UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Departamento de Electrónica

Text Detection and Recognition in Natural Images using Computer Vision Techniques

A Thesis submitted for the degree of Doctor of Philosophy

Author

Álvaro González Arroyo

Supervisor

Dr. D. Luis Miguel Bergasa Pascual

2013

A mi familia

Agradecimientos

En primer lugar quisiera darle las gracias a Luis Miguel Bergasa Pascual por la oportunidad que me ofreció para realizar esta tesis doctoral, por las innumerables revisiones de artículos y documentos, las reuniones de seguimiento y por permitirme disfrutar de los congresos a los que he asistido (Vigo, USA, Portugal, Japón), los congresos en los que he colaborado (el IV'12 en Alcalá) y por las estancias de investigación que he realizado (Inglaterra, Alemania), los cuales me han permitido conocer numerosos lugares y distintas culturas y me han hecho crecer como persona y, porqué no decirlo, sentirme algo más pequeño dentro de este mundo. Ya solo por esto, estos largos años de doctorado han merecido la pena.

Quiero tener una mención especial a Miguel Ángel Sotelo, del cual, aunque no haya sido mi tutor, he aprendido muchas cosas durante estos cinco años que he estado trabajando en el grupo de investigación Robesafe, en especial a trabajar en equipo y a colaborar en proyectos de investigación grandes e importantes, por su sentido del humor, por su iniciativa y por permitirme haber formado parte de ese proyecto que ha sido Vision Safety Technologies.

Durante estos años han pasado muchas personas por Robesafe. De todos ellos he aprendido mucho y he compartido muy buenos momentos con ellos. Gracias a Fer y sus chistes malos (y algunos no tan malos, todo hay que decirlo) que animan el laboratorio O202, a Ángel, Estefanía, Eduardo, al casado Yebes, a Almazán, Gavilán, Balky, Óscar, Carlos, Jesús, Pablo, Sebas, Iván, Noelia, Pedro, Llorca, Nacho, Raúl, Sergio, Garrido, Manuel Ocaña, y un largo etcétera.

Tengo muchísimo que agradecer a Krystian Mikolajczyk. Krystian me dio la oportunidad de acudir a su laboratorio en Guildford cuando aún no sabía casi nada de visión. Durante los tres meses que estuve trabajando con él aprendí mucho y fue capaz de orientar mi tesis cuando me encontraba un poco perdido sobre qué camino seguir. La experiencia vivida en Guildford fue de las mejores de mi vida, no solo en el plano laboral, sino sobre todo en el personal. Allí conocí a muchas personas a las que no quería dejar de mencionar, como a Edu (the Mozart of Guildford), Akshay, Raquel, Carmen, Lorena (nunca olvidaré ese "crucero" que tuvimos que tomar para poder volver a la madre patria), a Alberto, Leti, Dani, Laura (por esos *language exchanges* sin los que no habría podido aprender inglés tan rápido) y a todos los que me olvido, que sé que son muchos.

Gracias a Hermann Ney por permitirme visitar y trabajar en el laboratorio que él dirige en Aachen, y por todas las facilidades que me dio. Allí aprendí mucho y fue donde empezó a gestarse el final de esta tesis.

También quiero tener un especial agradecimiento al profesor Javier Macías, quien tuvo la amabilidad de introducirme de forma altruista en el mundo de la Programación Dinámica. Sin su ayuda, una buena parte de esta tesis no habría sido posible.

Sin duda a las personas a las que más tengo que agradecer son a mi familia, mis padres y mi hermano, por el apoyo constante durante todos estos años, por creer en mi y por la

vida que me han dado. Espero haber sido un buen hijo.

Y por último, y no menos importante, tengo mucho que agradecer a una persona que apareció hace unos meses en mi vida y que, sin duda, es la persona más especial que he tenido en mi vida y que tengo hoy en día. Llegó por casualidad, como mucha gente en la vida, y desde entonces me ha hecho muy feliz. Espero que yo también a ella y que sigamos juntos por mucho tiempo. Con ella estoy creciendo como persona y por ello tengo mucho que agradecerle. Gracias por estar ahí, Luz María. En la vida hay tinieblas, pero también hay luces. Y tú eres la luz de toda luz.

El secreto de la genialidad es el de conservar el espíritu del niño hasta la vejez, lo cual quiere decir nunca perder el entusiasmo.

Aldous Huxley.

Resumen

El reconocimiento de texto en imágenes reales ha centrado la atención de muchos investigadores en todo el mundo en los últimos años. El motivo es el incremento de productos de bajo coste como teléfonos móviles o Tablet PCs que incorporan dispositivos de captura de imágenes y altas capacidades de procesamiento. Con estos antecedentes, esta tesis presenta un método robusto para detectar, localizar y reconocer texto horizontal en imágenes diurnas tomadas en escenarios reales. El reto es complejo dada la enorme variabilidad de los textos existentes y de las condiciones de captura en entornos reales. Inicialmente se presenta una revisión de los principales trabajos de los últimos años en el campo del reconocimiento de texto en imágenes naturales. Seguidamente, se lleva a cabo un estudio de las características más adecuadas para describir texto respecto de objetos no correspondientes con texto.

Típicamente, un sistema de reconocimiento de texto en imágenes está formado por dos grandes etapas. La primera consiste en detectar si existe texto en la imagen y de localizarlo con la mayor precisión posible, minimizando la cantidad de texto no detectado así como el número de falsos positivos. La segunda etapa consiste en reconocer el texto extraído.

El método de detección aquí propuesto está basado en análisis de componentes conexos tras aplicar una segmentación que combina un método global como MSER con un método local, de forma que se mejoran las propuestas del estado del arte al segmentar texto incluso en situaciones complejas como imágenes borrosas o de muy baja resolución. El proceso de análisis de los componentes conexos extraídos se optimiza mediante algoritmos genéticos. Al contrario que otros sistemas, nosotros proponemos un método recursivo que permite restaurar aquellos objetos correspondientes con texto y que inicialmente son erróneamente descartados. De esta forma, se consigue mejorar en gran medida la fiabilidad de la detección. Aunque el método propuesto está basado en análisis de componentes conexos, en esta tesis se utiliza también la idea de los métodos basados en texturas para validar las áreas de texto detectadas.

Por otro lado, nuestro método para reconocer texto se basa en identificar cada caracter y aplicar posteriormente un modelo de lenguaje para corregir las palabras mal reconocidas, al restringir la solución a un diccionario que contiene el conjunto de posibles términos. Se propone una nueva característica para reconocer los caracteres, a la que hemos dado el nombre de Direction Histogram (DH). Se basa en calcular el histograma de las direcciones del gradiente en los pixeles de borde. Esta característica se compara con otras del estado del arte y los resultados experimentales obtenidos sobre una base de datos compleja muestran que nuestra propuesta es adecuada ya que supera otros trabajos del estado del arte. Presentamos también un método de clasificación borrosa de letras basado en KNN, el cual permite separar caracteres erróneamente conectados durante la etapa de segmentación. El método de reconocimiento de texto propuesto no es solo capaz de reconocer palabras, sino también números y signos de puntuación. El reconocimiento de palabras se lleva a cabo mediante un modelo de lenguaje basado en inferencia probabilística y el British National Corpus, un completo dictionario del inglés británico moderno, si bien el algoritmo puede ser fácilmente adaptado para ser usado con cualquier otro diccionario. El modelo de lenguaje utiliza una modificación del algoritmo forward usando en Modelos Ocultos de Markov.

Para comprobar el rendimiento del sistema propuesto, se han obtenido resultados experimentales con distintas bases de datos, las cuales incluyen imágenes en diferentes escenarios y situaciones. Estas bases de datos han sido usadas como banco de pruebas en la última década por la mayoría de investigadores en el área de reconocimiento de texto en imágenes naturales. Los resultados muestran que el sistema propuesto logra un rendimiento similar al del estado del arte en términos de localización, mientras que lo supera en términos de reconocimiento.

Con objeto de mostrar la aplicabilidad del método propuesto en esta tesis, se presenta también un sistema de detección y reconocimiento de la información contenida en paneles de tráfico basado en el algoritmo desarrollado. El objetivo de esta aplicación es la creación automática de inventarios de paneles de tráfico de países o regiones que faciliten el mantenimiento de la señalización vertical de las carreteras, usando imágenes disponibles en el servicio Street View de Google. Se ha creado una base de datos para esta aplicación. Proponemos modelar los paneles de tráfico usando apariencia visual en lugar de las clásicas soluciones que utilizan bordes o características geométricas, con objeto de detectar aquellas imágenes en las que existen paneles de tráfico. Los resultados experimentales muestran la viabilidad del sistema propuesto.

Abstract

Reading text in real-world scene images has focused the attention of many researchers all over the world during the last few years. The reason is the increasingly availability of cheap image-capturing devices in low-cost products such as smartphones and Tablet PCs. For this reason, this thesis presents a robust method to detect, locate and recognize horizontally-aligned text in natural images taken in real-world scenarios at daytime. This is a complex challenge due to the huge variability of text appearance and the capturing conditions in real scenarios. Initially a review of the main works of the last years in the field of text reading in real-world scene images is presented. Then, we carry out a study of the most suitable features to describe text versus non-text components.

A computer vision system for reading text in images typically is composed of two main stages. Firstly, a text location method is applied in order to detect if text is present in the image and to locate it with the highest precision possible and minimizing the amount of undetected text as well as the number of false positives. Secondly, a text recognition algorithm is applied in order recognize the extracted text.

The text location method here proposed is based on a connected-component analysis applied after a segmentation process, which combines a global method like MSER with a locally adaptive thresholding algorithm that improves the existing approaches by segmenting text even when blur motion is present in the images or if their resolution is too small. The connected component analysis process is optimized using genetic algorithms. Unlike other methods, we also propose a recursive method to restore character connected components initially erroneously discarded. This allows to improve the accuracy of the detection. Although the proposed system is based on connected component analysis, some ideas used on texture-based methods are also used in our approach.

On the other hand, our approach to recognize text is based on identifying single characters and then applying a language model to correct misspelled words, constraining the output to a dictionary of all the possible terms. A new feature based on gradient direction histogramming, which we name as Direction Histogram (DH), is proposed to characterize single letters. This new feature is compared to other state-of-the-art features and the experimental results obtained on a challenging dataset show that the proposed feature is more than adequate as it outperforms the results achieved in the state of the art. We present a fuzzy classification method based on KNN, which is useful to separate characters that can be wrongly connected during the segmentation process. The recognition method here proposed is able to recognize not only words, but also numbers and punctuation marks. The word recognition is carried out using a language model based on probabilistic inference on the British National Corpus, a dictionary of modern British English, although the algorithm can be easily adapted to be used with any other dictionary. The language model uses a modification of the forward algorithm used in Hidden Markov Models (HMMs).

To check the performance of the proposed system, experimental results have been obtained with several datasets that include images in different scenarios and situations. These datasets have been used as a benchmark for most of the researchers in the area of text reading in natural images during the last decade. The results show that the proposed system achieves state-of-the-art performance in terms of text location, while it overpass the state-of-the-art results in text recognition.

In order to show the applicability of the method proposed in this thesis, a system to detect and recognize traffic panels based on the developed text reading method is presented in this thesis. The aim of this application is to automatically create inventories of traffic panels of regions or countries that facilitate traffic signposting maintenance using images downloaded from Google Street View service. A dataset has been created for this purpose. We propose to model traffic panels using visual appearance instead of the classic approaches that use edge detection or geometrical characteristics, in order to detect the images where traffic panels are present. The experimental results show the viability of the proposed system.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Automatic text recognition has traditionally focused on analyzing scanned documents. Since Emanuel Goldberg presented in 1914 a machine that read characters and converted them into standard telegraph code, many technologies have been developed to convert images of scanned documents into machine-encoded text, so that they can be electronically stored and used in machine translation and text-to-speech processes. Many free and licensed systems that achieve a high performance have been created and commercialized. However, recognition of Latin-script, typewritten text in scanned documents is still not 100% accurate even where clear imaging is available. One study based on recognition of 19th- and early 20th-century newspaper pages concluded that accuracy of commercial software varied from 71% to 98% [Holley, 2009]. Total accuracy can be achieved only with human review. Other areas, including recognition of hand printing, cursive handwritting and printed text in other scripts (especially those East Asian language characters which have many strokes for a single character), are still the subject of active research.

However, commercial systems are not reliable for camera-captured real-world scenes and automatic text recognition in real-world images still remains one of the most challenging problems in computer vision due to complex backgrounds, uncontrolled lighting conditions and wide variety of text appearance. During the last years digital cameras have started to be embedded in low-cost consumer products such as smartphones and Tablet PCs, so that user applications related to digital image processing have become very popular, and the range of applications of automatic text reading systems has expanded, from support to robotic navigation in indoor and outdoor scenarios, image spam filtering, driver assistance or translation services for tourists, among others, since textual information can be found on any environment, both indoors and outdoors. For instance, figure 1.1 depicts different topics where text reading systems can have a tremendous applicability. Some recent applications in robotics and ICT (Information and Telecommunication Technologies) are:

- A system embedded in a PDA or smartphone to help visually handicapped people [Mancas-Thillou et al., 2007].
- A head-mounted device to aid visually impaired persons [Merino et al., 2011].
- Indoor [Liu and Samarabandu, 2005] and outdoor [Posner et al., 2010] mobile robot navigation.
- A translator robot [Shi and Xu, 2005].
- A PDA-based sign translator [Zhang et al., 2002].

- An automatic comic reader on mobile phones [Yamada et al., 2004].
- A translator of signboard images from English to Spanish [Rodríguez et al., 2009].
- A word translator from English to Spanish and vice versa [Press release, 2010].
- A street sign recognizer for geolocalization [Parizi et al., 2009].

