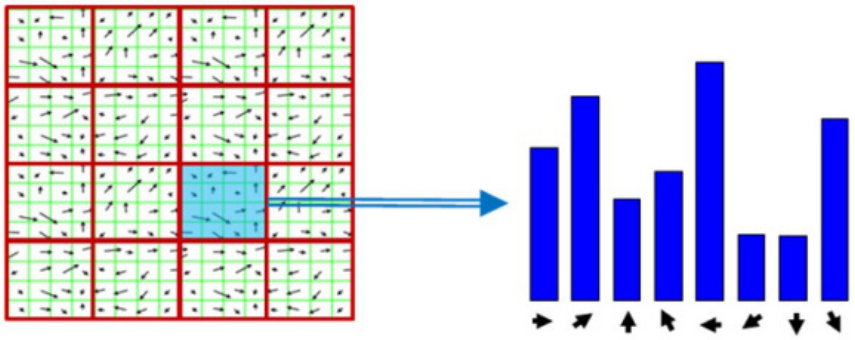

FIGURE 2.7: HoG features

We use the UOCTTI [38] variant HoG implementation from vlfeat [3]. According to this implementation, for 9 orientation bins, we have both directed and undirected orientation histograms, which are processed in four ways by averaging corresponding histogram

[3]http://www.vlfeat.org/overview/hog.html

dimensions. For each of these four ways, the l1 norm is also calculated, leading to feature vectors of size 4+3*bins = 31.

2.2.7 Support vector machine (SVM)

For character recognition we use the **S**upport **V**ector **M**achine (SVM) classifier[39]. These classifiers are supervised learning models and their most common use is for binary classification. The goal of this classifier is to construct the optimal hyperplane for separable classes (see figure 2.8). In order to achieve this, only a small part of the training set is necessary, called support vectors (the grays squares in figure 2.8). These entries define the linear decision function and maximize the margin between the classes. If the two classes are not linearly separable, the inputs can be mapped to much higher-dimensional space to facilitate the separation. Although this classifier is made for binary classification, it can be expanded to multiclass classification [40]. A strategy to do this is to build multiple one-versus-all classifiers and combine the outputs. We use the implementation of LIBSVM[4] with radial basis function kernel (RBF kernel)[41].

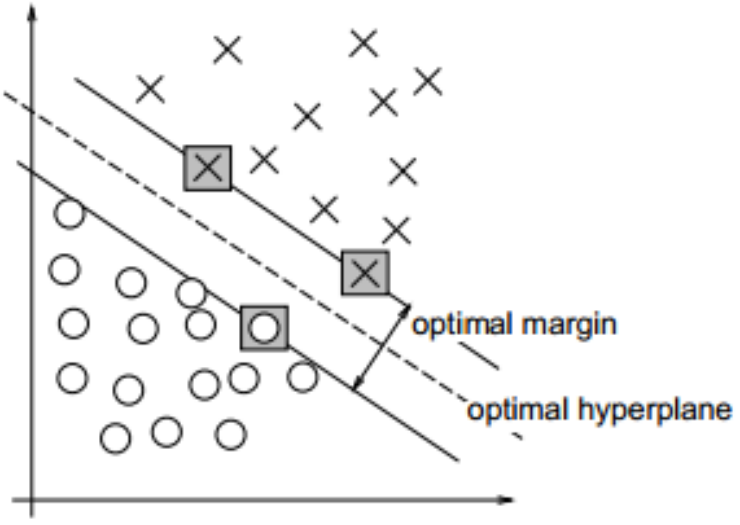

FIGURE 2.8: SVM classifier

2.2.8 Evaluation Metrics

To evaluate the results of our research, we will use the widely accepted metrics: precision, recall, f-measure.

[4]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

Precision

Indicates how many of the detected items are relevant(equation 2.10).

Recall

Indicates how many of the relevant items are detected(equation 2.11).

F measure

Correlates precision and recall. We will use the harmonic mean, where precision and recall are evenly weighted, and the β in equation is equal to 1(equation 2.12).

$$precision = \frac{relevant \cap detected}{detected} \tag{2.10}$$

$$recall = \frac{relevant \cap detected}{relevant} \tag{2.11}$$

$$F_\beta = (1 + \beta^2)\frac{precision * recall}{\beta^2 * precision + recall} \tag{2.12}$$

Chapter 3

Proposed Method

3.1 Introduction

Our method is a bottom-up approach. So, the first step is to detect possible character areas and then combine these areas, based on specific criteria, to detect words.

3.2 Character Detection

For the character detection, we extract from the input image a variety of color channels and we process them with MSER and Saliency text detection algorithm. Different color channels assist in detection of characters under different illumination conditions.We follow the pipeline in Figure 3.1. As it is obvious, although our method uses a variety of color channels, the processings are indipendent, so all the calculations can occur in parallel.

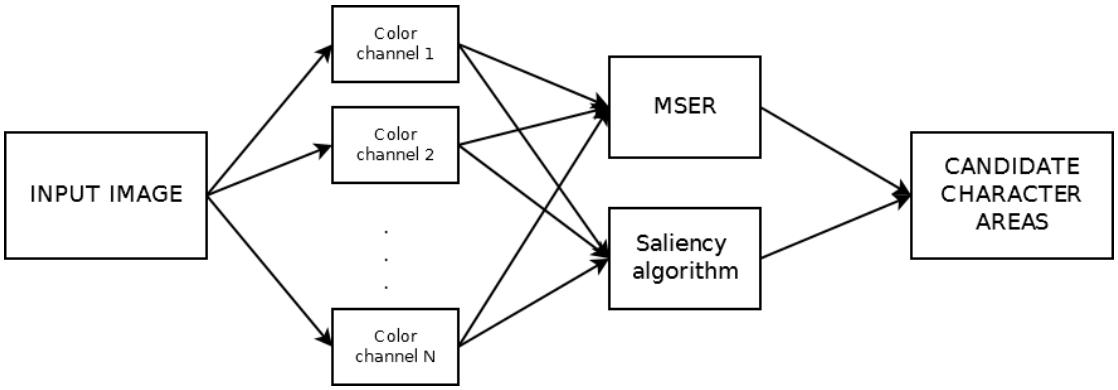

FIGURE 3.1: Character detection pipeline.

The connected components we get from the binarized output of the MSER-based and saliency-based text detection algorithms, are the candidate character areas.

Due to the illumination conditions of each image, characters that aren't detected by one color channel, can be detected by another channel. For example, after processing the G channel with MSER algorithm of a sample image, we miss the characters of the word 'FIRE', because of the illumination conditions (Figure 3.2). On the other hand, using the S channel with MSER algorithm we are able to detect those characters (Figure 3.3), since the S color channel is invariant to these illumination conditions (table 2.1).

FIGURE 3.2: Character detection example using G channel. (left) original image with the detected candidate character areas: red for false positive, green for true positive and blue for the groundtruth, (middle) grayscale image of the color channel (color channel G in this occasion), (right) the color channel processed with MSER

FIGURE 3.3: Character detection example using S channel

3.3 Word formation

For the word formation step, we have as input the candidate character areas from the previous step, but we also apply some preprocessing in order to reduce these areas without decreasing the character detection accuracy, as we explain in section 3.3.1. The combination criteria are based on the assumption that characters in a word obey some general rules, like similar character dimensions, similar spacing between characters in a

word etc. Then we use an exhaustive way to extract the candidate word areas. Finally, a filtering step is applied to reduce the false positive returned word areas.

3.3.1 Preprocessing

Since the previous step is the processing of different color channels of the same image, multiple candidate areas are returned for exactly the same area in the image, as we expected. Moreover, although we are able to detect the majority of the actual characters, a significant part of the returned areas are noise. In order to reduce the number of those areas without decreasing the accuracy of the character detection, we apply the following filtering.

At the beginning, we apply filtering based on prior knowledge, like contrast removal and aspect ratio removal. During the character detection step, we notice that parts of characters are detected as characters. The main reason of this is mostly illumination conditions of the environment, like highlights on shiny flat surfaces. For example, the processing of the H color channel, followed by the saliency algorithm of an image with a single character on a shiny surface may produce more than a thousand candidate character areas, like in figure 3.4, where we have 1146 candidate character areas. In order to reduce this number without losing the detected character, we remove candidate character areas that don't have any change in the overall contrast, when we expand their area.

In addition, we apply duplicate area removal. We define as duplicate areas, candidate character areas with overlap larger than 95%. At the end, we remove the abnormally elongated candidate character areas.

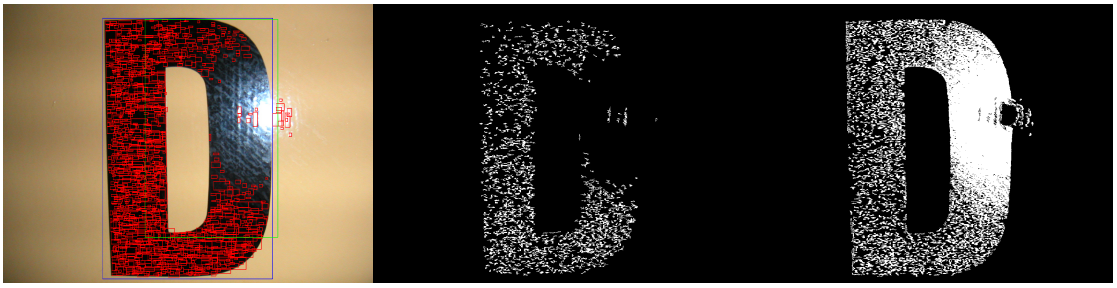

FIGURE 3.4: Character detection example. Processing of the H channel followed by the saliency algorithm produces 1146 candidate character areas.

3.3.2 Word Proposals

The next goal is to combine the character areas to form words, so we have to model the rules that exist. It is common that the characters have similar height with the

characters in its neighborhood. In order to define this neighborhood, we introduce the term 'test_radius', which is the factor with which we have to multiply the maximum of the dimensions of one character area until we reach the next character in a word. We prefer this dynamic measure to define the surrounding area, because the spacing between characters depends on the font size of the characters, so it cannot be an absolute number of pixels. In a similar way, we introduce the term 'test_slope', which is the slope between the centroids of two consecutive characters in a word. These two factors, test_slope and test_radius, define the search area for each one the candidate character areas(figure 3.5 (left)). When a candidate character area fulfill the aforementioned restrictions, it is safe to say that there is a connection between these two character areas(figure 3.5 (right)).

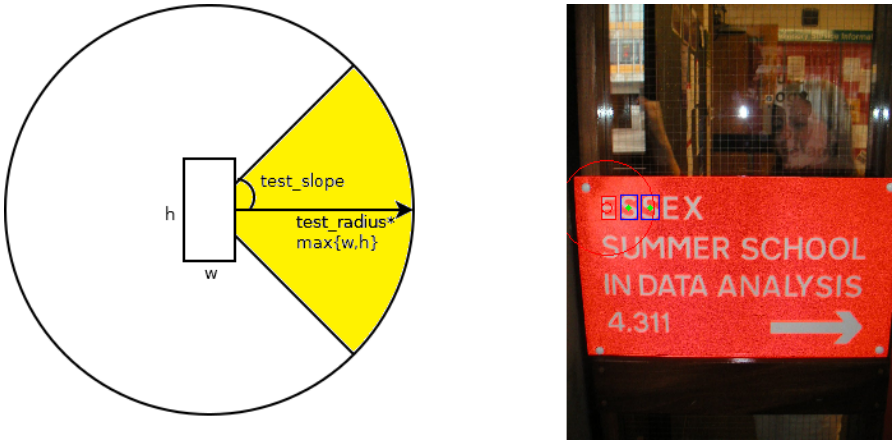

FIGURE 3.5: Search area(left) and connection step (right)

FIGURE 3.6: Connection graph

3.3.3 Extract paths from the connection graph

When we represent the candidate character areas as nodes and the connections between areas as edges, we have a Directed Acyclic Graph(DAG), which we name connection graph (figure 3.6). It is usual during the connection step to connect the end of one word with the beginning of the next one in the text line. For example, the big clusters of figure 3.7 are actually the second and third text lines of the input image. In order to

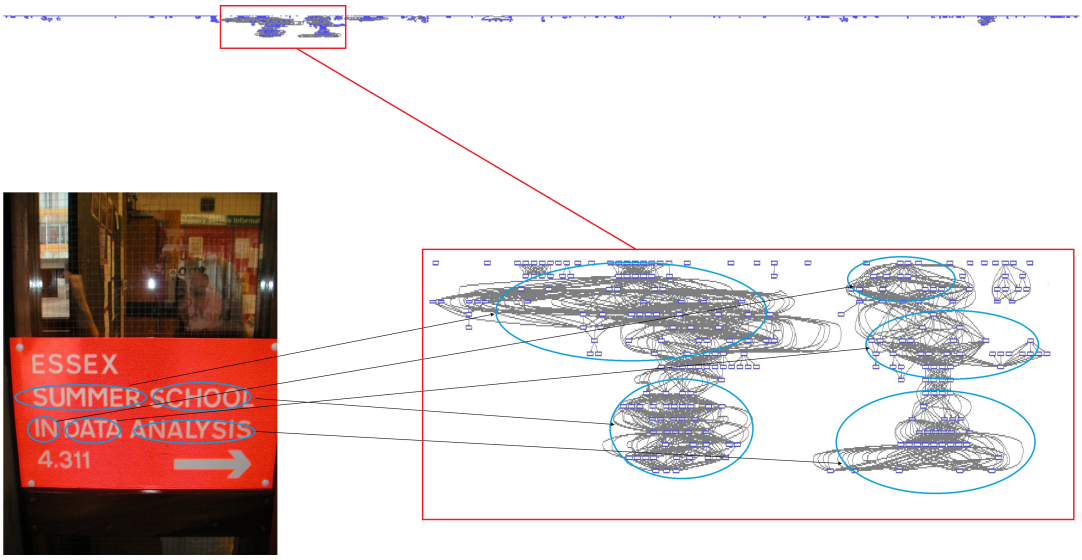

FIGURE 3.7: Connection graph analysis

retrieve the most probable word area, we have to extract all the possible paths from the connection graph. For example, for the below connections we get the connection graph of figure 3.8.

$$
\begin{aligned}
&1 \rightarrow [2] \\
&2 \rightarrow [4 \quad 5] \\
&\ldots \\
&4 \rightarrow [49] \\
&5 \rightarrow [6] \\
&6 \rightarrow [19 \quad 24] \\
&\ldots \\
&19 \rightarrow [] \\
&\ldots \\
&24 \rightarrow [] \\
&\ldots \\
&49 \rightarrow [] \\
&\ldots
\end{aligned}
$$

To extract all the possible paths from a root node of the connection graph, we perform **D**epth **F**irst **S**earch (DFS) until we reach a leaf node. In that case, the sequence of the visited nodes is returned as a candidate word path and the last connection breaks. We repeat this, until only the root node is left in the connection graph. When that happens, we continue by treating the first child as root and extracting all the paths

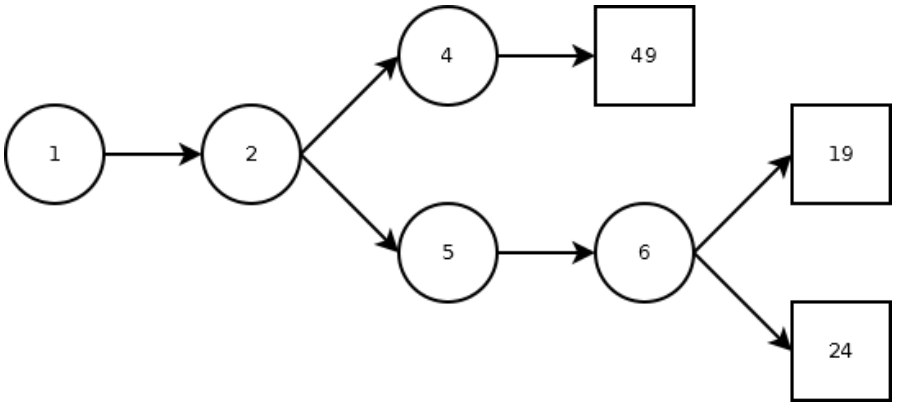

FIGURE 3.8: Example of connection graph

from its subgraph. For example, for the connection graph of figure 3.8, the returned paths are in order:

- [1 2 5 6 24]
- [1 2 5 6 19]
- [1 2 5 6]
- [1 2 5]
- [1 2 4 49]
- [1 2 4]
- [1 2]
- [1]
- [2 5 6 24]
- [...]
- [49]

This leads to a large number of returned candidate word areas, but we will try to eliminate the majority of those in the next step. This path extraction approach is aiming to extract as many paths as possible, concerning the beginning and the ending of the paths, and not concerning the sequence of the nodes. For example, if in the previous example there was a connection between node 4 and 6, with the proposed approach we are not able to extract the paths [1 2 4 6 19] and [1 2 4 6 24].

3.3.4 Postprocessing word proposals

The last step of text detection is the word area postprocessing, with the goal to remove the false positive word areas from the previous step. Again, we apply some prior knowledge filtering. In general, the words have horizontal orientation, so candidate word areas with diagonal orientation are removed. In addition, we have a maximum word path length threshold, since the maximum word in ICDAR 2003 train dataset has 10 characters, so any path with more than 10 characters is assumed as noise. Because we connect the character areas in an exhaustive way, there are a lot of paths that occupy almost the same area, so we just have to keep one of them and discard the rest. Our text detection pipeline is similar to Chen et al.[14], with the addition that we apply more text detection algorithms and on different color channels.

3.4 Character recognition

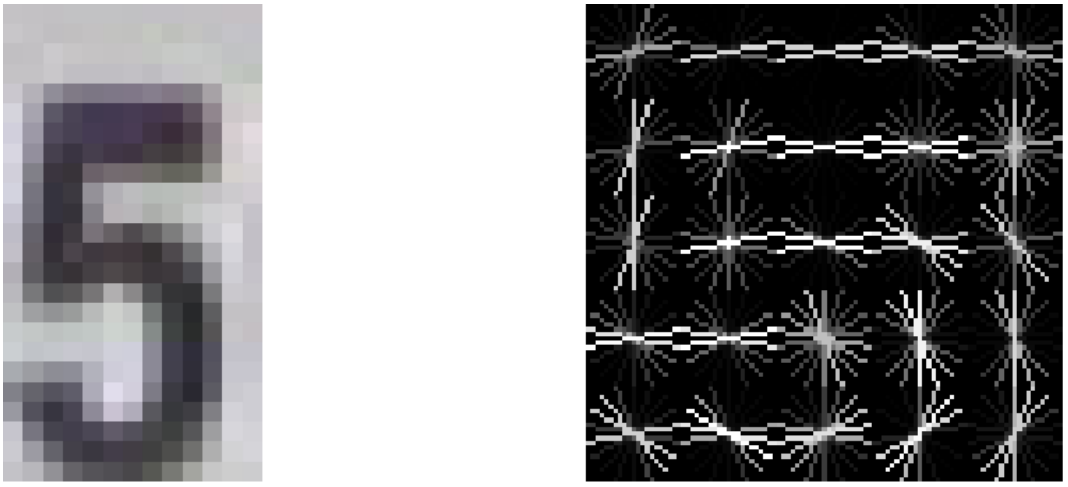

FIGURE 3.9: Sample character image (left) and the HOG features extracted with a 5x5 grid (right)

Character recognition is the classification of a character image into one of the 62 possible classes: 26 classes for the upper case characters, 26 for the lower case characters and 10 classes for the digits. For the character recognition we follow a simple but effective approach. First, we apply a dense grid without overlap on the input image (figure 3.9 left), which splits the image into grid_rows*grid_columns cells. Then, from each cell of the grid, we extract a HOG feature (figure 3.9 right). Finally, we concatenate these features into one final descriptor, with which we train a multiclass SVM classifier.