(a) Smartphone (b) Humanoid robot (c) A visually impaired user

(d) Intelligent vehicle

Figure 1.1: Examples of different scenarios that would benefit from text reading applications.

The applications of text recognition to Intelligent Transportation Systems (ITS) can be also multiple. Automatic text recognition could be useful to support drivers or autonomous vehicles to find a certain place by simply reading and interpreting street signs, road panels, variable-message signs or any kind of text present in the scenario, when Global Positioning Systems (GPS) suffer from lack of coverage, especially in high-density urban areas. Advanced Driver Assistance Systems (ADAS) could also benefit from text recognition for automatic traffic signs and panels identification. In addition, textual information could be also fused with other data obtained from image or RADAR in order to make more robust systems.

Up to now, most of works on text reading in natural images have focused on concrete subsets of the problem, such as extracting text in CD cover images [Escalera et al., 2009] or segmenting text in web images [Karatzas and Antonacopoulos, 2007]. This is due to the wide variety of text appearance because of different fonts, thicknesses, colors, sizes, textures, lighting conditions, image resolutions, languages, etc., together with the presence of geometrical distortions, partial occlusions and different shooting angles that can cause deformed text.

The aim of this thesis is to develop a computer vision method able to read text in any kind of scenario, both indoors and outdoors, and in any kind of image, both natural and born-digital images, with the highest precision possible and minimizing the amount of undetected text as well as the number of false text extracted. We simply constrain to machine-printed horizontally-aligned text and English language.

To check the performance of the proposed system, we propose to obtain experimental results from several datasets that include images in different scenarios and situations. These datasets have been used as a benchmark for most of the researchers in the area of text reading in natural images during the last decade. The first dataset was released for the Robust Reading Competition held in the frame of the 7th International Conference on Document Analysis and Recognition (ICDAR) in 2003. The competition was divided into three subproblems: text location, character recognition and word recognition. In this thesis, we will show our results for the three problems. The first one received five entries, while there was no participants in the character recognition and word recognition problems. The same competition was carried out two years later in the following conference, using the same dataset. Again, the participants only took part in the text locating problem.

In order to check the improvements achieved in the area of text detection and recognition since 2005, a new competition was celebrated in the ICDAR 2011 edition. The competition was divided into two contests. The first one was aimed at detecting and reading text in born-digital images (web, email), while the second one was very similar to the competitions held in 2003 and 2005, as it was aimed at reading text in real-world images, so the same dataset was released at this time with slight modifications. The first challenge was organized over three tasks: text location, text segmentation and word recognition. On the other hand, the second challenge consisted of two tasks: text location and word recognition. In both contests, most of the entries were done for the text location task, while there was hardly any participant in the word recognition problem. In this thesis, we will show our results for all the tasks.

In order to show that the method proposed in this thesis is applicable and can be generalized to any kind of situations, results are also obtained using a dataset composed of CD and DVD covers. Moreover, a system to detect and recognize traffic panels based on our text reading proposal is also developed. This application fits one of the main investigation lines of the Robesafe Research Group [1], as it is the research in ITS. The aim of this application is to automatically create inventories of traffic panels of regions or countries in order to facilitate traffic signposting maintenance and driver assistance.

1.1 Document structure

This document is divided in several parts, of which the present introduction is the first one. Chapter 2 contains a brief review of the state of the art in text detection and recognition. That chapter aims to present a global summary of the approaches to text detection and recognition, without going into specific details. These are introduced as needed in the subsequent chapters, as related techniques and methods are described.

Chapter 3 presents the text localization method developed and tests its performance on different public datasets in order to get a reliable comparison with other methods of the state of the art. Similarly, chapter 4 explains the text recognition method implemented

[1] `http://www.robesafe.com/`

in this work, making a comparison with other methods and showing the results obtained with different publicly available datasets.

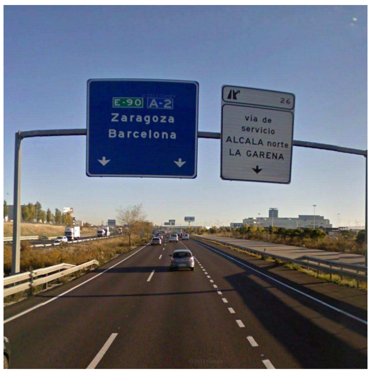
Chapter 5 describes a real application of the proposed text reading method to detect and recognize the information contained in traffic panels. The dataset created for this purpose using images from Google Street View is described in this chapter and detailed experimental results obtained with this dataset are shown.

Finally, chapter 6 contains the conclusions and main contributions of this work, and future research lines that may spring from it. The document is closed with the bibliography as well as with three appendices that show, on the one hand, the results obtained with one of the benchmarked datasets, and on the other hand, the explanation of the forward algorithm and the description of the UTM from/to Latitude and Longitude conversion equations.

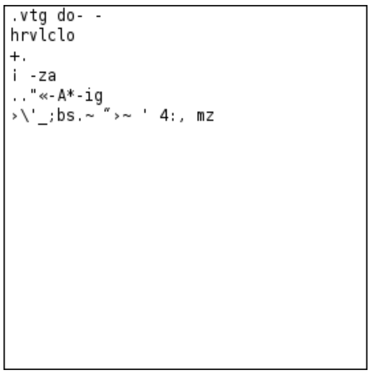
Chapter 2

State of the Art

Automatic text location and recognition in images has been one of the main challenges in computer vision ever. Most of the work in this field is based on optical character recognition (OCR), which consists of converting images of handwritten or printed text in scanned documents into machine-encoded text, so that they can be electronically shared, stored and displayed, as well as used in machine processes such as machine translation, text-to-speech and text mining. Many OCR systems have been developed and patented and achieve a very good performance when reading text of scanned images of documents. There are many free and licensed systems, such as Tesseract [Google, 2010] and ABBYY FineReader [ABBYY, 2009]. However, they do not work so well for camera-captured scenes, where the text is typically embedded in complex environments and text varies in size, style, color and layout. An example is shown in figure 2.1, where a public OCR, specifically Tesseract, has been applied over a natural image captured from Google Street View service and the results are unexpected.



(a) Input image



```
.vtg do- -
hrvlclo
+.
i -za
.."«-A*-ig
›\'_;bs.~ ”›~ ' 4:, mz
```

(b) OCR (Tesseract)

Figure 2.1: Text detection and recognition with a public OCR

In order to benchmark the state of the art in terms of text reading in natural images, a competition was held in the frame of the 7th International Conference on Document Analysis and Recognition (ICDAR) in 2003. Since then, many research groups have focused their attention in this field, using the datasets released for this competition and the subsequent ones as benchmark, and a huge improvement has been achieved in the last decade.

This chapter presents a brief survey of the state of the art in text detection and text recognition in natural images, both in terms of single character recognition and word recognition. This chapter does not intend to make an exhaustive review, as it would result in a lengthy chapter, both in time and space. The aim of what follows is to provide an overview of the most remarkable methods in each field of the last years, and those that are related to the contents of the following chapters of this thesis. This chapter closes with a discussion on the most adequate methods to be studied with their advantages and drawbacks, and the specific aims of this thesis.

2.1 Text detection

Text detection deals with the problem of finding if text appears in an image, and locating it if any is present. Almost every system that recognizes text in natural images has text detection as its first step.

Firstly, a general classification of text localization methods can be made into two main categories, as shown in figure 2.2: those methods based on a single frame and those based on multiple frames. The algorithms based on a single frame use implementations based either on connected component (CC) analysis [Yao et al., 2007; Neumann and Matas, 2012; Epshtein et al., 2010; Pan et al., 2011; Chen et al., 2011; Yi and Tian, 2011], on edge analysis [Liu et al., 2005; Shivakumara et al., 2008; Liu and Sarkar, 2008; Zhang and Kasturi, 2010], or on texture analysis [Chen and Yuille, 2004; Ye et al., 2007; Wang et al., 2009; Hanif and Prevost, 2009; Tu et al., 2006; Minetto et al., 2010; Pan et al., 2011]. On the other hand, the algorithms based on multiple frames can be split into multi-frame averaging methods [Hua et al., 2002; Wang et al., 2004], and time-based minimum pixel search methods [Sato et al., 1999]. Since the method proposed in this thesis is aimed at locating text on single images, the focus of this chapter are methods based on a single frame.

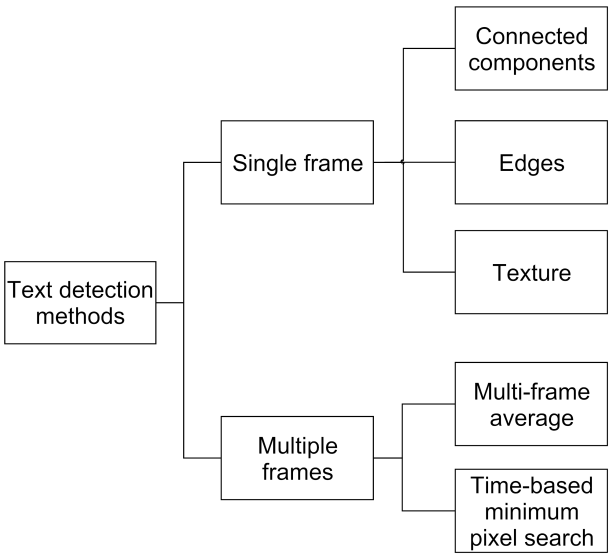


Figure 2.2: Classification of text detection methods

2.1.1 Connected component-based methods

The CC-based approaches segment a frame into multiple small CCs, apply different geometric constraints to discard non-text candidates and join the text candidates into several larger text regions by analizing their geometrical arrangement on the image and their common features. These methods usually consist of three stages:

1. CC extraction from segmented image.
2. CC analysis to filter out non-text components using heuristic rules or classifiers.
3. Post-processing to group text components into text blocks.

Image segmentation is a key step in order to achieve an accurate text detection. Typically, two approaches are used: local and global segmentation methods. The first one generates too many false positives, while the main drawback of the second approach is that it is not suitable to segment images whose illumination of the scenario is not homogeneous. [Yao et al., 2007] use locally adaptive thresholding to segment an image. Then, certain geometric features are extracted from CCs and used to discard most non-character CCs by a cascade of threshold classifiers. Non-discarded CCs are fed into a SVM in order to be classified into characters or non-characters. Finally, the character CCs verified by the classifier are merged into candidate text regions according to certain neighboring properties such as proximity, stroke width and height.

On the other hand,[Neumann and Matas, 2012] assume characters to be Maximally Stable Extremal Regions (MSERs) [Matas et al., 2002] in certain scalar image projections (intensity, red channel, blue channel, green channel). The resulting CCs of the MSER detection stage are classified into character and non-character based on certain basic features such as aspect ratio, compactness or color consistency. Then, text line candidates are formed based on geometric character attributes. Later, character recognition is applied together with a typographic model to correct inconsistencies.

[Merino and Mirmehdi, 2007] use an adaptive thresholding to initially binarize the image and to obtain CCs. However, in a later work ([Merino et al., 2011]), the same authors propose to use MSER for image segmentation. In both works, after the segmentation process, a tree is constructed representing the topological relationship between CCs in the binary image. The outermost region of the image is the root of the tree, while the innermost regions of the image are the leaves of the tree. Then, a hierarchical filtering of the tree nodes is carried out. This allows to reject many candidate regions without classification, because when a node has children already classified as text, it can be discarded as non-text. After that, the remaining tree nodes are filtered using a cascade of text classifiers related to size, aspect ratio, complexity, border energy and texture. Finally, CCs are grouped into text regions or lines taking into account the relative position and size of adjacent regions.

[Epshtein et al., 2010] propose the Stroke Width Transform (SWT), a local image operator which computes per pixel the width of the most likely stroke containing the pixel. A stroke is a contiguous part of an image that forms a band of a nearly constant width. The idea under the SWT is to look for matched pairs of pixels in a small region with corresponding opposing gradients. Pixels with similar stroke width are merged into CCs and letter candidates are found from certain geometric basic rules concerning aspect ratio, number of holes or variance of the stroke width. Then, letters are grouped into text lines if they have similar features such as stroke width, height or average color.

[Chen et al., 2011] follow the same idea of Ephstein's work. However, they only apply the SWT on the CCs resulting after a MSER detection stage. In addition, they propose an alternative way of computing the SWT from a binary image. The way of computing the SWT proposed by [Epshtein et al., 2010] relies on calculating the gradients along the edges of each CC. However, when the text is badly contrasted, the gradient may not be well determined and the calculation of the stroke width may fail. This problem is overcome by the proposal of [Chen et al., 2011], which calculates the distance that separate each pixel in the considered CC from the background using the distance transform, and then spread the maximum distances along the stroke width. They consider that the real width of the stroke is twice the resulted transform.

[Pan et al., 2011] propose an hybrid approach. Firstly, a text region detector is applied to estimate probabilities of the text position and scale information. This detector is based on Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] and a WaldBoost cascade classifier [Sochman and Matas, 2005], on image pyramids. The information extracted from each scale is merged into a single text confidence map and text scale map. Secondly, the gray-level image is segmented using the Niblack's local binarization algorithm [Niblack, 1986] and a CC analysis is carried out with a conditional random field (CRF) model [Lafferty et al., 2001] to assign candidate components as text and non-text by considering both unary component properties, such as width, height, aspect ratio and compactness, and binary contextual component relationships, such as spatial distance, overlap ratio and gray-level difference.

Several works, which have taken part recently in the robust reading competition held in the ICDAR 2011 conference, are presented in [Karatzas et al., 2011]. The one known as *TH-TextLoc* extracts CCs using an adaptive binarization method and candidates are classified into text or non-text using SVM and geometric, shape and stroke features. Then, text candidates are grouped into text regions analysing projection histograms. On the other hand, the *TDM IACAS* system computes the And-Ridge and And-Valley images proposed by [Shao et al., 2010] and resulting CCs are classified as character and non-character using a set of binary SVM classifiers using gradient features. Then, characters are grouped together using certain neighboring constraints. *OTCYMIST* binarises each channel R, G and B separately and extracts CCs in each binary image and its complement one. Resulting CCs are filtered using certain geometric characteristics. The system known as *SASA* is an hybrid approach, as it is explained in [Yi and Tian, 2011]. It extracts CCs from the magnitude gradient difference of the input image and does adjacent character grouping taking into account similar height and horizontal alignment between sibling components. Then, Haar features are extracted from gradient maps and stroke orientation maps by the block patterns presented in [Chen and Yuille, 2004]. These features are the input of an AdaBoost-based text classifier, which determines if the candidate patches are text regions or not.

Finally, several works have been recently presented in the 21st International Conference on Pattern Recognition (ICPR) in November 2012. [Fehli et al., 2012] uses MSER to extract text candidates. Then, a graph that connects similar and neighboring CCs is constructed. Geometry and color difference are used for constructing the graph. A text descriptor based on the stroke width variance is applied on each node of the graph in order to eliminate nodes that have very low probability to be text. [Chowdhury et al., 2012] obtain CCs after edge-linking on the Canny edgemap, and geometric, graylevel and color-based features are used to filter non-text regions using a multi-layer perceptron (MLP) classifier. [Li and Lu, 2012] also use a contrast-enhanced MSER algorithm to extract CCs, and simple geometric constraints, including stroke width, are applied to remove false positives. Another work that uses MSER to extract letter candidates is proposed by [Yin et al., 2012]. After elimination of non-letter candidates using geometric information, candidate regions are constructed by grouping similar CCs. Candidate region features based on horizontal and vertical variances, stroke width, color and geometry are extracted. An AdaBoost classifier is built from these features and text regions are identified. [Liu et al., 2012] utilize a multi-scale adaptive local thresholding operator to binarize the original image. CCs are then extracted and filtered using geometric features, and the obtained candidate components are checked on the word level by using a graph to represent spatial relation of different components. Scene text regions are localized by searching the collinear maximum group over the graph. [Zhang and Lai, 2012] use a Laplacian of Gaussian filter to smooth noise and CCs are extracted after applying the Otsu's binarization method [Otsu, 1979] and morphological closing to merge adjacent characters into whole regions. Then, the minimum moment of inertia of each candidate text region is computed, and the orientation and minimum bounding box of each CC are obtained. Based on the fact that corners are frequent and essential patterns in text regions, the work proposes a geodesic distance between corners and the skeleton of text regions to measure the effective distance between corners and text. Finally, a geodesic distance weighted corner saturation parameter is given to determine which candidate regions are the true text regions.

2.1.2 Edge-based methods

The edge-based methods separate between text and background detecting the borders between these regions as edges follow certain homogeneous patterns. Typically, the edges are detected by an edge filter and then merged by morphological operators. However, it is difficult to draw a distinction between edge-based methods and CC-based algorithms, as typically additional information like color or geometry is necessary to validate or discard the extracted regions in edge-based approaches.

[Liu et al., 2005] firstly apply edge detection to get four edge maps in horizontal, vertical, up-right and up-left direction. Secondly, the feature is extracted from these four edge maps to represent the texture property of text. Then k-means algorithm is applied to detect the initial text candidates. Finally, the text areas are identified by some heuristic rules and refined through project profile analysis. [Shivakumara et al., 2008] explore new edge features such as straightness for removing non-significant edges. It identifies text block candidates by combining Arithmetic Mean Filter, Median Filter and edge analysis.

[Liu and Sarkar, 2008] use an intensity histogram-based filter and an inner distance-based shape filter to extract text blocks and remove false positives whose intensity histograms are similar to those of their adjoining areas and the components coming from the same object. [Bai et al., 2008] use a multi-scale Harris-corner-based method to extract candidate text blocks. The position similarity and color similarity of Harris corners are used to generate boundaries of text objects.

[Zhang and Kasturi, 2010] propose a method based on edge gradients and Graph Spectrum. It firstly extracts text edges from an image and localizes candidate character blocks using HOG. Then, Graph Spectrum is used to capture global relationship among candidate blocks and to cluster candidate blocks into groups to generate bounding boxes of text objects in the image.

An edge-based work has taken part in the ICDAR 2011 robust reading competition [Karatzas et al., 2011]. The name of the system is *Textorter*. It extracts edges from a greyscale image and morphological operators are applied on the image aiming at connecting any broken edges. A filtering stage removes noise components based on their aspect ratio and size. Remaining components are classified as text or non text on the basis of features such as size, aspect ratio and binary transitions. In the same competition, but in the other challenge [Shahab et al., 2011], the *ECNU-CCG* method computes an edge map by combining four inidividual edge maps with typical directions (horizontal, vertical, up-right and up-left) and potential text areas are extracted using CC analysis. Finally, an N-level scale space model is constructed and the spatial responses to the Laplacian-of-Gaussian operator are computed. The scale at which the strongest spatial responses are present, indicates the stroke width of the text characters. Therefore, it uses the distribution of the strongest response and the scale where they appearing to identify the candidate text regions.

2.1.3 Texture-based methods

Texture-based approaches typically extract distinctive texture features from regions on the image and these regions are identified as text or non-text by a classifier trained either with machine learning techniques or by heuristics. Generally, a texture-based approach consists of two stages:

1. Text detection to estimate text existing confidence in local image regions by classification.
2. Text localization to cluster local text regions into text blocks, and text verification to filter out non-text regions.

Among texture-based methods, [Chen and Yuille, 2004] is one of the most significant works. It uses a set of informative features based on the intensity, gradient direction and intensity gradient. Weak classifiers, using joint probabilities for feature responses on and off text, are used as input to an AdaBoost cascade classifier. Regions selected by the classifier are clustered into groups according to their location and size. Then, an adaptive binarization algorithm is applied and CCs are extracted. Later, the CCs are grouped into lines followed by an extension algorithm to find missing boundary letters.

[Ye et al., 2007] use a color quantization method to separate text from its background and applies a spatial layout analysis. Generalized Learning Vector Quantization (GLVQ) is used to group pixels of similar color into the same cluster in LUV color space. For text and non-text classification of the candidates, histogram features of wavelet coefficients and color variance are extracted to capture the texture properties of text and fed into a SVM classifier.

[Wang et al., 2009] employ gray-scale contrast, edge orientation histogram and SVM to verify detected text objects. [Hanif and Prevost, 2009] use a small set of heterogeneous features (Mean Difference Feature (MDF), Standard Deviation (SD) and HOG) which are spatially combined to build a large set of features. A neural-network-based localizer learns the localization rules automatically. This system is known as *Text Hunter*. [Tu et al., 2006] calculate the average intensity and statistics of the number of edges from training samples. Then, AdaBoost is used to classify the candidate blocks. Text boundaries are matched with pre-generated deformable templates based on shape context and informative features.

[Minetto et al., 2010] carry out an image segmentation based on a morphological operator called toggle mapping. It produces a set of regions. Those which contain text are discriminated from those that do not by taking into account three families of descriptors (Fourier moments, pseudo-zernike moments and a polar representation) that are fed into a hierarchical SVM classifier. Then an hypothesis validation scheme based on HOG descriptors is applied over each detected window in order to remove false positives. This system has been presented to the ICDAR 2011 robust reading competition with the name of *LIP6-Retin*.

A hybrid approach of texture-based and edge-based features is proposed in [Lee et al., 2010]. It generates text region candidates using two assumptions: homogeneity of text color and distinctiveness between text and background regions. A k-means algorithm is used to separate the image into regions of the same color and the regions are constrained using edges. Then, the candidates are verified using a Markov Random Field representation that models spatial relationships among regions as an undirected graphical model.

Another hybrid approach of CC-based and texture-based features is presented in [Escalera et al., 2009]. This method is based on learning spatial information of gradient-based features and Census Transform images using a cascade of classifiers.

Kim's method, which has competed in the ICDAR 2011 robust reading competition [Shahab et al., 2011], extracts blobs in an image using MSER, and neighboring blobs are merged when their sizes and colors are similar. A cascade classifier is used to discriminate text from non-text regions using gradient features. In the same competition, the *KAIST AIPR* system introduces the concept of superpixel, which is a coherent local region that preserves most of the object boundaries, and the concept of segment, which is an enlarged region that has one or more neighboring superpixels together. This system assumes that text regions can be modeled as segments.

[Phan et al., 2012] propose to use the Gradient Vector Flow (GVF) [Xu and Prince, 1998] to extract both intra-character and inter-character symmetries. Then, horizontally aligned symmetry components are grouped into text lines based on several constraints on size, position and color. Finally, a classifier based on SVM and HOG is used to remove false positives.

Each one of the approaches explained (CC-based, edge-based and texture-based methods) has different advantages and drawbacks. For instance, CC-based and edge-based methods are able to detect text at any scale simultaneously and are not limited to horizontal alignment of the text, but they have problems when the text is in a complex background or in contact with other graphical objects. On the other hand, texture-based methods have difficulty to find accurate boundaries of text areas and need to scan the image at different scales thus leading to higher processing time than CC-based algorithms. An overview of the main characteristics of the above mentioned methods is shown in table 2.1.

2.2 Text recognition

Text recognition aims at identifying the characters and words detected in an image. The performance of the recognition strongly depends on the accuracy of the detection. The clearer and the more precise the extraction of the bounding boxes of the text and the separation into words and single characters is, the more reliable the recognition is. Therefore, most of the work in the area of automatic text recognition is based on developing very accurate text localization methods and then applying commercial OCRs where text areas have been found in the image. However, due to the huge variability of font styles, thickness, colors, texture, resolution or illumination, among other factors, the number of scenarios where commercial OCRs work well is very limited. Some attempts to develop robust character recognition techniques for natural images have been carried out.

Methods for offline recognition of handprinted characters [Plamondon and Srihari, 2000; Pal et al., 2007] successfully tackle the problem of intra-class variation due to

Method	Approach	Features	Classifier	Evaluation
[Yao et al., 2007]	CC	Intensity, Geometry	SVM	ICDAR'03
[Neumann and Matas, 2012]	CC	ER, Geometry	Heuristics	ICDAR'03, ICDAR'11-2
[Merino et al., 2011]	CC	ER, Geometry, Intensity, Texture	Cascade	ICDAR'03
[Epshtein et al., 2010]	CC	Geometry, SWT	Heuristics	ICDAR'03
[Chen et al., 2011]	CC	Geometry, MSER, Gradient, SWT	Heuristics	ICDAR'03
[Pan et al., 2011]	CC+T	HOG, Geometry, Gray-level difference	WaldBoost	ICDAR'03
TH-TextLoc [Karatzas et al., 2011]	CC	Geometry, Shape, Stroke	SVM	ICDAR'11-1, ICDAR'11-2
TDM IACAS [Karatzas et al., 2011]	CC	Gradient, Geometry	SVM	ICDAR'11-1, ICDAR'11-2
OCTYMIST [Karatzas et al., 2011]	CC	Geometry	Heuristics	ICDAR'11-1
[Fehli et al., 2012]	CC	ER, Geometry, Color, SWT	Heuristics, SVM	ICDAR'03, ICDAR'11-2
[Chowdhury et al., 2012]	CC	Geometry, Intensity, Color, SWT	MLP	ICDAR'03
[Li and Lu, 2012]	CC	ER, Geometry, SWT	Heuristics	ICDAR'03, ICDAR'11-2
[Yin et al., 2012]	CC	ER, Geometry, SWT, Color, Gradient	AdaBoost	ICDAR'11-2
[Liu et al., 2012]	CC	Geometry	Heuristics	ICDAR'03
[Zhang and Lai, 2012]	CC	Geometry	Heuristics	ICDAR'03
SASA [Yi and Tian, 2011]	CC+T	Geometry, Gradient, Haar features	AdaBoost	ICDAR'11-1, ICDAR'11-2
[Liu et al., 2005]	E	Geometry, Edge direction	Heuristics	Own dataset
[Shivakumara et al., 2008]	E	AF, MF, Edge strength	Heuristics	ICDAR'03
[Liu and Sarkar, 2008]	E	Gray-level, Shape	Heuristics	ICDAR'03
[Bai et al., 2008]	E	Harris corners, Geometry, Color	Heuristics	Own dataset
[Zhang and Kasturi, 2010]	E	HOG, Geometry, Edge intensity	Heuristics	ICDAR'03
Textorter [Karatzas et al., 2011]	E	Geometry	Heuristics	ICDAR'11-1
ECNU-CCG [Shahab et al., 2011]	E+CC	LoG, Geometry	Heuristics	ICDAR'11-2
[Chen and Yuille, 2004]	T	Intensity, Gradient direction, Gradient intensity	AdaBoost	Own dataset
[Ye et al., 2007]	T	Color, Geometry, Wavelet coefficients	SVM	Own dataset
[Wang et al., 2009]	T	Contrast, Edge direction	SVM	Own dataset
Text Hunter [Hanif and Prevost, 2009]	T	MDF, SD, HOG	Neural Network	ICDAR'03, ICDAR'11-1, ICDAR'11-2
[Tu et al., 2006]	T	Intensity, Geometry	AdaBoost	Own dataset
LIP6-Retin [Minetto et al., 2010]	T	Fourier moments, Zernike moments, HOG	SVM	ICDAR'03, ICDAR'11-2
[Lee et al., 2010]	E+T	Color, Geometry	MRF	ICDAR'03
[Escalera et al., 2009]	CC+T	Gradient, Census Transform, Geometry	Cascade	CoverDB
Kim's method [Shahab et al., 2011]	T	Color, Geometry	Cascade	ICDAR'11-2
KAIST AIPR [Shahab et al., 2011]	T	Color, Geometry	Heuristics	ICDAR'11-2
[Phan et al., 2012]	T	Symmetry, Color, Geometry, Gradient direction	SVM	ICDAR'03

CC: Connected component-based method. **E**: Edge-based method. **T**: Texture-based method. **MSER**: Maximally Stable Extremal Regions. **ER**: Extremal Regions. **SWT**: Stroke Width Transform. **MLP**: Multi-layer Perceptron. **AF**: Arithmetic mean Filter. **MF**: Median Filter. **HOG**: Histogram of Oriented Gradients. **MDF**: Mean Difference Feature. **SD**: Standard Deviation. **MRF**: Markov Random Field. **ICDAR'03**: ICDAR 2003 Robust Reading Competition dataset. **ICDAR'11-1**: ICDAR 2011 Robust Reading Competition Challenge 1 dataset. **ICDAR'11-2**: ICDAR 2011 Robust Reading Competition Challenge 2 dataset.