Merged classes	
C	c
I	i
J	j
K	k
O	o
P	p
S	s
U	u
V	v
W	w
X	x
Y	y
Z	z

TABLE 3.1: Merged character classes

3.4.1 Class merging

The Latin alphabet has the specialty that some characters have similar morphology for their upper and lower case. It is safe to merge the characters of table 3.1, as the recognition either as upper case character or lower case character doesn't affect the overall spelling of the word that this character appears in. This class merging has been used previously in [42].
On the contrary, the morphology of 0 (zero) is similar to O (upper case) and o (lower case), as well as the morphology of l (lower case) to I (upper case) and i (lower case), but the false recognition of a character of these classes would change the spelling of the word, so we avoid the merging of these classes.

3.4.2 Grid size

The next step of character recognition is to define the most appropriate size for the aforementioned grid. Intuitively, we examine grids with equal number of rows and columns. In addition, we try grids where we have more rows than columns and the opposite. Our goal is to find the most suitable combination, which also prevents overfitting.

3.5 Word Recognition

The last part of our project is the text recognition. In other words, the task of recognizing the word, given i)the word area from the word proposal and ii)using the character recognition that we described in the previous section. For this, we borrow an idea from

the offline handwritting recognition, the dissection[43], which is the decomposition of the word image into subimages/character units, where in turn we can apply character recognition.

3.5.1 Word Segmentation

An efficient way to extract character segments of the word is to extract the vertical profile. This approach has been used by Lavrenko et al.[44] in extracting the character areas from a continuous trace during text recognition in historical documents. To get this, we sum vertically the values of the grayscale input word image. The local ultima of the smoothed vertical profile is a good indication for character segments: Local minima are good indications for segments of light text on dark background (figure 3.10), whereas local maxima for segments of dark text on light background (figure 3.11).

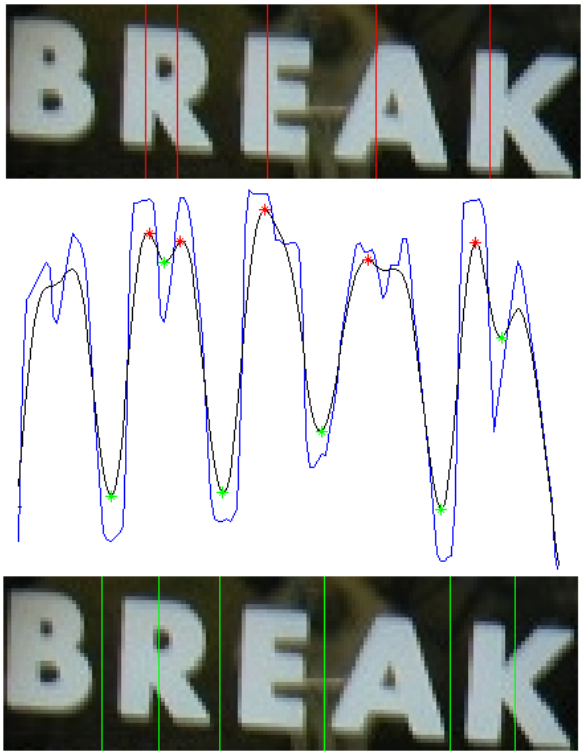

FIGURE 3.10: Character segmentations for light text on dark background
(top) Character segmentations based on local maxima
(middle) Vertical profile of the word, local maxima with red, local minima with green
(bottom) Character segmentations based on local minima

It is interesting to note that in both figures 3.10 and 3.11, the wrong segmentations are on the higher local maximum or on the lower local minimum. We will use this information in an iterative procedure, to remove the false segments.

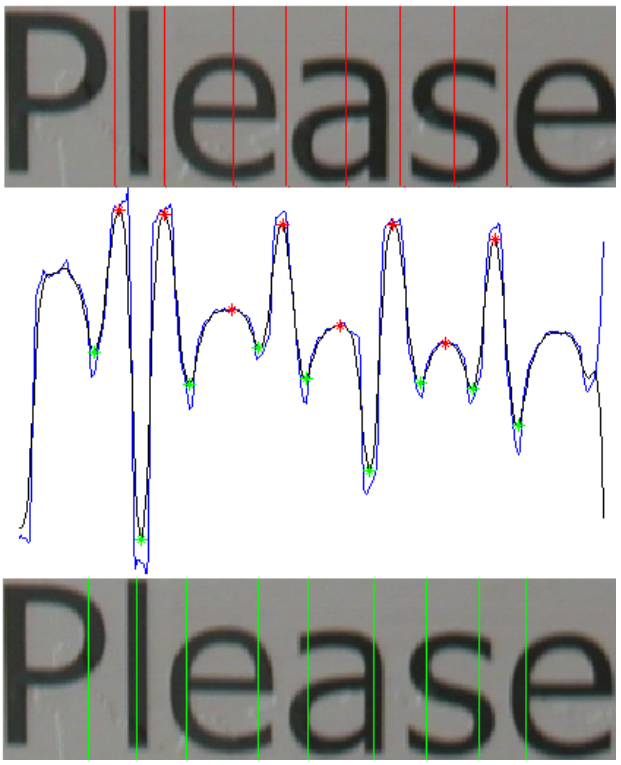

FIGURE 3.11: Character segmentations for dark text on light background
(top) Character segmentations based on local maxima
(middle) Vertical profile of the word, local maxima with red, local minima with green
(bottom) Character segmentations based on local minima

3.5.2 Word Recognition

To complete the word recognition, we use an iterative, brute-force approach. We assume that each segment is actually a character and we apply the character recognition. For the next iteration we remove the segment that is most probable to be false, meaning the lower local maximum, or the higher local minimum and we repeat the recognition until we end up with only 2 segments.
For example, for the segments of figure 3.10, we have the following recognitions:

- $['BltEAIC']$
- $['BREAIC']$
- $['BREAK']$
- $['BRMK']$
- $['BMK']$
- $['MK']$

As a final step, we use a spell checker function trained on a dictionary, consisting of the words which appear in the dataset. Lucene spellchecker function[45] uses Levenshtein distance[46] to measure the string similarity, which calculates the minimum number of single-character modifications that have to be applied to one string to transform it to another. This function returns a ranked list of suggestions for the given word area, even if the character-by-character recognition fails in more than one cases, and also doesn't return anything if the previous recognition is just a sequence of random characters. For example, it is able to correctly recognize 'EJNGJNEERJNG' as 'ENGINEERING', although it is the output of both wrong segmentations and misrecognized characters (figure 3.12). On the other hand, it is able to discard the prediction 'EJNXSJJNELIJXJJNXS' of the same oversegmented word as unknown word. The latter ability can be useful to reduce the number of the false positives word areas from the text detection step.

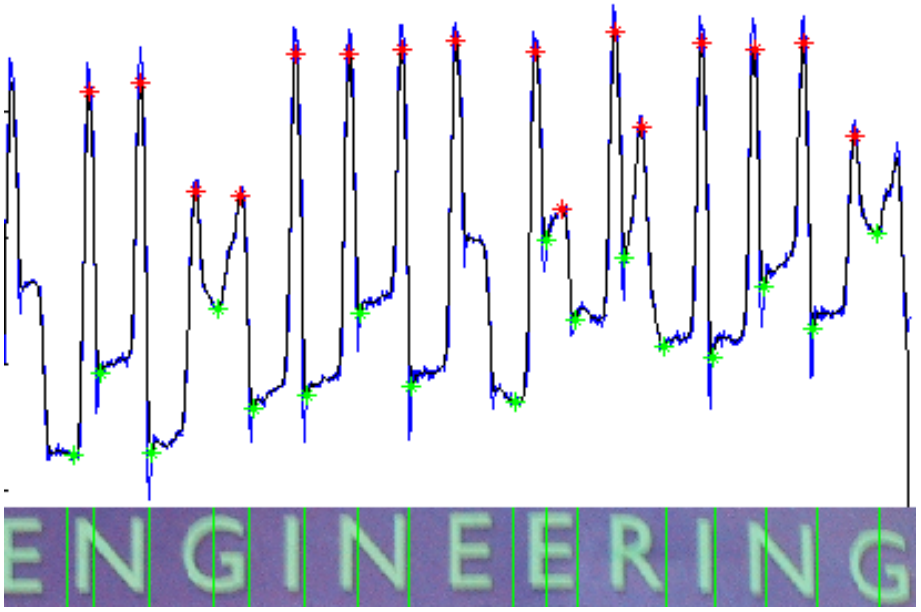

FIGURE 3.12: Vertical profile and segmentations of the word 'ENGINEERING'

3.6 Summary

To summarize, the whole thesis can be split into text detection and text recognition. First, we perform text detection on the input images of natural scene, which consists of the following steps: i) character detection, by extracting color channels and process them with two text detection algorithms, ii) preprocessing, by discarding non-character areas, iii) Word formation, by exhaustively combining the character areas into words and iv)word detection postprocessing, by removing non-word areas (figure 3.13). Then, we apply text recognition on the detected word areas, which consists of the following steps:

i) word segmentation, a brute-force approach where we try to extract the character segments, ii) word recognition, where we combine the character recognition classifier, that we have trained offline, with the candidate character segments from the previous step, and finish with iii)spell checker step, where the recognized words are refined based on a dictionary or discarded as noise(figure 3.13).