Table 2.1: A comparison of text detection methods

different writing styles. However, these approaches typically consider only a limited number of appearance classes, not dealing with variations in color and texture, both in foreground and background.

For natural scenes, some works integrate text detection and recognition in a single framework. The problem of this kind of works is that they are not able to detect characters in the image that are difficult to be segmented, such as small letters in low resolution images or characters written with fonts that emulate handwriting. For instance, [Tu et al., 2006] model text characters by 62 deformable templates corresponding to the 10 digits and the 26 letters in both upper and lower cases. Shape classification is carried out using nearest neighbor matching. [Jin and Geman, 2006] assume that letters are built with *terminal bricks* or blobs, which are semantic variables, like edges, strokes, junctions and shapes, obtained using local image filters. Each letter is modeled as a set of bricks. Nevertheless, this method needs to have a huge a priori knowledge of the scenario. Therefore, it works well for well-defined applications, like license plate recognition, but the performance decreases in unrestricted scenarios. [Weinman and Learned-Miller, 2006] propose a method that computes image features using Gabor filters, which decompose geometry into local orientation and scale. This approach assumes certain a priori knowledge of the language with a bigram model, which takes into account the likelihood of appearing two certain letters together, and letter case to improve recognition accuracy in context as English language rarely switches case in the middle of a word. Character classification is carried out using a measure of dissimilarity. This method needs to have a huge in-depth a priori knowledge of the language and it uses a bigram model that only works to model pairs of letters but not whole words, thus it is able to correct typos at syllable level but not at word level.

Recognition based on classifying raw images is explored only for digits recognition by [LeCun et al., 1998; Zhang et al., 2006] on the MNIST and USPS datasets. Another approach is based on modeling the recognition as a shape matching problem ([Belongie et al., 2002]): several shape descriptors are detected and extracted and point-by-point matching is computed between pairs of images. A comparison of this descriptor with other local features (Geometric Blur [Berg et al., 2005], Scale Invariant Feature Transform [Lowe, 1999], Spin image [Lazebnik et al., 2005], Maximum Response of filters [Varma and Zisserman, 2002] and Patch descriptor [Varma and Zisserman, 2003]) is presented in [de Campos et al., 2009]. Using a bag-of-visual-words representation, the authors achieve a performance far superior to commercial OCR systems for images of street scenes containing English and Kannada characters. They show that Geometric Blur and Shape Context achieve the best performance in their experiments. However, the classification time for these descriptors is very low, as achieving a good performance requires to have a huge training set in order to deal with variations in styles, fonts, colors and deformations.

In [Neumann and Matas, 2010], character recognition is carried out using contour-based features and a multi-class SVM classifier, which is trained using synthetic data. The method takes into account multiple hypotheses in text localization and recognition stages and selects the best one in the final stage using a unigram language model. However, this unigram model is used only for differentiating between the upper-case and lower-case variants of certain letters, such as “C” and “c” or “P” and “p”, which are ambiguous.

Grayscale features and Convolutional Neural Networks (CNNs) are proposed in [Zhu et al., 2012] to achieve character recognition with a very good performance, but they do not apply any method to correct typos. In addition, this approach requires applying an image enhancement method since it works with gray-level values. Grayscale features are also used in [Coates et al., 2011], but character images are divided into 8-by-8 pixels sub-patches in order to have spatial information, and a simple binary classifier is used to decide if each of these sub-patches correspond to text or not. Like the previous work, it does not apply any error correction method.

A new feature based on local symmetry and orientation is proposed in [Newell and Griffin, 2011] for character recognition in natural images. This method has a series of parameters that has to be tuned heuristically. In addition, this method neither does typo correction. [Chan and Pun, 2011] propose to use grayscale features and Fisher Component Analysis (FCA) to reduce the dimensionality of the data by finding the most distinctive characteristics between classes. However, the results obtained with this method are poorer than with other methods of the state of the art, thus meaning that this approach is not the most suitable. Again, this method does not apply any typo correction method.

A gradient-based feature computed on the gray-scale image is proposed in [Liu and Ding, 2005] to recognize single characters. Horizontal and vertical gradients are extracted at each image pixel using Sobel operator. Then, L directions with an equal interval $2\pi/L$ are defined and the gradient vector is decomposed into its two nearest directions in a parallelogram manner. In this way, a L-dimensional gradient code is obtained at each image pixel. In order to have spatial information, these codes are obtained within sub-blocks. The classification is carried out using a Modified Quadratic Discriminant Function (MQDF). However, results with this method for English alphabet are unavailable as it has been tested with MNIST dataset, which is composed only with digits, and ETL9B and HCL2000, which consist of images of only Chinese characters.

An overview of the main characteristics of the above mentioned methods is shown in table 2.2.

2.3 Discussion

Previous sections have introduced a number of published methods for text detection and recognition. Even though there are many more, it has been shown that the diversity of the approaches is high. Some of the works previously presented have served as an inspiration to the work proposed in this thesis. They will be discussed in this section.

Performing a reliable text detection highly depends on achieving an accurate image segmentation. Different segmentation methods have been tested in the literature. However, image segmentation applied to text detection in natural images seems to have peaked with the introduction of MSERs, which are able to extract text regions very well even in complex backgrounds, as [Chen et al., 2011; Neumann and Matas, 2012; Merino et al., 2011; Fehli et al., 2012; Li and Lu, 2012; Yin et al., 2012] have shown. However, MSER has proved to be sensitive to image blur. This is an important drawback, especially when trying to segment small letters in images of limited resolution. [Chen et al., 2011] have addressed this problem combining MSER with a Canny edge detector. However, accurate edge extraction can be difficult when the text is embedded in a complex background, it is in contact with other objects or the illumination of the text in the image is not homogeneous and some parts of a same word are differently illuminated than others. In this case, a global thresholding method does not perform well and it has been proved to be more reliable to use locally adaptive thresholding methods like the one proposed by [Yao et al., 2007]. On the other hand, the problem of using a local segmentation method is that it generates a huge amount of false positives.

Once an image has been segmented and text candidates have been generated, different

Method	Features	Classifier	Evaluation	Performs typo correction
[de Campos et al., 2009]	SC, GB, SIFT, SI, MR8, PCH	NN, SVM, MKL	Own dataset	No
[Tu et al., 2006]	Shape deformable templates	NN+Bayes	Own dataset	No
[Jin and Geman, 2006]	Edges, strokes, junctions, shapes	Bayes	Own dataset	No
[Weinman and Learned-Miller, 2006]	Gabor filters	Distance	Own dataset	At syllable level
[Neumann and Matas, 2010]	Contour	SVM	ICDAR'03	At character level
[Zhu et al., 2012]	Grayscale	CNN	ICDAR'03	No
[Coates et al., 2011]	Grayscale	SVM	ICDAR'03	No
[Newell and Griffin, 2011]	Local symmetry and orientation	NN	chars74k, ICDAR'03	No
[Chan and Pun, 2011]	Grayscale	NN	ICDAR'03	No
[Liu and Ding, 2005]	Gradient direction	MQDF	MNIST, ETL9B, HCL2000	No

SC: Shape Context. **GB**: Geometric Blur. **SIFT**: Scale Invariant Feature Transform. **SI**: Spin Image. **MR8**: Maximum Response of filters. **PCH**: Patch descriptor. **NN**: Nearest Neighbor. **SVM**: Support Vector Machine. **MKL**: Multiple Kernel Learning. **CNN**: Convolutional Neural Network. **MQDF**: Modified Quadratic Discriminant Function.

Table 2.2: A comparison of single character recognition methods

features have been tested in order to distinguish between text and non-text objects. However, the state of the art shows that, at the end of the day, it is inevitable to use some geometric constraints, such as aspect ratio of the letters or alignment in one direction. In this sense, [Epshtein et al., 2010] have shown that letters are made of strokes of typically the same width. They have proposed the Stroke Width Transform (SWT), which is a local image operator that computes per pixel the width of the most likely stroke containing the pixel. [Chen et al., 2011] have proposed an alternative and easiest way to compute the SWT using the Distance Transform.

Many works have also shown that text regions are well-defined high-density areas in the image with a large intensity of the gradient. This fact has been initially shown by [Chen and Yuille, 2004] and it has been longer exploited by other authors such as [Escalera et al., 2009; Hanif and Prevost, 2009; Zhang and Kasturi, 2010; Pan et al., 2011; Phan et al., 2012].

Different approaches have been used to discard non-text objects. Typically, some threshold values are applied on the features extracted for the text candidates. These values can be obtained either heuristically, as in [Neumann and Matas, 2012; Epshtein et al., 2010; Chen et al., 2011], or by means of machine learning algorithms, such as SVM ([Ye et al., 2007; Wang et al., 2009; Minetto et al., 2010]), neural networks ([Hanif and Prevost, 2009; Chowdhury et al., 2012]) or AdaBoost ([Yi and Tian, 2011; Tu et al., 2006; Yin et al., 2012]).

Text recognition has been less in-depth researched, as most of the works have focused their efforts on improving text localization. Usually, the use of commercial OCRs has been proposed, as they seem to work well once a text region has been accurately detected and the font, style, color and layout do not differ very much from those of typical black-and-white scanned documents. However, the performance of these systems decreases when these conditions are not fulfilled, which typically occurs in most of the natural images. Some of the solutions proposed for single character recognition consist of using features based on the fact that characters are closed regions. [de Campos et al., 2009] have shown that Geometric Blur and Shape Context perform well, but they require to have a huge set of templates in order to carry out a reliable matching. A similar approach based on taking into account the orientation of the boundary pixels (direction of the gradient) of the characters is proposed by [Neumann and Matas, 2012; Newell and Griffin, 2011; Liu and Ding, 2005]. However, most of works do not apply any method to correct erroneously recognized single characters, except for [Neumann and Matas, 2012; Weinman and Learned-Miller, 2006]. The first one uses a language model to distinguish between some letters that, in principle, are virtually impossible to be differentiated in their upper-case and lower-case variants without knowing the heights of other letters in the same context. The second method applies a bigram model that corrects typos at syllable level, but not at word level.

Most of the methods presented in this chapter have tested their performance on a series of public datasets that were released for two conferences (ICDAR 2003 and ICDAR 2011). These datasets have proved to be very challenging, as they contain images that have been obtained in different outdoor and indoor scenarios with a wide variety of text appearance in terms of font, thickness, color, size, texture, layout, lighting conditions and occlusions. Therefore, these datasets are considered as a solid way to validate the robustness and performance of any text detection method for natural images.

2.4 Aim of this thesis

After the review of the state of the art and taking into account the opened points detected on it, the aims of this thesis are as follows:

1. To develop an automatic text detection and recognition system for natural images taken with image-capturing devices without any kind of restriction, except for the fact that the images must be taken at daytime and the text must be horizontally aligned. The method should achieve or improve state-of-the-art performance, both in terms of localization and recognition, trying to address some of the problems that other works have not been able to solve. In terms of text localization, the aim is to improve the accuracy of the detection and to reduce the number of false positives. In terms of text recognition, the use of a language model at word level seems to be an interesting challenge that other methods have not tried to apply, as it will serve as a solution to correct the errors made during the single character recognition step.

2. To research the adequacy of techniques like MSER and HOG, which have been considered of interest a priori based on the research carried out by some of the most significant works of the state of the art.

3. To assess the performance of the proposed system using the same datasets used by other authors in order to have a reliable comparison.

4. To develop an application useful in the field of Intelligent Transportation Systems (ITS) based on the previous research. The application will be addressed at detecting and recognizing the information contained in traffic panels on images extracted from the Street View service by Google.

Chapter 3

Text Detection in Natural Images

The algorithm proposed in this thesis can be divided into two main independent parts. The first one aims at detecting if any text is present in an image and to locate it, while the second part deals with the problem of recognizing the text detected in the first stage. Figure 3.1 shows the flowchart of the proposed framework, which is the typical used in the state of the art for CC-based methods, although some modifications and contributions have been developed, which will be in-depth explained in the corresponding chapters and sections. The text detection and location algorithm consists of four main blocks. The first one extracts candidates to be text objects after segmenting the image into regions. The second one analyses the regions or components extracted using several unary component features, which characterize single component's geometric and textural properties, in order to discard those regions that do not correspond with text and to validate those that do. Later, the validated regions are put together into text lines using different binary component features, which characterize the spatial relationship and geometric and textural similarity between neighboring components. Then, text lines are classified into text or non-text regions using machine learning techniques in order to discard false positives. For this purpose, although our method is based on CC analysis, the idea of classifying text lines comes from texture-based methods, thus the proposed framework is a combination of CC-based and texture-based algorithms. Finally, text lines are separated into words. The resulting words are the input to the second main block of the algorithm, the text recognition step. Typically, commercial OCRs are used for this purpose, but they do not achieve a good performance when recognizing text in natural images due to the complexity of the scenarios. We propose to apply in first place a recognition method to identify single characters. Then, we correct the errors that may have been generated in the character recognition step with the help of a language model which reduces the possible outputs to a search space defined by a set of all possible candidate solutions.

This chapter describes the text detection method proposed in this thesis to detect if any text is present in an image and to locate it. In first place, it is explained in section 3.1 the study that has been carried out to obtain the set of the most suitable unary component features to describe text. Then, section 3.2 shows the proposed text detection and location algorithm. Finally, the chapter ends with experimental results obtained with several datasets and main conclusions are drawn. On the other hand, the text recognition method will be detailed in the following chapter.

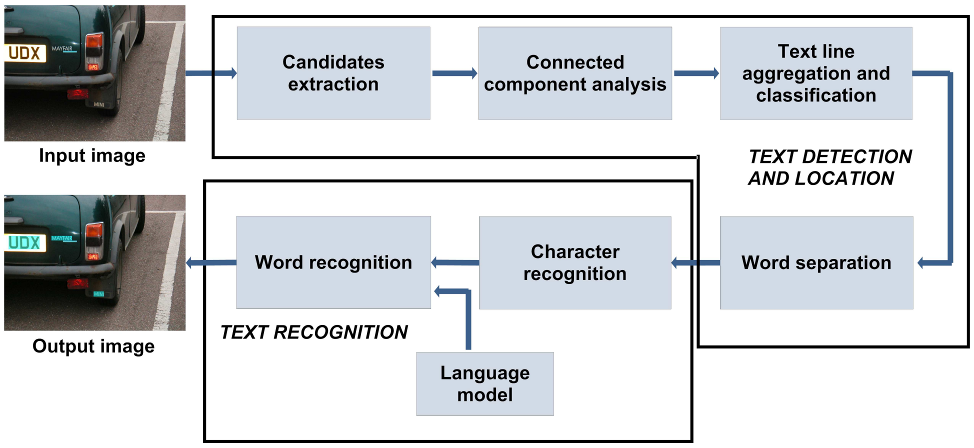


Figure 3.1: Flowchart of the proposed framework

3.1 Text features analysis

Most of the state-of-the-art CC-based approaches work in a similar way. They segment the input image into different regions or connected components and these are filtered using features that try to describe geometric and textural properties of character CCs differentiating themselves from non-character CCs. [Yao et al., 2007] propose to use the features summarized in table 3.1.

Height	Width
Area	Perimeter
Occupy rate	Aspect ratio
Compactness	Contour roughness
Stroke width size ratio	Max stroke width size ratio
Stroke width variance ratio	Stroke density

Table 3.1: Features used by [Yao et al., 2007].

Area, width, height and perimeter are directly obtained from the bounding boxes of the CCs, while the rest of the features are detailed in (3.1)-(3.8).

$$Occupy\ rate = \frac{area}{height * width} \tag{3.1}$$

$$Aspect\ ratio = \frac{max(width, height)}{min(width, height)} \tag{3.2}$$

$$Compactness = \frac{area}{perimeter * perimeter} \tag{3.3}$$

$$Contour\ roughness = \frac{Number\ of\ Rough\ Pixels}{height + width} \tag{3.4}$$

$$Stroke\ width\ size\ ratio = \frac{Stroke\ width}{max(height, width)} \tag{3.5}$$

$$Max\ stroke\ width\ size\ ratio = \frac{Max\ stroke\ width}{max(height, width)} \tag{3.6}$$

$$Stroke\ width\ variance\ ratio = \frac{Stroke\ width\ variance}{Stroke\ width} \tag{3.7}$$

$$Stroke\ density = \frac{Number\ of\ pixels\ in\ the\ median\ axis}{height + width} \tag{3.8}$$

Contour roughness tries to measure the regularity of the contours of the CCs. Character CCs are supposed to have a higher regularity, while non-character CCs lower. Contour roughness is computed using 180 3-by-3 convolution masks at the contour pixels. Two examples of these templates are shown in figure 3.2 (to the best of our knowledge, the rest of the templates have not been published). If the contour pixel of a CC matches a rough pixel template, it is considered as a rough pixel. The contour roughness is computed from the number of rough pixels.

1	0	0
0	1	0
0	0	0

(a)

1	0	0
1	1	0
1	0	0

(b)

Figure 3.2: Two examples of the rough pixel templates used by [Yao et al., 2007].

On the other hand, the stroke width seems to be one of the most distinctive features of character CCs against non-character CCs. This idea has been exploited not only by [Yao et al., 2007], but also by [Epshtein et al., 2010] and [Chen et al., 2011]. [Yao et al., 2007] compute the stroke width on each pixel after applying a thinning algorithm to obtain the median axis of the CCs and counting the number of iterations to verify the pixel on the median axis. The stroke width is the average of the stroke width of all the pixels, while the variance is the stroke width variance of all the pixels.

The main drawback of the proposal by [Yao et al., 2007] is that the upper and lower thresholds used to classify each of the above features into character or non-character are considered to be the maximum and the minimum of every feature of character CCs in the train image database. Therefore, these thresholds are not the optimum.

A smaller but very similar set of features is proposed by [Epshtein et al., 2010], as shown in table 3.2. The last feature is computed as the ratio between the diameter of the CC and its median stroke width. The thresholds of all these features are learned by optimization on a train dataset.

Stroke width	Stroke width variance
Aspect ratio	Height
Diameter-Median ratio	

Table 3.2: Features used by [Epshtein et al., 2010].

Similarly, [Chen et al., 2011] propose the features shown in table 3.3. The thresholds for all these features are learned heuristically.

[Neumann and Matas, 2010] propose the features shown in table 3.4. They are learned using a standard SVM classifier with Radial Basis Function (RBF) kernel. The classifier

Height	Aspect ratio
Number of holes	Stroke width variance

Table 3.3: Features used by [Chen et al., 2011].

was trained on a set of 1227 characters and 1396 non-characters obtained from real-world images. The training set is relatively small and it does not contain a representative set of all possible fonts, scripts and even characters.

Aspect ratio	Relative segment height
Compactness	Number of holes
Convex hull area to surface ratio	Character color consistency
Background color consistency	Skeleton length to perimeter ratio

Table 3.4: Features used by [Neumann and Matas, 2010].

In order to obtain a set of distinctive features capable of distinguishing character objects from non-character objects, we have made an analysis of certain unary component features under the ICDAR 2003 Reading Competition dataset. The four state-of-the-art approaches above referenced are taken into account as seminal works for the study explained below. The objective is to discover the optimum set of features that describe the geometric and textural properties of character CCs and allow to discard non-character CCs. The ICDAR 2003 dataset contains a total of 509 realistic images with complex background, captured in a wide variety of situations, with different cameras, at different resolutions and under different lighting conditions. The dataset is divided in two sections: a training set that contains 258 images and a test set which have 251 images. We have worked with the training set for the analysis that we present below.

There are 6185 single characters in the training set, but we have analysed only 5438 characters, because not all samples are valid for this goal due to different reasons, such as too small size or partial occlusions. The training set includes uppercase and lowercase letters as well as digits. Using the segmentation method that will be explained in section 3.2, we have binarised and manually labelled every sample and have computed several geometric and textural features. In figure 3.3, we show some examples of the binarised characters from which the features explained below have been computed. On the other hand, a total of 5978 non-character components have been extracted from the same train database using the same segmentation method. Some examples are shown in figure 3.4.

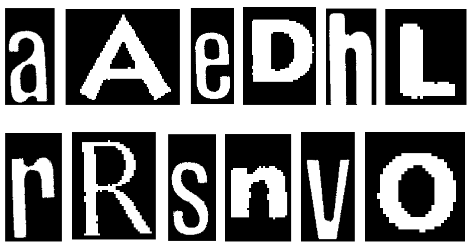


Figure 3.3: Some positive training samples.

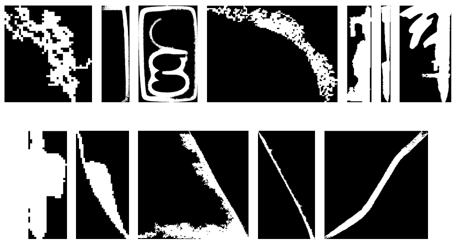


Figure 3.4: Some negative training samples.

Taking some of the features used by the works explained above as a starting point and adding new features that we consider of interest, it has been seen that the features shown in table 3.5 are very adequate to describe single character components and to discard non-character objects. We draw this conclusion from the fact that, as it will be shown later in this section, the probability density functions of these features follow a Gaussian distribution for character CCs but they do not follow any known distribution for non-character CCs.

Occupy rate, aspect ratio, compactness, stroke width size ratio, maximum stroke width size ratio and stroke width variance ratio are computed as in (3.1), (3.2), (3.3), (3.5), (3.6) and (3.7) respectively. The height refers to the height in pixels of the bounding box that contains the component. The number of holes is the number of internal regions of the object that are completely surrounding by pixels corresponding to the component. On the other hand, the new two features proposed in this thesis, the occupy rate convex area and the solidity, are computed using (3.9) and (3.10).

Height	Aspect ratio
Number of holes	Occupy rate
Compactness	Solidity
Occupy rate convex area	Stroke width size ratio
Max stroke width size ratio	Stroke width variance ratio

Table 3.5: Features proposed in this thesis.

$$Solidity = \frac{area}{convex\ area} \tag{3.9}$$

$$Occupy\ rate\ convex\ area = \frac{convex\ area}{height * width} \tag{3.10}$$

The convex area is the area of the convex hull, which is the smallest convex polygon that contais the region, as it is shown in figure 3.5.

Unlike [Yao et al., 2007], the stroke width and its derivative features are not computed using a thinning algorithm, but using the Stroke Width Transform, a local image operator proposed by [Epshtein et al., 2010] that computes per pixel the width of the most likely stroke containing the pixel. Specifically, we use the way of computing the SWT in [Chen et al., 2011] from the Distance Transform instead of calculating gradient directions, which cannot be well determined when the text is badly contrasted. The SWT is a more reliable way of computing the stroke width, as it does not depend on any thinning algorithm that may have undesirable behaviour under difficult lighting conditions. For this reason,

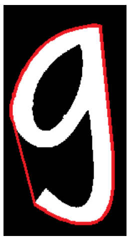


Figure 3.5: Convex hull (red)

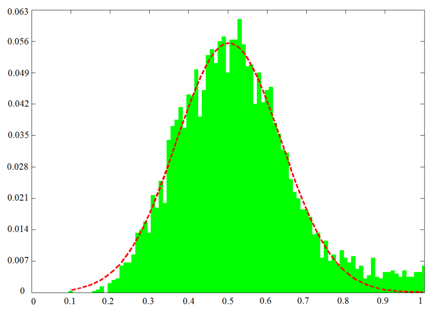
the stroke density feature proposed by [Yao et al., 2007], which is specified in (3.8), has been discarded to be used. Similarly, the contour roughness has not been possible to be studied, because the convolutional masks that are needed to compute this feature are not publicly available. In the same way, most of the features proposed by [Neumann and Matas, 2010], which were shown in table 3.4, such as the background color consistency, the relative segment height, the character color consistency and the skeleton length to perimeter ratio, have not been possible to be analysed, because they have not been clearly defined by the authors.

The reason for considering the chosen set of features as very adequate is that, if we represent the normalized histogram of each of the features shown in table 3.5, except for the height and the number of holes, which only depend on the font size and the letter respectively, and we compute their probability density functions, we see that they follow a unimodal Gaussian distribution, or half an unimodal Gaussian distribution in case of the aspect ratio, as it is displayed in figure 3.6. However, the same approximation cannot be made for non-character components, as it is shown in figure 3.7. The values of mean (μ) and standard deviation (σ) for each of these features are shown in table 3.6.

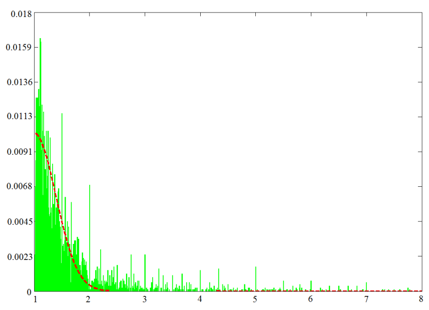
Feature	Mean (μ)	Std deviation (σ)
Occupy rate	0.5227	0.1485
Aspect ratio	1.0	1.1357
Compactness	0.0257	0.0127
Solidity	0.6253	0.1514
Occupy rate convex area	0.8342	0.0993
Stroke width size ratio	0.1081	0.0406
Max stroke width size ratio	0.1356	0.0485
Stroke width variance ratio	0.2427	0.3005

Table 3.6: Mean and standard deviation of each feature.