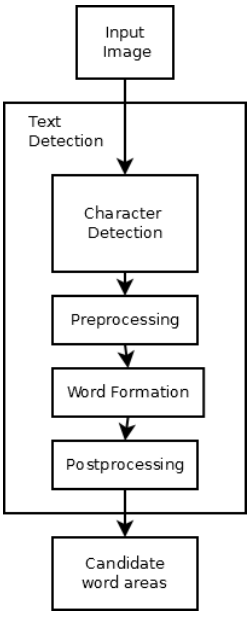

FIGURE 3.13: The proposed text detection pipeline

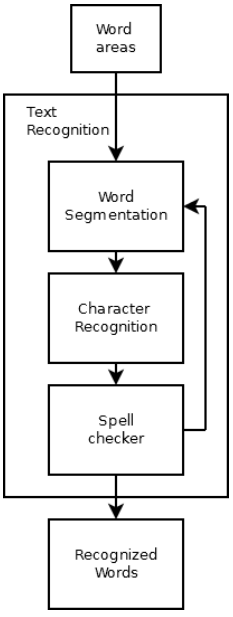

FIGURE 3.14: The proposed text recognition pipeline

Chapter 4

Experiments

In this chapter we present the experimental setup and the results for each of the task that our project tries to tackle. In section 4.1, we describe the dataset, the experimental setup, and the performance of the character detection, followed by the performance of word detection algorithm. In section 4.2, we present the character recognition approach, starting with the description of the datasets, and followed by the recognition experiments concerning the class merging and the grid size applied on character images, which lead to remarkable character recognition results. Finally, in section 4.3, we apply the character segmentation and character recognition on a widely used dataset and present our results.

4.1 Text Detection

Because our approach is a bottom up approach, we first evaluate the detection of the parts (characters) and then the detection of the whole (word).

4.1.1 Datasets

For the purposes of our research, we use the ICDAR 2003 dataset[1] for text locating. This dataset contains 249 images with natural scenes, with a total of 1107 words and 5370 alphanumeric characters. As the second part of this section is focused on combination of candidate characters in order to form words, we will experiment on a subset of ICDAR 2003 dataset, which contains images of natural scenes with multi-character words. This subset consists of 236 images with a total of 1006 multi-character words. Finally, in order to compute the parameters test_radius and test_slope that we introduced in section 3.3

[1] http://algoval.essex.ac.uk/icdar/Datasets.html

we will use the ICDAR 2013 training dataset, which is similar to ICDAR 2003, but its groundtruth is more appropriate for our experiments, as there is a clear segmentation for each character on the original image.

4.1.2 Character detection

To detect characters in natural images, we follow the pipeline in figure 3.1. According to this pipeline, the color channels are extracted and processed by MSER-based and saliency-based text detection algorithm. The results of each of these approaches are displayed in table 4.1. In order for a character to be counted as 'detected', we define an overlap threshold between the detected area and the character area of the groundtruth. But, because the character area of the groundtruth is based on the word height, and no on the character height, we choose a relaxed overlap threshold of 0.33.

	Saliency		MSER	
color channel	accuracy %	# of boxes	accuracy %	# of boxes
R	66.83	12770	47.52	19955
G	74.52	13772	53.48	20702
B	71.09	12482	50.81	21562
O1	13.87	1483	9.23	3283
O2	11.50	959	6.87	3791
O3	38.52	7882	24.82	10525
L	75.43	13408	53.79	20669
a	14.37	435	3.18	2866
b	14.60	352	2.51	4154
H	30.16	85333	17.07	35374
S	50.07	27219	28.34	29790
V	69.03	12609	46.83	21231
r	22.53	3079	8.84	8775
g	12.98	1998	3.89	3672
b	18.34	2538	5.00	7362
all channels	88.36	196319	80.16	213711
	total accuracy: 92.27		total number of boxes: 410030	

TABLE 4.1: Character detection results

As we can see in this table, if we had followed each color channel individually, we would have a maximum detection of 75.43% of the characters, as a result of the L color channel followed by the saliency algorithm. In similar way, if we had chosen to process all the color channels either by saliency or MSER algorithm, we would have a detection accuracy of 88.36% and 80.16% respectively. In addition, the overall detection result of processing all the color channels, followed by both MSER and saliency algorithms, is 92.27% and

produces a total of 410030 candidate character areas.

On the other hand, if we dive in the results, we can see that the most noisy color channel is the H, as it produces a total of 120707 candidate character areas and detects 30.16% or 17.07%, when it is followed by the saliency or MSER algorithm respectively. The overall contribution of this color channel is summed up to 43 out of 5370 character areas that aren't detected by any other approach, as we can see in table 4.2. In the same table, we can also see that the contribution of color channels 'O2' and 'g' is non-existent, while the contribution of seven more color channels is single digit number. That may happen because of the plethora of color channels that we used, in such extend, that the majority of the characters are detected by more than one color channels.

color channel	# characters exclusively detected per color channel
R	21
G	8
B	27
O1	3
O2	0
O3	14
L	12
a	2
b	2
H	43
S	40
V	9
r	5
g	0
b	1

TABLE 4.2: Color channel contribution table

FIGURE 4.1: Hard cases in character detection

When we try analyze when the character detection fails, we can see that sometimes, the text in natural scenes is hard to detect even for the human eye, much more to recognize it (figure 4.1). This is the result not only of the illumination conditions, but also of the

color similarity between the text color and the background, or because of the confusing word texture etc.

4.1.3 Character area preprocessing

As we described in section 3.3.1, it is necessary to reduce the number of candidate character areas without decreasing the character detection accuracy, before we proceed to the next step. For this purpose, we apply a prior knowledge filtering, like removing homogeneous character areas, elongated areas etc., and we perform duplicate removal, because a lot of character areas are detected by more than one color channel. In table 4.3 we present the affect that this filtering has on the character detection results.

	Before preprocessing		After preprocessing	
Algorithm	accuracy %	# of boxes	accuracy %	# of boxes
Saliency	88.36	196319	86.67	122209
MSER	80.16	213711	78.35	78111
Both	92.27	410030	91.04	184417

TABLE 4.3: Character detection results before and after preprocessing

In this table we can see that this filtering of the candidate character areas, discards almost one third of the candidate character areas detected by saliency algorithm and more than 60% of the areas detected by MSER algorithm, with a cost of 1.69% and 1.81% respectively. When we apply this filtering on the total character areas, we discard more than half of the detected character areas with a minimal loss of 1.23% on the character detection accuracy.

We can also see that the saliency-based algorithm constantly outperforms the MSER-based algorithm, with a difference of 8%. A possible explanation for this, can be the fact that the saliency algorithm takes under consideration both the color and the curvature of the characters. On the contrary, MSER algorithm is based on the intensity of the character areas.

4.1.4 Word formation

For the word formation step, we have to find the most appropriate values for test_radius and test_slope variables that we introduced in section 3.3.2. For this purpose we use the groundtruth of the ICDAR 2013 text segmentation dataset (figure 4.3). In order to have more accurate results, we train our algorithm only on a subset of this dataset, where the characters are clearly separated. In this subset, we have 2616 multi-character

words with 9971 connections.

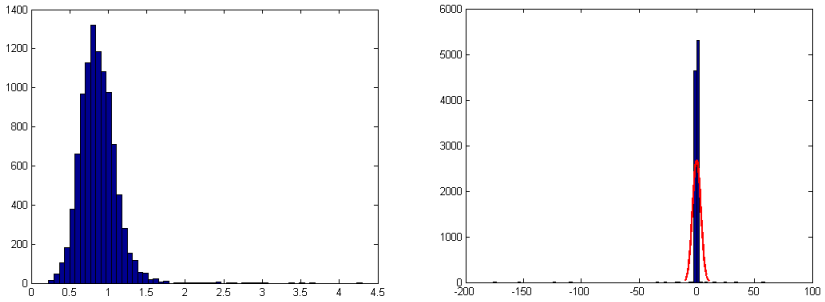

FIGURE 4.2: Values for test_radius (left) and test_slope (right) in ICDAR 2013 trainset

For the factor test_radius, which is the factor with which we have to multiply the maximum of one character dimensions in order to reach the next character of the word, we have the results of figure 4.2 (left). The maximum value is 4.29, the minimum 0.22 and the average 0.86. For the purposes of our research we will use the value 1.7 for test_radius, which includes the 99.41% of our training samples, in order to compensate for a possible character misdetection in the middle of a word.
For the factor test_slope, which is the slope between the centroids of two consecutive characters in a word, we have the results of figure 4.2 (right). As we can see, the majority of the words have horizontal orientation, as the slope between the consecutive characters is close to zero. The average value of the absolute slopes is 0.21, so for the purposes of our research, we will use the value 0.3 which includes 92.84% of our training samples. We prefer this value, because it is close to the average value.