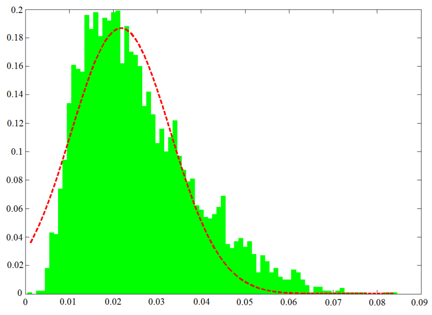
In the next section, we will explain the proposed text location method to detect if any text is present in the image. The rule to decide if a CC corresponds to a character or a non-character is based on the idea that we are showing in this section. If the values of all the features of the CC lie within the range $(\mu - 2 \cdot \sigma, \mu + 2 \cdot \sigma)$, then the object is classify as character. The reason for choosing these upper and lower thresholds $\mu - 2 \cdot \sigma$ and $\mu + 2 \cdot \sigma$ comes from one of the properties of a Gaussian distribution, which states that around 95.44% of the elements of the distribution lie within 2 standard deviations of the mean. However, due to the fact that the standard deviations for the aspect ratio and the stroke width variance ratio are much higher than for the rest of the features, the thresholds for these two features have been set to $\mu - \sigma$ and $\mu + \sigma$, in order to make these



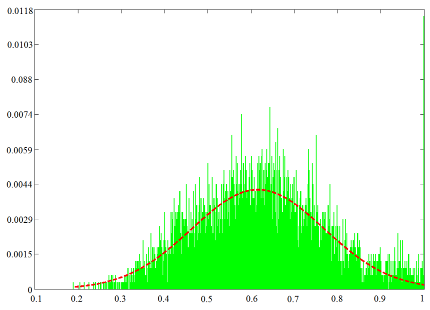
(a) Occupy rate



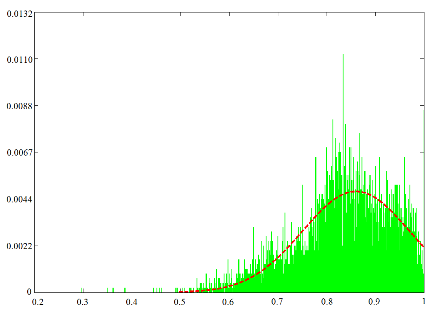
(b) Aspect ratio



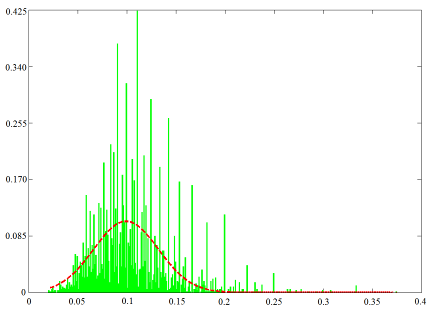
(c) Compactness



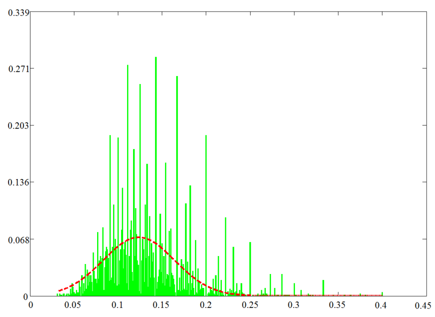
(d) Solidity



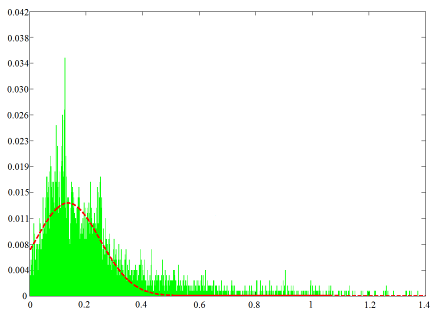
(e) Occupy rate convex area



(f) Stroke width size ratio

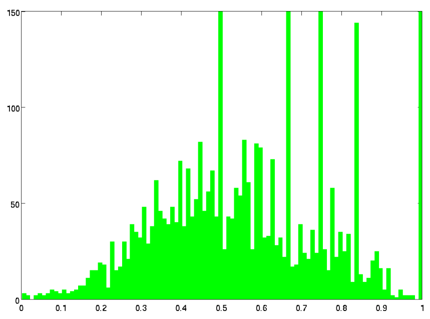


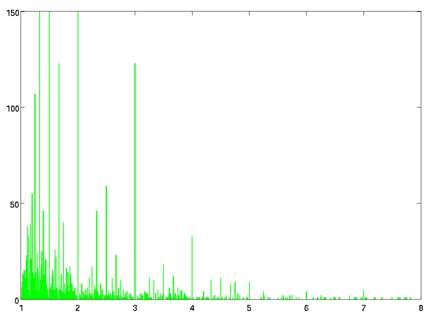
(g) Max stroke width size ratio

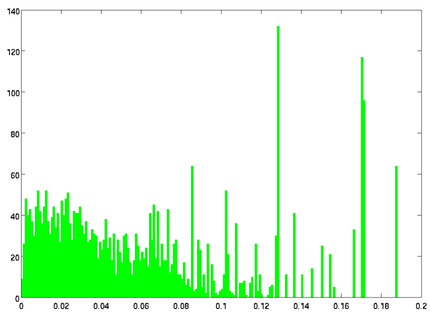


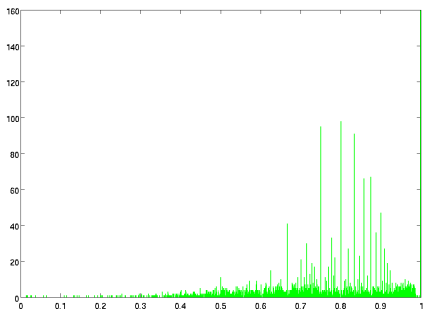
(h) Stroke width variance ratio

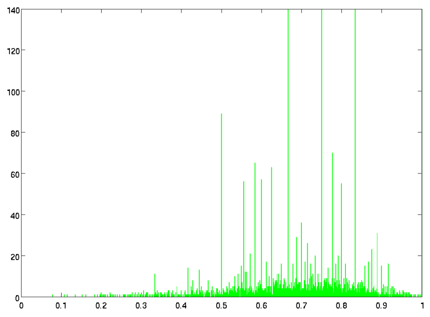
Figure 3.6: Normalized histograms of features vs approximated Gaussian functions for character components on ICDAR'03 training set.

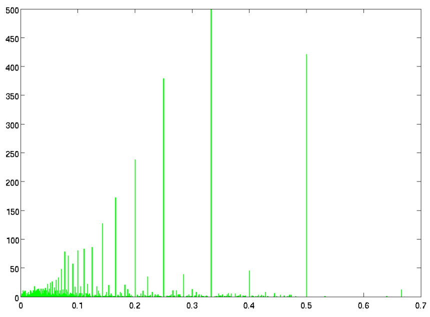

(a) Occupy rate

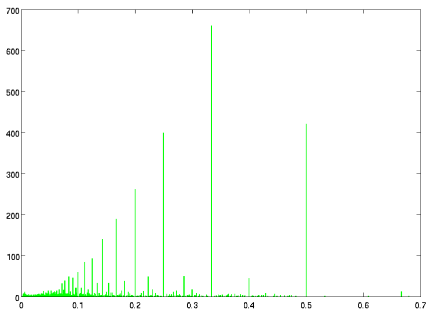

(b) Aspect ratio


(c) Compactness


(d) Solidity


(e) Occupy rate convex area


(f) Stroke width size ratio


(g) Max stroke width size ratio

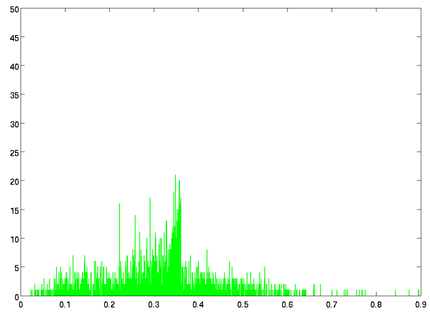

(h) Stroke width variance ratio

Figure 3.7: Histograms of features for non-character components on ICDAR'03 training set.

features more restrictive than choosing the whole range $(\mu - 2 \cdot \sigma, \mu + 2 \cdot \sigma)$. Those values imply that only 68.27% of the elements of the Gaussian distribution are within 1 standard deviation of the mean.

This classification decision can be considered very restrictive. Our aim is to classify as text what is really text. In other words, we want to minimize the number of false positives. However, as it will be explained in the following section, we propose to apply later a restoration method in order to bring back the erroneously rejected character CCs using some attributes related to proximity and similarity to the initially accepted character CCs.

Table 3.7 shows the upper and lower thresholds for each of the features, including the height in pixels and the number of holes. We do not want the algorithm to detect letters smaller than 10 pixels, so the lower threshold of the height is set to this value. On the other hand, the upper threshold for the number of holes is 2, because the maximum number of holes that a letter can have is 2, as it is the case of ‘B’ and ‘8’. For computing the number of holes for each component, only those holes with a certain minimum size respect to the size of the candidate are taken into account, so small holes due to noise generated at the binarization stage are not considered. By definition, the aspect ratio cannot be lower than 1.0. Similarly, the stroke width variance ratio is a positive value, so the lower threshold for this feature is set to 0.0 instead of -0.0578, which would be the mathematical corresponding value if we apply the formula $\mu - \sigma$.

Feature	Lower threshold	Upper threshold
Height	10	*Inf*
Number of holes	0	2
Occupy rate	0.23	0.82
Aspect ratio	1	2
Compactness	0	0.05
Solidity	0.32	0.93
Occupy rate convex area	0.63	1.03
Stroke width size ratio	0.027	0.19
Max stroke width size ratio	0.04	0.23
Stroke width variance ratio	0	0.5

Table 3.7: Lower and upper thresholds for each feature.

The previous study has been done for the whole training set of the ICDAR 2003 dataset, which includes all the uppercase letters ‘A’-‘Z’, all the lowercase letters ‘a’-‘z’ and all the digits from ‘0’ to ‘9’. In total, there are 62 different classes. We have carried out the same analysis for each class separatedly. For many classes, we only have a few samples, thus the following conclusions should be confirmed using more data. We have realised that the probability density functions of the features for each separated class can be also approximated by Gaussian functions. The values of the standard deviation for each class and for each feature are, in general, lower than the values shown in table 3.6 for the general case that includes all the classes, but not as low as it could be expected. The values of the means are very similar to the general case. It means that the variability of a single class is almost as large as the variability of all the classes together. Some examples, for some of those letters for which we have more samples, are displayed in table 3.8. We show the relative deviations of the mean and the standard deviation of the Gaussian distibution of each separated class, respect to the values of the Gaussian distibution of the general case. These relative deviations are computed using (3.11) and (3.12), where $\mu_{all\ classes}$ and $\sigma_{all\ classes}$ refer to the parameters displayed in table 3.6, and μ_{class} and σ_{class} are the mean and standard deviation of each Gaussian distribution that models each separated class.

$$Rd_{\mu} = \frac{|\mu_{all\ classes} - \mu_{class}|}{\mu_{all\ classes}} \tag{3.11}$$

$$Rd_{\sigma} = \frac{|\sigma_{all\ classes} - \sigma_{class}|}{\sigma_{all\ classes}} \tag{3.12}$$

For the classes shown in table 3.8, it can be seen that the relative deviation of the mean is very low, less than 10% in general, except for the compactness, the aspect ratio and the stroke width variance ratio, whose relative deviations achieve 40% for some letters and even 74% for the last feature. Similarly, the relative variations of the standard deviation are also very low, less than 10% in general, except for the compactness, the occupy rate convex area, the aspect ratio and the stroke width variance ratio, whose relative variations rise to 50% and, even, 60% and 80%. The initial conclusion that can be drawn from this is that the compactness, the occupy rate convex area, the aspect ratio and the stroke width variance ratio seem to be the least discriminative features. On the other hand, the stroke width size ratio and the maximum stroke width size ratio seem to be the most discriminative features, as there is hardly no difference between the classes.

However, these are only a few examples and we want to see if the same behaviour happens to the rest of the classes. For this purpose, instead of showing a table that includes the values for each class separatedly, which would be very large as there are 62 different classes, we have computed the averages of all the relative deviations of the mean and the standard deviation of each separated class. The averages are weighted to the number of samples N_i that there are per class, as shown in (3.13) and (3.14), where C is the number of classes (62 in this case).

$$Wr_{\mu} = \frac{\sum_{i=1}^{C} N_i \cdot Rd_{\mu_i}}{\sum_{i=1}^{C} N_i} \tag{3.13}$$

$$Wr_{\sigma} = \frac{\sum_{i=1}^{C} N_i \cdot Rd_{\sigma_i}}{\sum_{i=1}^{C} N_i} \tag{3.14}$$

The last column in table 3.9 shows the searched values, which confirm the conclusion drawn in the previous lines. The compactness, the occupy rate convex area, the aspect ratio and the stroke width variance ratio are the least discriminative features. In order to see if there is any difference in the degree of variability depending on if the class corresponds to an uppercase letter, a lowercase letter or a digit, we have broken down the weighted means into these three kinds of classes, and the results are displayed in the third to fifth columns in table 3.9. From them, we can conclude that the uppercase letters seem to be less variable than the lowercase letters, as the weighted means are in general lower,

Feature		'E'	'N'	'a'	'e'	'2'
Occupy rate	Mean	0.31%	0.33%	8.95%	2.43%	1.99%
	Std deviation	8.01%	3.03%	23.30%	30.03%	34.95%
Compactness	Mean	38.52%	41.24%	7.78%	0%	26.85%
	Std deviation	41.73%	48.03%	11.02%	25.20%	61.42%
Solidity	Mean	10.31%	10.41%	5.29%	3.79%	8.44%
	Std deviation	6.47%	6.87%	21.46%	20.48%	31.90%
Occupy rate convex area	Mean	12.06%	11.21%	3.42%	1.16%	7.18%
	Std deviation	43.91%	26.89%	53.07%	58.11%	65.06%
Stroke width size ratio	Mean	6.75%	0.37%	6.94%	1.02%	6.85%
	Std deviation	8.87%	3.94%	11.08%	9.61%	26.60%
Max stroke width size ratio	Mean	6.42%	11.36%	14.16%	10.84%	4.87%
	Std deviation	3.09%	2.68%	6.19%	7.84%	20.62%
Aspect ratio	Mean	4.60%	23.10%	26.27%	26.27%	13.43%
	Std deviation	60.79%	71.83%	80.70%	77.79%	78.72%
Stroke width variance ratio	Mean	38.50%	74.28%	20.86%	6.92%	10.02%
	Std deviation	40.73%	5.19%	21.83%	27.22%	30.25%

Table 3.8: Relative deviation of the mean and standard deviation of each feature per class respect to the values for all the classes.