FIGURE 4.3: Sample image from ICDAR 2013 dataset (left) and the groundtruth (right)

4.1.5 Word detection

After forming the connections between the candidate character areas, we are able to construct the connection graph and extract all the possible paths from it, as we mentioned in section 3.3.3. Then, we also apply the postprocessing word proposal step, where we

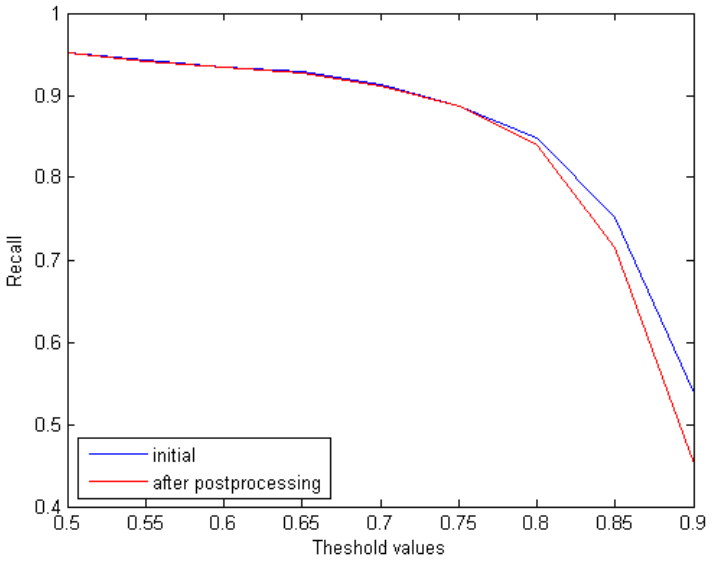

FIGURE 4.4: Text detection Recall for different values of overlap threshold

discard the false positive word areas. While initially we have a total of 5969469 candidate word areas for the whole dataset, after postprocessing the one tenth has remained, 572726 candidate word areas in particular. This is translated to less than 2500 candidate word areas per image, which is one fourth of what Hosang et al. are expecting in [3].

FIGURE 4.5: Text detection example

In order for a word to be counted as detected, the overlap between the candidate word area and the groundtruth must be above a threshold. In figure 4.4, we display the word detection recall for different values of the overlap threshold before and after the postprocessing step. We can see that this step is really efficient as we end up with one tenth on the initial words, and the cost that we pay is minimal and visible only for strict values of threshold. In ICDAR competitions there is a threshold of 0.8, which in our case gives us a recall of 0.84, while it increases to 0.91, if we use a more relaxed overlap threshold of 0.7. In Figure 4.5, we diplay an example of text detection. We can see that

the bounding box proposals are focused around the actual text areas. The fact that we have so many word proposals can be explained by the exhaustive way we used to extract tha paths from the connection graph. that

Method	Recall
SWT[11]	0.60
Alex Chen[47]	0.60
snoopertext[48]	0.61
hinnerk becker[47]	0.67
CRF[49]	0.71
proposed	**84.0**

TABLE 4.4: Text detection results on ICDAR 2003 text locating dataset

In table 4.4, we compare our text detection results with similar works. As we can see, our method outperforms other methods, but we have to mention that the other researchers also try to have high precision, while we are covered by the research of Hoslang et al. in [3], which says that so many bounding box proposals are acceptable for detection approaches that doesn't use sliding window.

4.2 Character Recognition

As we mentioned before, character recognition is the classification of a cropped character image into one of the possible character classes. These character images are extracted from natural scene images, where there are complex backgrounds and a variety of textures, fonts and lighting conditions.

4.2.1 Datasets

FIGURE 4.6: Sample image from character recognition datasets: ICDAR 2003 robust OCR dataset (left), chars74k dataset (right)

For the character recognition task, we use the widely used and commonly accepted datasets:

Chars74k[21]

This dataset includes characters from natural scenes, hand drawn characters and synthesized characters from computer fonts. We use the subset with the segmented characters from natural scenes[2], which contains 7700 images with alphanumeric characters (see figure 4.6 (right)).

ICDAR 2003 robust OCR trainset

This dataset is part of the ICDAR 2003 [3], and especially of the robust OCR task. It contains 5430 single characters extracted from natural scenes, of which 5379 are alphanumeric (see figure 4.6 (left)).

ICDAR 2003 robust OCR testset

This dataset is also part of the ICDAR 2003 and contains 6185 characters from natural scenes, of which 6113 are alphanumeric. These characters are extracted from the images of the ICDAR 2003 text location dataset.

4.2.2 Experimental setup

In this section we describe the experimental setup for the character recognition task. First, we examine how the class merging affects the character recognition accuracy. Then, we experiment on different grid sizes to find the most appropriate for the character recognition task.

4.2.2.1 Merging classes

To evaluate the effect of class merging, we repeat the experiment two times, one time with all 62 classes (0-9,A-Z,a-z) and one time with the reduced 49 classes. We test with the same grid of 5x5. The results are displayed in table 4.5. At first, we train our algorithm on either the chars74k dataset or ICDAR trainset and test the performance on the two remaining datasets. Finally, we train a character recognition classifier on both chars74k dataset and ICDAR trainset, with a total of 13813 train samples, and test on the ICDAR testset. The latter classifier is referred as 'combined'. We can see that with the merged classes we have a constant improvement on different combinations of training and testing sets, with a range from 2.89% to 9.73%.

In figure 4.7(left) and 4.8(left) we have the confusion matrices, where the rows are the predicted classes and the columns are the groundtruth classes. We can see that the majority of the test characters are correctly classified, since they belong to the diagonal.

[2] http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k
[3] http://algoval.essex.ac.uk/icdar/

		62 classes		49 classes	
trained on	tested on	accuracy	results	accuracy	results
chars74k	icdar train	66.76%	4073/6113	74.12%	4668/6113
chars74k	icdar test	66.63%	3591/5379	76.36%	3987/5379
icdar train	chars74k	76.31%	5381/7700	79.20%	5962/7700
icdar train	icdar test	69.88%	4104/5379	77.43%	4260/5379
combined	icdar test	79.55%	4279/5379	82.60%	4443/5379

TABLE 4.5: Character recognition results depending on the number of classes

On the other hand, in figures 4.7(right) and 4.8(right) we have the binarized confusion matrices, that display which classes are confused with which classes. In the first figure, we notice the lines that are parallel to the diagonal. These misrecognitions are actually the lower/upper case characters of table 3.1, which are corrected in the second confusion matrix.

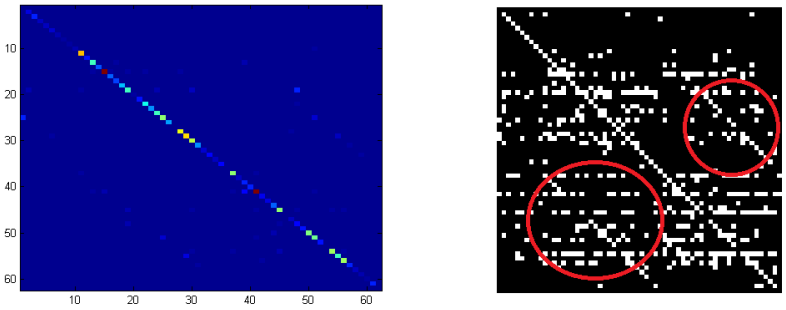

FIGURE 4.7: Confusion matrix with 62 classes(left), binarized confusion matrix with 62 classes (right)

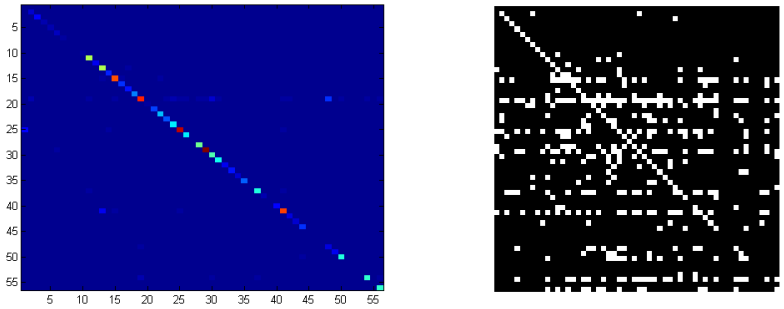

FIGURE 4.8: Confusion matrix with 49 classes(left), binarized confusion matrix with 49 classes (right)

In figure 4.9 we display the distribution of the misclassified characters. In figure 4.10 we analyze the most common of them. We can see that the most usual confusion, as expected, is the classification of 'l' (class number 48) as 'I' (class number 19). The second most common misclassification is class 'e' (class number 13) as class 'C' (class

number 41), followed by the misrecognition of '0' (class number 1) as 'O' (class number 25). Finally, the character 'I' (class number 19) is falsely recognized either as 'r' (class number 54) or 'l' (class number 48), character 'O' (class number 25) is confused with character 'e' (class number 41) and character 'T' (class number 30) is confused with character 'I' (class number 19).

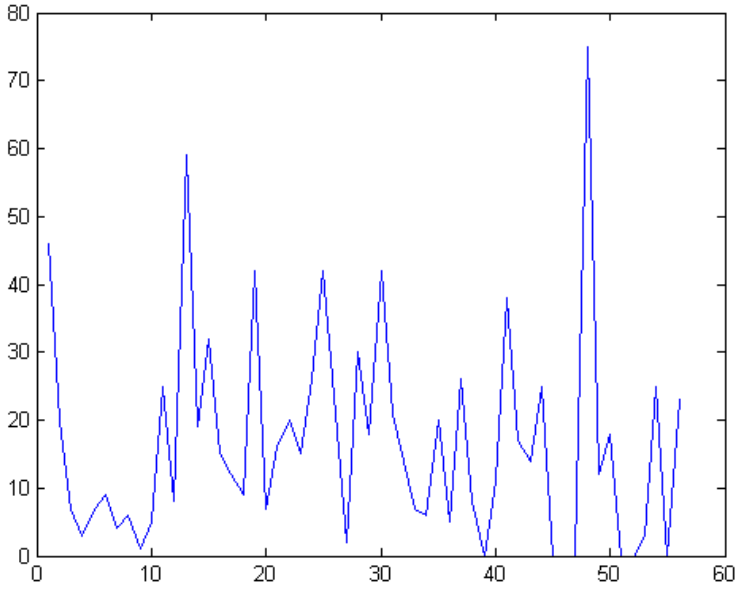

FIGURE 4.9: Distribution of misclassified characters

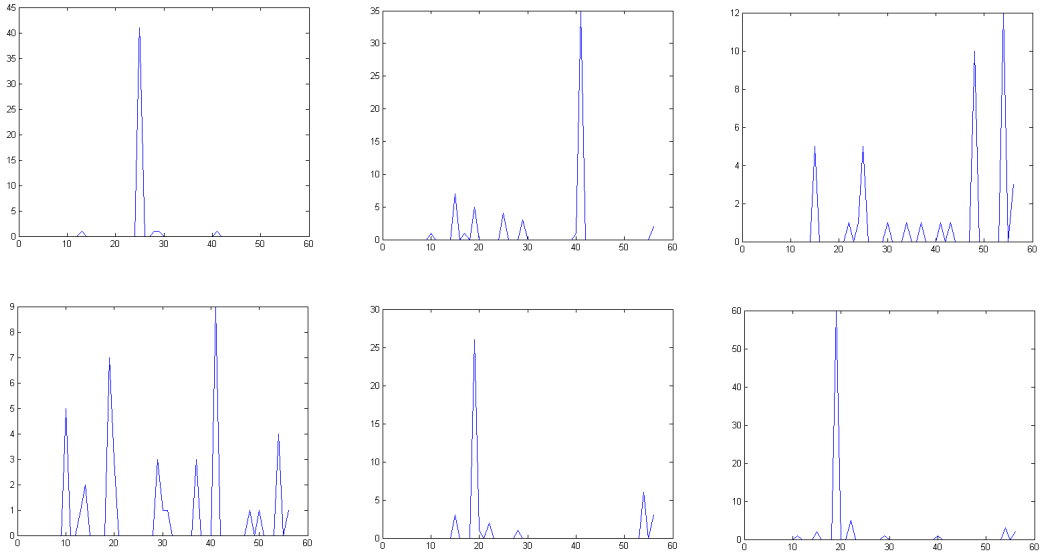

FIGURE 4.10: Misclassifications of character recognition
(top) class '0' (class number 1), class 'C' (class number 13), class 'I' (class number 19)
(bottom) class 'O' (class number 25), class 'T' (class number 30), class 'l' (class number 48)

4.2.2.2 Grid Sizes

The next step in our research is to examine different grid sizes of character images, as we described in section 3.4, and how they affect the character recognition results. As we mentioned, intuitively we start with grids that have same number of rows and columns. Then, we also experiment with grids where we have more rows than columns and vice versa. The total number of grid sizes and how they split the image can been seen in figure 4.11.

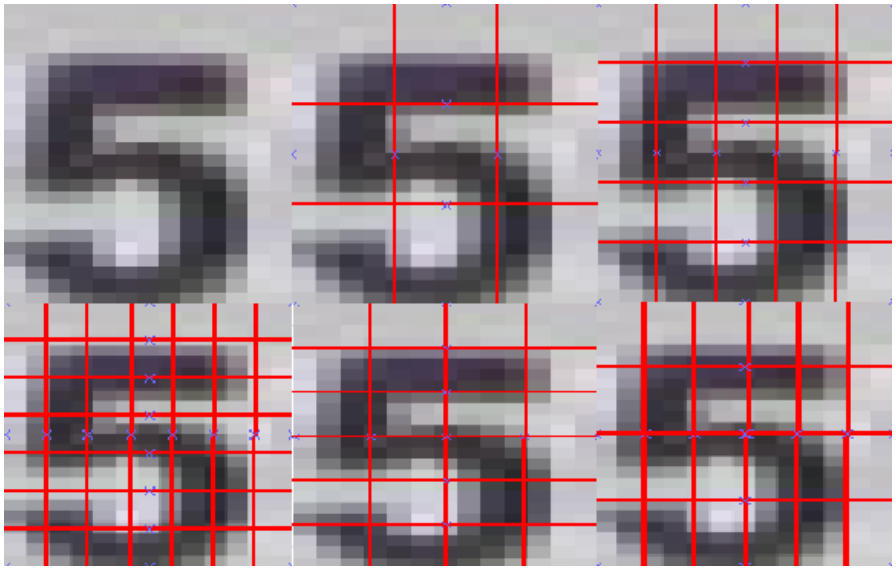

FIGURE 4.11: Grid combinations
(top) original image, 3x3 grid, 5x5 grid,
(bottom) 7x7 grid, 6x4 grid, 4x6 grid

		accuracy for different grids				
trained on	tested on	3x3	4x6	5x5	6x4	7x7
chars74k	icdar train	71.95%	75.95%	76.36%	76.66%	74.51%
chars74k	icdar test	70.24%	74.22%	74.12%	73.95%	72.21%
icdar train	chars74k	73.69%	76.96%	77.43%	76.81%	75.69%
icdar train	icdar test	74.72%	78.55%	79.20%	79.07%	77.88%
combined	icdar test	78.62%	82.11%	**82.60%**	82.26%	81.52%

TABLE 4.6: Character recognition results for different grids

In table 4.6, we have the results of this experiment. We trained and tested our classifier on different datasets each time, in order to have more robust results. The classifier, which is trained on both chars74k and ICDAR train datasets, is referred as 'combined'. We can see that the grid with five rows and five columns is the most informative, as it outperforms the rest constantly. Finally, the drop in the performance for the grid with seven rows and seven columns, can be explained as a case of overfitting.

Table 4.7 shows the character recognition results of different approaches on the ICDAR 2003 robust OCR dataset. We can see that the proposed approach is only outperformed by the co-occurence histogram of oriented gradients by Tian et al.[20]. This can be explained from the fact that co-hog captures tha spatial relationship among the HOG features extracted by each of the grid cells.

method	accuracy
ABBYY FineReader 10 [50]	26.6
HOG + NN[19]	51.5
NATIVE FERNS[16]	64.0
MSER[51]	67.0
Region-based feature pooling [52]	79.0
proposed	**79.2**
co-hog (case sensitive)[20]	79.4
co-hog (case insensitive)[20]	83.6

TABLE 4.7: Character recognition results on ICDAR 2003 cropped character dataset

4.3 Word Recognition

Word recognition is the task given an image containing a word to recognize which word is this. Since our pipeline is starting by detecting candidate character areas and then combining them exhaustively into words, for the word recognition a first approach would be to apply character recognition on the nodes of the returned paths. But, as we explained in section 3.3.3, our path extraction approach is focused on the candidate word areas and not on the sequence of the nodes. So, the accuracy of 24% that we get when we try this without the refinement of the spellchecker seams reasonable.

4.3.1 Dataset

For the purposes of this task, we use another dataset from ICDAR 2003, the Robust Word Recognition dataset, which contains 1110 single word images. Like previous works [23, 25, 26], we will test our algorithm on a subset of this dataset, where we exclude single character words and words that contain non-alphanumeric characters. This subset has a total of 865 single word images.

4.3.2 Cropped word recognition

Applying our algorithm on the word images, assuming that the word is with light text on dark background and vice versa, we have a total of 11881 recognitions for the 865

words of the groundtruth, of which 283 are correct (32%). By applying the spell checker function, we are able to recognize correctly 498 out of 865 words, with 7956 recognitions and the rest are discarded as non existent words. At the end, when we keep only the first suggestion of the ranked list returned by the spell-checker, we can recognize correctly 410 words with 4023 recognitions. In that way, we have a drop in the recall, since the correct word isn't always the first suggestion, but this also reduces the number of recognitions by almost 50%.

When we try to analyze the word recognition results, we are able to identify three main reasons of failure. One of them is the poor quality or the low resolution of the input word images. This leads to sparse local ultima, which in turn leads to wrong segmentation (figure 4.12). Another reason for failed word recognition is the misrecognition of more than one characters in a word. For example, in figure 4.13, although we have the correct segments, the misrecognition of the first '1' as 'J' and that of the second '1' as 'T', leads to a total word recognition of '3JT'. Even the spell checker step is unable to correct this case. One final reason, is the wrong order of segment removal. As we explained, we start by applying character recognition on all the initial character segments, and we continue by removing the lower local maximum or the higher local minimum, which isn't correct always, as you can see in figure 4.14.

recognition method	# recognitions	correct recognitions	Precision	Recall	F-measure
without dictionary	11881	283	0.0238	0.3271	0.0443
with dictionary, all suggestions	7956	498	0.0625	0.5757	0.1127
with dictionary, first suggestion	4023	410	0.1019	0.4739	0.1677

TABLE 4.8: Cropped word recognition results

method	accuracy
proposed	**57.5**
SYNTH + PLEX[53]	62.0
TSM + CRF[23]	70.47
TSM + PLEX[23]	79.30
Strokelets [10]	80.33

TABLE 4.9: Word recognition results on ICDAR 2003 Robust Word Recognition dataset

In table 4.9, we can see that our proposed approach has a poor performance and it is outperformed by any of the related approaches. This happens, because we chose a simple, training-free approach to find the character segments inside the word image. This part is open for future research.