Feature		Uppercase letters	Lowercase letters	Digits	All classes
Occupy rate	Mean	9.65%	7.47%	7.61%	8.49%
	Std deviation	19.02%	28.83%	38.53%	24.56%
Compactness	Mean	30.11%	23.98%	33.57%	27.12%
	Std deviation	33.00%	27.62%	44.59%	30.64%
Solidity	Mean	9.19%	5.92%	11.04%	7.60%
	Std deviation	13.18%	22.70%	35.02%	18.64%
Occupy rate convex area	Mean	9.70%	5.63%	7.12%	7.57%
	Std deviation	44.68%	47.74%	57.89%	46.62%
Stroke width size ratio	Mean	7.55%	10.67%	10.55%	9.21%
	Std deviation	13.54%	9.23%	32.02%	11.93%
Max stroke width size ratio	Mean	7.89%	12.32%	8.55%	10.15%
	Std deviation	12.04%	10.27%	26.22%	11.58%
Aspect ratio	Mean	14.85%	29.31%	21.84%	22.36%
	Std deviation	63.47%	73.95%	65.13%	68.81%
Stroke width variance ratio	Mean	23.35%	21.33%	33.34%	22.63%
	Std deviation	25.30%	36.77%	36.09%	31.41%

Table 3.9: Weighted mean of the relative deviations of the mean and standard deviation of each feature for the uppercase letters, the lowercase letters, the digits and for all the classes, respect to the values that include all the classes.

and at the same time, the variability of the digits is bigger than the variability of the rest of cases.

The study explained in this section has been done using the training set of the ICDAR 2003 dataset. The experimental results at the end of this section will show that the proposed method achieves state-of-the-art performance for other datasets without necessity of re-training. Therefore, the conclusion that can be drawn from this section is that the proposed set of features models character CCs very well, so it is not necessary to model again the features for other datasets and the models here obtained can be generalized.

3.2 Text location

The flowchart of the proposed text location algorithm is shown in figure 3.8. Initially, letter candidates are found with a segmentation method based on MSER and a locally adaptive thresholding method. Then, the resulting candidates are filtered using certain constraints based on the study shown in the previous section. Character candidates are grouped into lines and each line is classified into text or non-text using a classifier based on gradient features. Finally, words within a text line are separated, giving segmented word areas at the output of the system.

3.2.1 Candidates extraction

First stage of the proposed text detection method consists of finding character candidates. For this purpose, we have combined two different region detectors commonly used for text detection in the state of the art.

Firstly, MSER [Matas et al., 2002] has been proved by [Chen et al., 2011; Neumann and Matas, 2012] to be one of the best region detectors as it is robust against changes in scale, viewpoint and lighting conditions. However, it is very sensitive to image blur and MSER cannot detect small letters in blurred images. The MSER algorithm extracts from an image $I(x, y)$ a number of co-variant regions, called Maximally Stable Extremal Regions (MSERs). An MSER is a stable connected component of some level sets of the image $I(x, y)$. MSER regions fulfill the following properties:

- Invariance to affine transformation of image intensities.
- Covariance to adjacency.
- Stability.
- Multi-scale detection.

MSERs are extracted from the gray-scaled version of the input color image (MSER+) and from the inverted image (MSER-). In the first case, bright-on-dark regions are detected, while in the second case dark-on-bright components are extracted.

On the other hand, the image decomposition method proposed by [Yao et al., 2007] is able to detect most of characters in an image, even small characters, but it produces too many character candidates. It is a locally adaptive thresholding method, which computes a threshold over each pixel by a local mean. This method backs on the idea that characters in an image usually have sharp edge and similar color. This algorithm is described by (3.15).

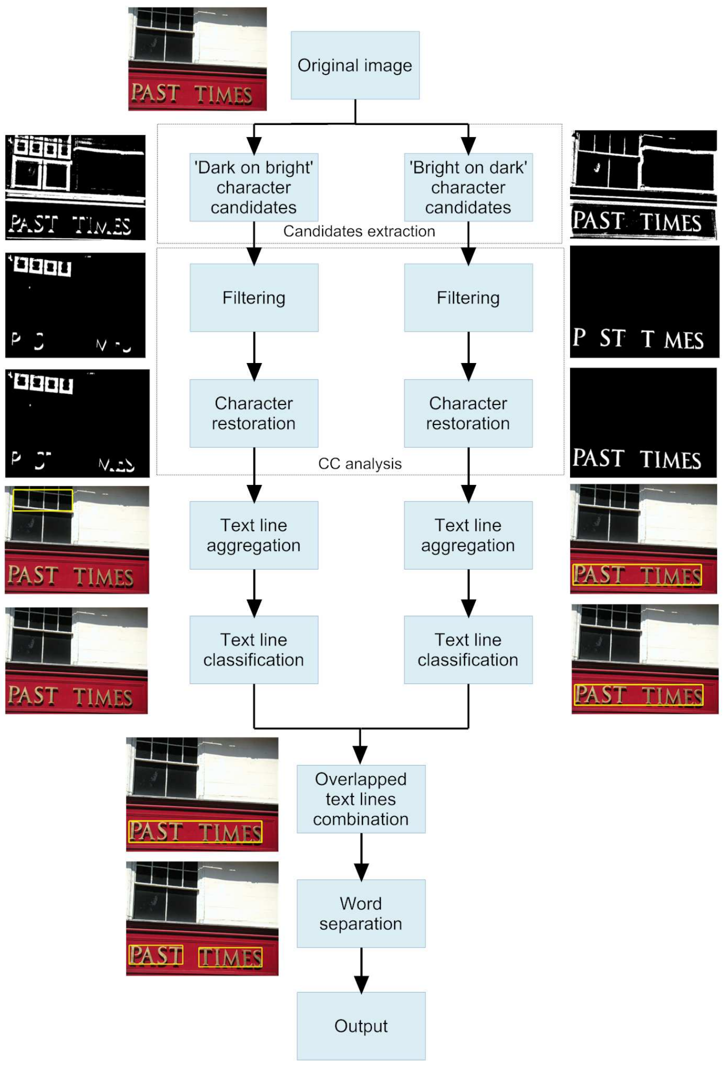


Figure 3.8: The flowchart of the text location algorithm

$$Segment(x,y) = \begin{cases} 255 & \text{if } I(x,y) > T_{+}(x,y) \\ 0 & \text{if } I(x,y) < T_{-}(x,y) \\ 128 & \text{otherwise} \end{cases} \tag{3.15}$$

$I(x,y)$ is the gray-scaled version of the input image. $T_{+}(x,y)$ and $T_{-}(x,y)$ are the local upper and lower thresholds corresponding to $I(x,y)$. They are computed using (3.16).

$$T_{\pm}(x,y) = Mean(x,y,W_B) \pm offset \tag{3.16}$$

$Mean(x,y,W_B)$ is the mean of the pixels in the square block whose center is (x,y) and the size of the block is W_B. The offset is a positive integer by which pixels that do not belong to character components are segmented into the gray layer. This parameter can be considered as a threshold of variance which removes non-character pixels which have low variance, as pixels that belong to character objects have high locally variance. For the parameters of the algorithm, we respect the values provided by [Yao et al., 2007], which are claimed to be optimum by the authors. These values are: $offset = 10$ and $W_B = 71$.

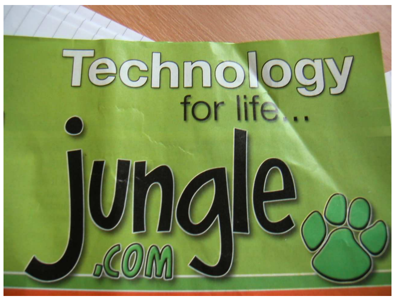
The output of this method is a three-layer image (white, gray and black) in which objects in white or black layers are candidate character pixels. Objects in the white layer are bright objects on darker regions in the original image, while objects in the black layer are dark objects on brighter regions. The pixels on the gray layer do not belong to character components.

In this thesis, it is proposed to combine the complementary properties of the MSER detector and Yao's method. This is done using a AND logical operation, but it is necessary to take into account the fact that the polarity of text in an image can be bright or dark respect to its background. Consequently, both situations can be present in an image. We search for both dark-on-bright and bright-on-dark letter candidates. For the subsequent CC analysis, we are assuming that the polarity of text in a line does not change. Firstly, input image is decomposed using the Yao's method, thus pixels are classified in one of these three layers: black, gray and white. Later, dark-on-bright MSER regions (MSER-) are extracted for the input image and are combined with the pixels of the black layer using an AND logical operation. In the meantime, bright-on-dark MSER regions (MSER+) are extracted and combined with the pixels of the white layer in the same way. The conjunction of both methods helps to reduce the number of letter candidates given by the Yao's method and as a consequence it increases the efficiency of the whole method proposed here. It is also capable of separating some regions that MSER is not able to segment on its own. Figure 3.9 shows the whole segmentation process.

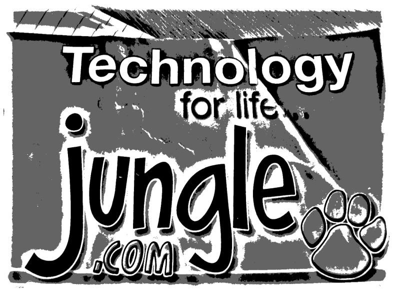
$$\begin{aligned} Bright-on-dark\ CCs &= (MSER+)\ AND\ (White\ layer) \\ Dark-on-bright\ CCs &= (MSER-)\ AND\ (Black\ layer) \end{aligned} \tag{3.17}$$

3.2.2 Connected component analysis

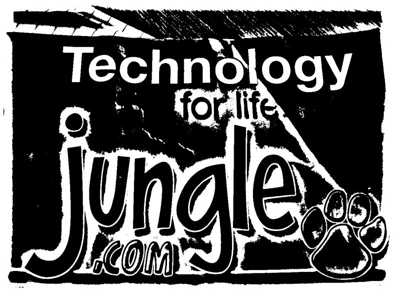
After the previous stage, two binary images are obtained. The foreground CCs for each of these images are considered character candidates. In order to filter out non-character components, the prior features obtained in section 3.1 are applied at this moment of the algorithm. Features in table 3.7 are computed for each candidate and the corresponding thresholds are applied on each one. Those objects for which at least one of the features does not lie within the range given by the thresholds, are discarded. However, some text



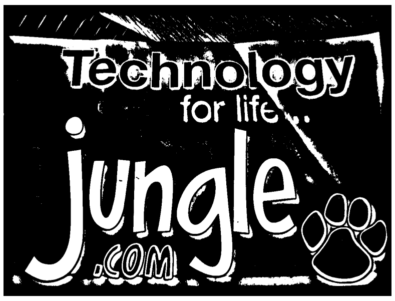
(a) Input image



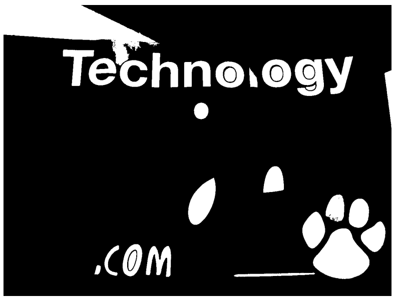
(b) Image decomposed by Yao's method



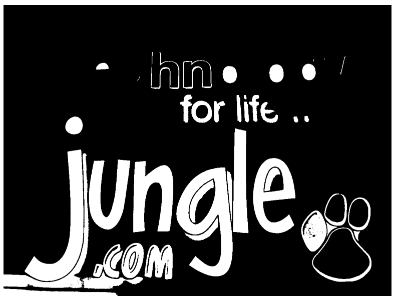
(c) Bright-on-dark objects (Yao)



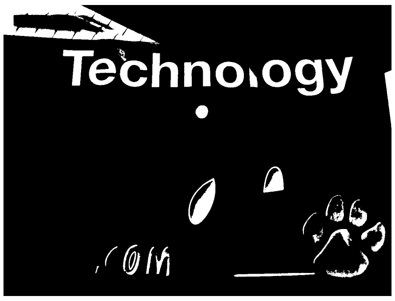
(d) Dark-on-bright objects (Yao)



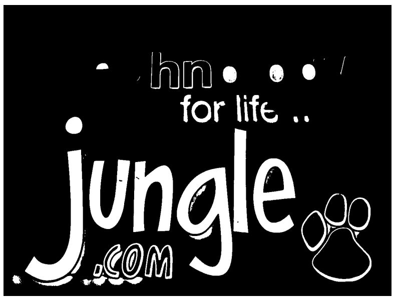
(e) Bright-on-dark regions (MSER)



(f) Dark-on-bright regions (MSER)



(g) Bright-on-dark character candidates (*(c)* AND *(e)*)



(h) Dark-on-bright character candidates (*(d)* AND *(f)*)

Figure 3.9: Image segmentation.

candidates can be erroneously discarded, especially those letters which have a high aspect ratio such as 'i' or 'l'. In order to bring back the mistakenly removed characters, a method to restore them is applied.

We are assuming that text is horizontally aligned. Therefore, for each removed object, we look for the closest non-rejected objects to the left and to the right and apply some constraints between the non-removed and the removed objects. These rules relate to position, size, alignment and stroke width and are shown in (3.18)-(3.25).

$$\frac{Width_{removed}}{Height_{removed}} \leq T_1 \tag{3.18}$$

$$Height_{removed} \geq 10 \tag{3.19}$$

$$Nholes_{removed} \leq 2 \tag{3.20}$$

$$\frac{Max(Height_{removed}, Height_{non\ removed})}{Min(Height_{removed}, Height_{non\ removed})} \leq T_2 \tag{3.21}$$

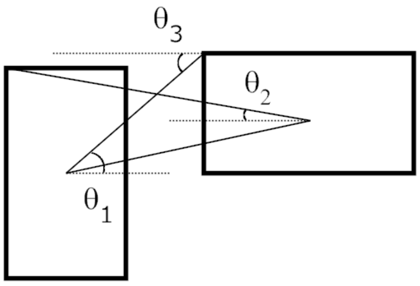
$$\frac{Max(Stroke\ width_{removed}, Stroke\ width_{non\ removed})}{Min(Stroke\ width_{removed}, Stroke\ width_{non\ removed})} \leq T_3 \tag{3.22}$$

$$\theta_1 < T_4\ or\ \theta_2 < T_5\ or\ \theta_3 < T_6 \tag{3.23}$$

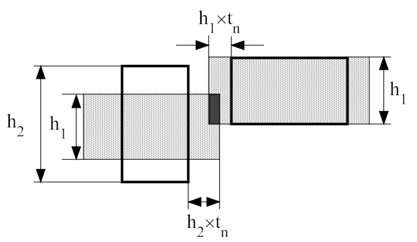
$$Area_{overlap} > T_7 \tag{3.24}$$

$$\frac{Max(Area_{removed}, Area_{non\ removed})}{Min(Area_{removed}, Area_{non\ removed})} \leq T_8 \tag{3.25}$$

θ_1, θ_2 and θ_3 refer to the angles shown in figure 3.10(a). On the other hand, condition (3.24) means that the removed and the non-removed objects have to be overlapped in the magnified regions shown in figure 3.10(b). Conditions (3.23)-(3.25) have been proposed by Ashida in [Lucas et al., 2005].