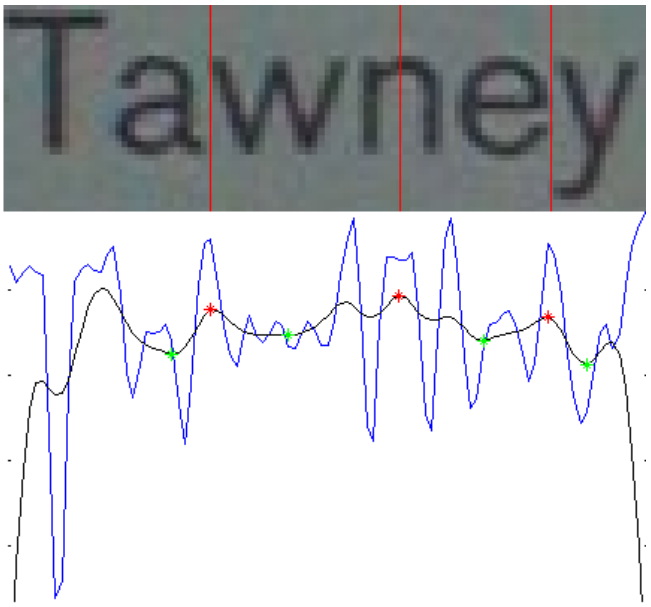

FIGURE 4.12: Example of failed word recognition because of bad segmentation

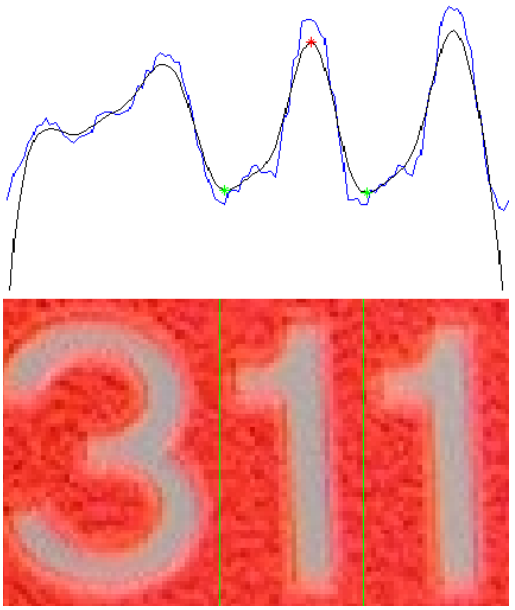

FIGURE 4.13: Example of failed word recognition because of failed character recognition

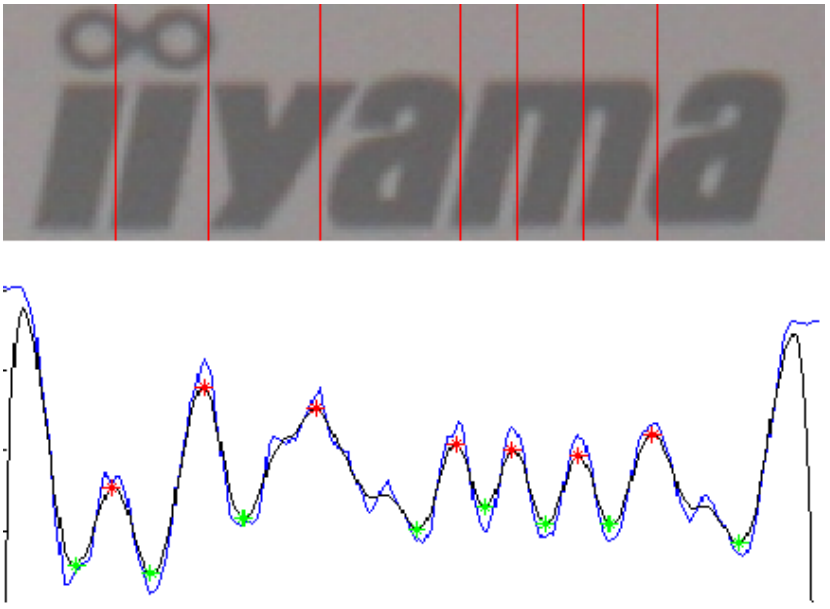

FIGURE 4.14: Example of failed word recognition because of wrong order of segment removal

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this chapter we summarize our research. For each of the tasks we discuss about the proposed methods and we analyze the results. Then, we give possible directions for future research.

5.1.1 Text detection

The main contribution of our research is focused on reducing the search space during the text detection. Instead of the commonly used sliding window approach, we perform a guided detection towards locating the characters. Then we combine them, based on specific criteria, in order to form words, as the Selective Search is suggesting. During this process, we use a variety of color channels, since each one behaves differently under different illumination conditions, and then we apply two different text detection algorithms. An advantage of this approach is that doesn't require any training, so it can be applied as it is. We also analyze the detection performance of each color channel and each one of the two detection algorithms that we use. The number of word bounding box proposals is on average one fourth of what Hosang et al. are expecting in [3].

5.1.2 Character recognition

For the character recognition, we followed a simple but effective approach. We combine the information of two different datasets for character recognition. We extract the informative HoG features from each cell after applying a dense grid over the character images, and we train an SVM classifier. The results are really promising.

5.1.3 Word Recognition

For the word recognition task, we apply a simple idea to segment a word image into characters, that also doesn't require any training. Then, we apply the pre-trained character recognition classifier on the possible segments to recognize the word. After removing the most probable false segmentation, we repeat the whole recognition process. At the end, we refine the predicted sequence of characters with a spell-checker function.

5.2 Future work

A possible direction for future work, can be the use of even more color channels, or the avoidance of using the least informative and most noisy color channels. This won't add any complexity to the overall detection pipeline, since each color channel is extracted and processed in parallel. After observing the results of the experiments, we strongly believe that a preprocessing of the input images, as proposed by Pan et al. in [54], would help the detection and recognition. Furthermore, it is necessary to reduce the number of the false positives outputs of the text detection. With the proposed method, we have on average 2500 candidate words for each image, which is one fourth of what Hosang et al. are expecting in [3]. Future researchers should find a way to reduce this number even more, without decreasing the detection accuracy.

For the character recognition task, our method proved to be really effective. Our pipeline splits the character image uniformly. A future researcher can examine different combinations of grid sizes, even with unequal cells or with overlap.

For the word recognition task, we apply a brute-force word segmentation, since we segment the word in two ways: firstly as light text on dark background, and later as dark text on light background. A possible improvement would be the use of a classifier able to classify the input word image either as dark text on light background or light text on dark background. That would reduce the number of recognitions to half, since we would focus only on the local minima or the local maxima, depending on the result of the classification. Also, patterns on the vertical profile, which gives the segments, may lead to more effective false segmentation removal, like in [55], where a neural network is trained to find the most appropriate segment positions. Finally, the expansion of the spell-checker dictionary is essential for broader use.

Bibliography

[1] Alain Trémeau, Christoph Godau, Sezer Karaoglu, and Damien Muselet. Detecting text in natural scenes based on a reduction of photometric effects: Problem of color invariance. In Raimondo Schettini, Shoji Tominaga, and Alain Trémeau, editors, *CCIW*, volume 6626 of *Lecture Notes in Computer Science*, pages 214–229. Springer, 2011. ISBN 978-3-642-20403-6. URL http://dblp.uni-trier.de/db/conf/cciw/cciw2011.html#TremeauGKM11.

[2] Alain Trémeau, Basura Fernando, Sezer Karaoglu, and Damien Muselet. Detecting text in natural scenes based on a reduction of photometric effects: Problem of text detection. In Raimondo Schettini, Shoji Tominaga, and Alain Trémeau, editors, *CCIW*, volume 6626 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 2011. ISBN 978-3-642-20403-6. URL http://dblp.uni-trier.de/db/conf/cciw/cciw2011.html#TremeauFKM11.

[3] J. Hosang, R. Benenson, and B. Schiele. How good are detection proposals, really? In *BMVC*, 2014.

[4] Jie Yang, Jiang Gao, Ying Zhang, Xilin Chen, and Alex Waibel. An automatic sign recognition and translation system. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, PUI '01, pages 1–8, New York, NY, USA, 2001. ACM. doi: 10.1145/971478.971490. URL http://doi.acm.org/10.1145/971478.971490.

[5] A. GonzÃ¡lez, Luis Miguel Bergasa, J. Javier Yebes, and Javier AlmazÃ¡n. Traffic panels detection using visual appearance. pages 1221–1226, Gold Coast, Australia, June 2013. doi: 10.1109/IVS.2013.6629633.

[6] Ali Zandifar and Antoine Chahine. A video based interface to textual information for the visually impaired. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, ICMI '02, pages 325–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1834-6. doi: 10.1109/ICMI.2002.1167016. URL http://dx.doi.org/10.1109/ICMI.2002.1167016.

[7] Michele Merler, Carolina Galleguillos, and Serge Belongie. Recognizing groceries in situ using in vitro training data. In *SLAM*, Minneapolis, MN, 2007.

[8] Ming Zhao, Shutao Li, and James Kwok. Text detection in images using sparse representation with discriminative dictionaries. *Image Vision Comput.*, 28(12): 1590–1599, December 2010. ISSN 0262-8856. doi: 10.1016/j.imavis.2010.04.002. URL http://dx.doi.org/10.1016/j.imavis.2010.04.002.

[9] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *European Conference on Computer Vision*, 2014.

[10] Cong Yao, Xiang Bai, Baoguang Shi, and Wenyu Liu. Strokelets: A learned multi-scale representation for scene text recognition. June 2014.

[11] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR*, pages 2963–2970. IEEE, 2010. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#EpshteinOW10.

[12] M. R. Lyu, Jiqiang Song, and Min Cai. A comprehensive method for multilingual video text detection, localization, and extraction. *IEEE Trans. Cir. and Sys. for Video Technol.*, 15(2):243–255, February 2005. ISSN 1051-8215. doi: 10.1109/TCSVT.2004.841653. URL http://dx.doi.org/10.1109/TCSVT.2004.841653.

[13] Kwang In Kim, Keechul Jung, and Jin Hyung Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(12):1631–1639, December 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1251157. URL http://dx.doi.org/10.1109/TPAMI.2003.1251157.

[14] Huizhong Chen, Sam S. Tsai, Georg Schroth, David M. Chen, Radek Grzeszczuk, and Bernd Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *2011 IEEE International Conference on Image Processing*, Brussels, September 2011.

[15] Lluis Gomez and Dimosthenis Karatzas. Mser-based real-time text detection and tracking. In *22nd International Conference on Pattern Recognition*, 2014.

[16] Tao Wang, David J. Wu, Adam Coates, and Andrew Y. Ng. End-to-end text recognition with convolutional neural networks.

[17] Zohra Saïdane and Christophe Garcia. Automatic Scene Text Recognition using a Convolutional Neural Network. In *International Workshop on Camera-Based Document Analysis and Recognition (CBDAR 2007)*, pages 100–106, September 2007. URL http://liris.cnrs.fr/publis/?id=6094.

[18] C. Strouthopoulos and N. Papamarkos. Text identification for document image analysis using a neural network. *Image and Vision Computing*, 16 (12–13):879 – 896, 1998. ISSN 0262-8856. doi: http://dx.doi.org/10.1016/S0262-8856(98)00055-9. URL http://www.sciencedirect.com/science/article/pii/S0262885698000559.

[19] Kai Wang and Serge Belongie. Word spotting in the wild. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 591–604, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15548-0, 978-3-642-15548-2. URL http://dl.acm.org/citation.cfm?id=1886063.1886108.

[20] Shangxuan Tian, Shijian Lu, Bolan Su, and Chew Lim Tan. Scene text recognition using co-occurrence of histogram of oriented gradients.

[21] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.

[22] Yuanping Zhu, Jun Sun, and Satoshi Naoi. Recognizing natural scene characters by convolutional neural network and bimodal image enhancement. In Masakazu Iwamura and Faisal Shafait, editors, *Camera-Based Document Analysis and Recognition*, volume 7139 of *Lecture Notes in Computer Science*, pages 69–82. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29363-4. doi: 10.1007/978-3-642-29364-1_6. URL http://dx.doi.org/10.1007/978-3-642-29364-1_6.

[23] Cunzhao Shi, Chunheng Wang, Baihua Xiao, Yang Zhang, Song Gao, and Zhong Zhang. Scene text recognition using part-based tree-structured character detection. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:2961–2968, 2013. ISSN 1063-6919. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2013.381.

[24] Anand Mishra, Karteek Alahari, and C.V. Jawahar. Scene Text Recognition using Higher Order Language Priors. In *BMVC 2012 - 23rd British Machine Vision Conference*, Surrey, United Kingdom, September 2012. BMVA. doi: 10.5244/C.26.127. URL https://hal.inria.fr/hal-00818183.

[25] Tatiana Novikova, Olga Barinova, Pushmeet Kohli, and Victor Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *ECCV*, pages 752–765, 2012. URL http://link.springer.com/content/pdf/10.1007%2F978-3-642-33783-3_54.pdf.

[26] A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012.

[27] Jon Almazan, Albert Gordo, Alicia Fornes, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99), 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2014.2339814.

[28] Albert Gordo. Supervised mid-level features for word image representation. *CoRR*, abs/1410.5224, 2014. URL http://arxiv.org/abs/1410.5224.

[29] Vibhor Goel, Anand Mishra, Karteek Alahari, and C. V. Jawahar. Whole is greater than sum of parts: Recognizing scene text words. In *ICDAR*, pages 398–402. IEEE, 2013. URL http://dblp.uni-trier.de/db/conf/icdar/icdar2013.html#GoelMAJ13.

[30] B. A. Wandell. *Foundations of Vision*. Sinauer Associates, Inc., 1995.

[31] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010. URL https://ivi.fnwi.uva.nl/isis/publications/2010/vandeSandeTPAMI2010.

[32] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013. doi: 10.1007/s11263-013-0620-5. URL http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.

[33] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002.

[34] Sezer Karaoglu, JanC. van Gemert, and Theo Gevers. Object reading: Text recognition for object recognition. In Andrea Fusiello, Vittorio Murino, and Rita Cucchiara, editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, volume 7585 of *Lecture Notes in Computer Science*, pages 456–465. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33884-7. doi: 10.1007/978-3-642-33885-4_46. URL http://dx.doi.org/10.1007/978-3-642-33885-4_46.

[35] S. Karaoglu, J. C. van Gemert, and T. Gevers. Con-text: Text detection using background connectivity for fine-grained object classification. In *ACM International Conference on Multimedia*, 2013. URL https://ivi.fnwi.uva.nl/isis/publications/2013/KaraogluICM2013.

[36] J. E. Hopcroft and R. E. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[37] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE, 2005.

[38] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, September 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.167. URL http://dx.doi.org/10.1109/TPAMI.2009.167.

[39] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[40] Kaibo Duan and S. Sathiya Keerthi. Which is the best multiclass SVM method? an empirical study. In *Multiple Classifier Systems, 6th International Workshop, MCS 2005, Seaside, CA, USA, June 13-15, 2005, Proceedings*, pages 278–285, 2005. doi: 10.1007/11494683_28. URL http://dx.doi.org/10.1007/11494683_28.

[41] JP Vert, K Tsuda, and B Schölkopf. *A Primer on Kernel Methods*, pages 35–70. MIT Press, Cambridge, MA, USA, 2004. URL http://www.is.tuebingen.mpg.de/fileadmin/user_upload/files/publications/pdf2549.pdf.

[42] Cunzhao Shi, Chunheng Wang, Baihua Xiao, Yang Zhang, Song Gao, and Zhong Zhang. Scene text recognition using part-based tree-structured character detection. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:2961–2968, 2013. ISSN 1063-6919. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2013.381.

[43] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):690–706, July 1996. ISSN 0162-8828. doi: 10.1109/34.506792. URL http://dx.doi.org/10.1109/34.506792.

[44] Victor Lavrenko, Toni M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04)*, DIAL '04, pages 278–, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2088-X. URL http://dl.acm.org/citation.cfm?id=968882.969387.

[45] the Lucene search engine spell checker. http://wiki.apache.org/lucene-java/SpellChecker.

[46] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[47] Simon M. Lucas. Text locating competition results. In *ICDAR*, pages 80–85. IEEE Computer Society, 2005. ISBN 0-7695-2420-6. URL http://dblp.uni-trier.de/db/conf/icdar/icdar2005.html#Lucas05.

[48] Rodrigo Minetto, Nicolas Thome, Matthieu Cord, Jonathan Fabrizio, and Beatriz Marcotegui. Snoopertext: A multiresolution system for text detection in complex visual scenes. In *ICIP*, pages 3861–3864. IEEE, 2010. ISBN 978-1-4244-7994-8. URL http://dblp.uni-trier.de/db/conf/icip/icip2010.html#MinettoTCFM10.

[49] Yi-Feng Pan, Xinwen Hou, and Cheng-Lin Liu. Text localization in natural scene images based on conditional random field. *2013 12th International Conference on Document Analysis and Recognition*, 0:6–10, 2009. doi: http://doi.ieeecomputersociety.org/10.1109/ICDAR.2009.97.

[50] ABBYY FineReader 10. http://www.abbyy.com/.

[51] Lukas Neumann and Jiri Matas. A method for text localization and recognition in real-world images. In *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part III*, ACCV'10, pages 770–783, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-19317-0. URL http://dl.acm.org/citation.cfm?id=1966049.1966110.

[52] Chen-Yu Lee, Anurag Bhardwaj, Wei Di, Vignesh Jagadeesh, and Robinson Piramuthu. Region-based discriminative feature pooling for scene text recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 4050–4057, 2014. doi: 10.1109/CVPR.2014.516. URL http://dx.doi.org/10.1109/CVPR.2014.516.

[53] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2011.

[54] Zhixun Su Jinshan Pan, Zhe Hu and Ming-Hsuan Yang. Deblurring text images via l0-regularized intensity and gradient prior. 2014.

[55] Amjad Rehman, Dzulkifli Mohamad, and Ghazali Sulong. Implicit vs explicit based script segmentation and recognition: A performance comparison on Benachmark Database. *Int. J. Open Problems Compt. Math*, 2, 2009.