(a) Alignment



(b) Adjacency ($t_n = 0.4$)

Figure 3.10: Restoring conditions.

If all of these constraints are fulfilled, the removed component is accepted as a valid character object. This method is done recursively for each removed component, until no more objects can be restored.

The threshold values T_i ($i = 1 \ldots 8$) in (3.18)-(3.25) have been optimized using genetic algorithms (GAs) [Goldberg, 1989]. GAs are adaptive heuristic search algorithms premised on the evolutionary ideas of natural selection and genetic by mimicking the same processes that mother nature uses. GAs create a population of solutions and apply genetic operators such as mutation and crossover to evolve the solutions in order to find the best ones. GAs have been widely studied, experimented and applied in many fields in the engineering world. Not only do GAs provide an alternative method to solve a problem, they consistently outperform other traditional methods. They are less susceptible to getting stuck at local optima than gradient-search methods. However, they tend to be computationally expensive. In our case, we try to maximize for the training set the function shown in (3.26).

$$G = \frac{Sensitivity + Specificity}{2} \tag{3.26}$$

The sensitivity is the proportion of characters components correctly restored respect to the total of character objects that were initially discarded. It is defined in (3.27). On the other hand, the specificity is the proportion of non-characters correctly discarded respect to the total number of non-characters initially discarded. Its definition is shown in (3.28).

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.27}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.28}$$

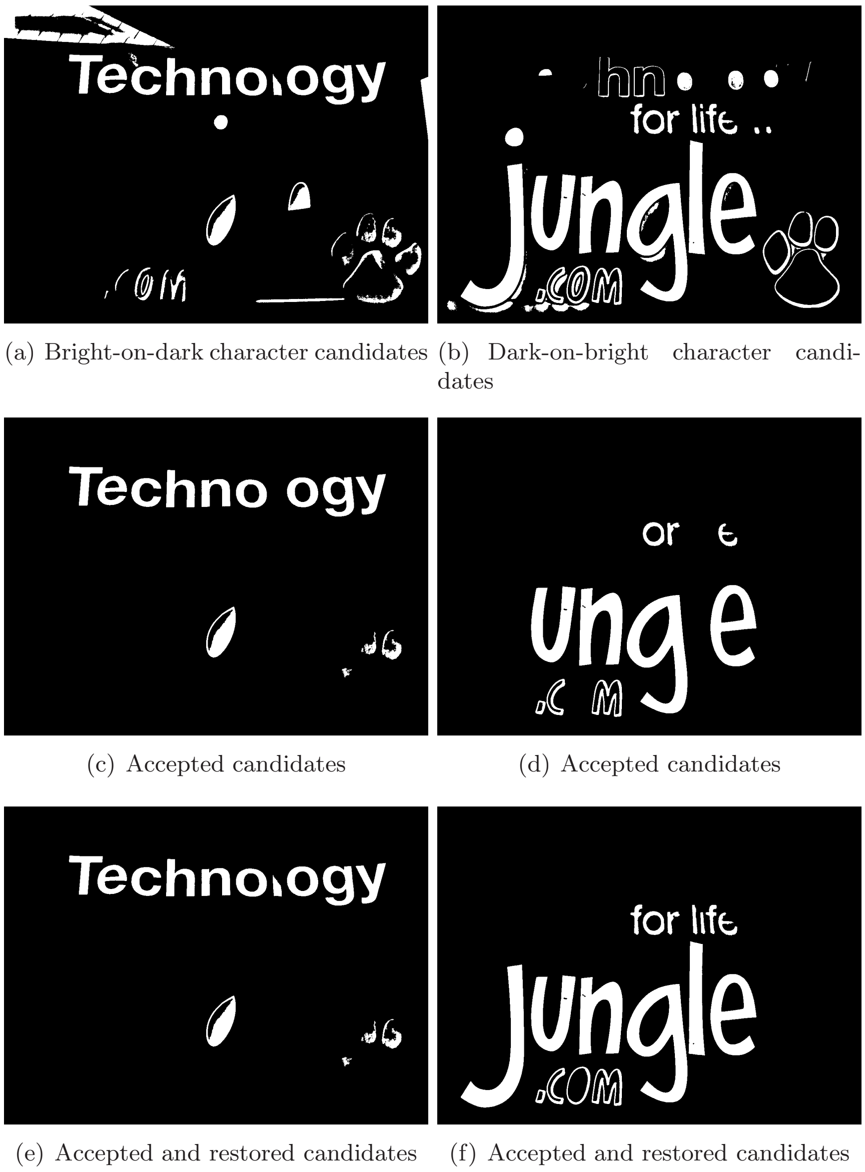
TP stands for the number of true positives (characters correctly restored), FP is the number of false positives (non-characters erroneously restored), TN is the number of true negatives (non-characters correctly discarded) and FN stands for the number of false negatives (characters erroneously discarded).

The optimum values of the thresholds achieved after applying genetic algorithms are shown in table 3.10.

Threshold	Value
T_1	4.5584
T_2	1.8460
T_3	2.4618
T_4	40.6427°
T_5	56.2280°
T_6	44.8965°
T_7	13.1653
T_8	11.0159

Table 3.10: Optimum thresholds for restoring conditions.

The CC analysis explained in this section is carried out for each binary image separatedly, because we are assuming that the polarity of text along a text line does not change. Figure 3.11 shows an example of how the method detailed in this section works.

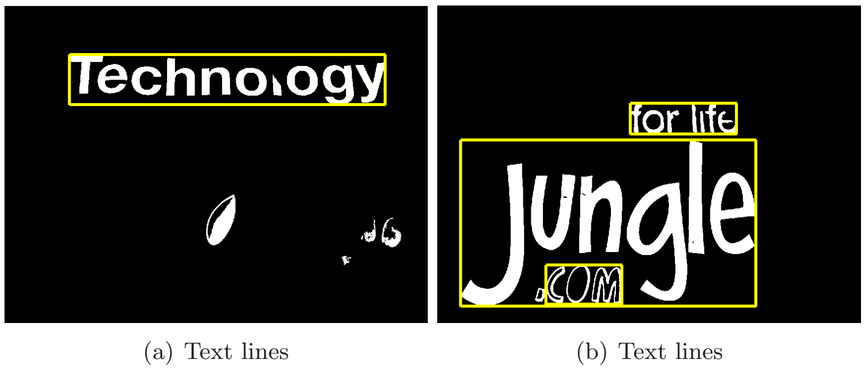


(a) Bright-on-dark character candidates (b) Dark-on-bright character candidates

(c) Accepted candidates (d) Accepted candidates

(e) Accepted and restored candidates (f) Accepted and restored candidates

Figure 3.11: Connected component analysis.



(a) Text lines (b) Text lines

Figure 3.12: Text line aggregation.

3.2.3 Text line aggregation

The accepted components in the previous section are considered character candidates. Letters usually do not appear alone in images. They form words and groups of letters. In this thesis, we are assuming text horizontally aligned, thus characters on a line are expected to have some similar attributes, such as stroke width, height, alignment, adjacency and constant inter-letter and inter-word spacing. These properties have been exploited by most of works in the state of the art. For instance, [Yao et al., 2007; Neumann and Matas, 2012; Epshtein et al., 2010; Chen et al., 2011; Pan et al., 2011] propose very similar sets of binary component features to characterize the similarities in terms of spatial relationship and geometry between two neighboring components, as shown in tables 3.11, 3.12, 3.13, 3.14 and 3.15.

Horizontal alignment	Short spatial distance
Stroke width similarity	Gray-level similarity
Width similarity	Height similarity
Aspect ratio similarity	

Table 3.11: Binary component features proposed by [Yao et al., 2007].

Short spatial distance	Height similarity
Stroke width similarity	Color similarity

Table 3.12: Binary component features proposed by [Epshtein et al., 2010].

Short spatial distance	Height similarity
Stroke width similarity	

Table 3.13: Binary component features proposed by [Chen et al., 2011].

Short spatial distance	Height similarity
Width similarity	Color similarity
Aspect ratio similarity	Horizontal alignment
Stroke width similarity	

Table 3.14: Binary component features proposed by [Neumann and Matas, 2012].

Short spatial distance	Height similarity
Width similarity	Gray-level similarity
Overlap area	Ordering confidences

Table 3.15: Binary component features proposed by [Pan et al., 2011].

In this thesis, it is proposed to use the same features than other authors have done unanimously, except for the ordering confidences used by [Pan et al., 2011], which is a probability value given by the text region detector proposed in that work and it is not applied in this thesis, and the color and gray-level similarities used by [Yao et al., 2007; Neumann and Matas, 2012; Epshtein et al., 2010; Pan et al., 2011]. Taking into account color or gray-level similarities can make that adjacent characters that belong to the same line can be erroneously not grouped together in case they have different color or different illumination, which can be a typical situation in natural images, especially the last condition as illumination changes are very common. Aspect ratio similarity is neither used as it can make that adjacent characters that belong to the same text line and have a very different aspect ratio (for instance, 'l' and 'm') are erroneously not grouped together. For the same reason, width similarity is not taken into account. Therefore, it is proposed to use the set of features shown in table 3.16.

Short spatial distance	Height similarity
Horizontal alignment	Adjacency
Size similarity	Stroke width similarity

Table 3.16: Binary component features proposed in this thesis.

The formulation to express this set of binary component features is given below.

$$\frac{Max(Height_{ch1}, Height_{ch2})}{Min(Height_{ch1}, Height_{ch2})} \leq T_2 \tag{3.29}$$

$$\frac{Max(Stroke\ width_{ch1}, Stroke\ width_{ch2})}{Min(Stroke\ width_{ch1}, Stroke\ width_{ch2})} \leq T_3 \tag{3.30}$$

$$\theta_1 < T_4\ or\ \theta_2 < T_5\ or\ \theta_3 < T_6 \tag{3.31}$$

$$Area_{overlap} > T_7 \tag{3.32}$$

$$\frac{Max(Area_{ch1}, Area_{ch2})}{Min(Area_{ch1}, Area_{ch2})} \leq T_8 \tag{3.33}$$

θ_1, θ_2, θ_3 and $Area_{overlap}$ refer to the figure 3.10. Height similarity and stroke width similarity are given by (3.29) and (3.30) respectively. Short spatial distance, horizontal alignment and adjacency are given by (3.31) and (3.32), while (3.33) means size similarity. The threshold values T_2-T_8 are the optimum ones shown above in table 3.10.

It is important to remark what it is understood as text line in this thesis. Not all the characters that are horizontally aligned in an image (that is, with the same vertical coordinates) may belong to the same line. They must share some other attributes such as having similar height, similar stroke width and being near from each other, which mean that they form part of the same context. Therefore, groups of characters that are horizontally aligned but are written with different font, have different size and/or are too far one from each other, would belong to different text lines. Even, words that belong to the same context in the image can be considered by this method to belong to different lines if they are too far from each other.

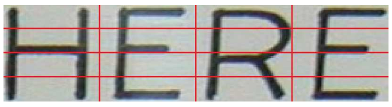
Initially, letter candidates are grouped into pairs if they fulfill the conditions (3.29)-(3.33). Then, two pairs are merged together if they share one of their initial or ending elements. This process of merging chains of candidates is repeated until no more groups can be merged. We are assuming that letters do not appear alone in an image, so we reject those groups which have less than 3 elements. It will be shown in the results section that this value is the optimum one. The accepted groups are considered to be text lines. Figure 3.12 shows an example.

Finding these groups of letters means that the proposed method cannot detect isolated characters and text lines with less than 3 objects, but it is an effective way of filtering out objects that were erroneously classified as letters in the previous stage of the algorithm.

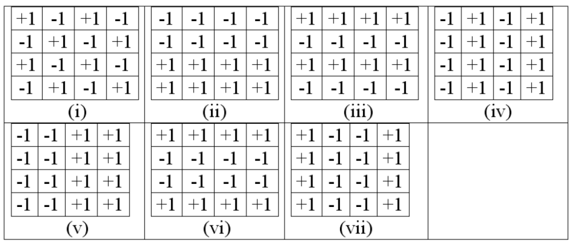
3.2.4 Text line classification

Repeating structures such as windows, bricks or fences, commonly seen in urban scenes, can lead to mistakenly accept text lines in the previous step of the algorithm. This problem has been tried to be solved by some authors. For instance, [Chen et al., 2011] applies template matching among the letter candidates in a text line and, if a significant portion of the objects are repetitive, the text line is considered to be a false positive and is discarded. Also, based on the assumption that most letters have low solidity, which is the proportion of the object pixels in the convex hull (refer to equation 3.9), a text line is rejected if most of the objects within that line have a very large solidity.

A less heuristic approach has been proposed by [Hanif and Prevost, 2009]. Three different types of features are extracted on each text region: Mean Difference Feature (MDF), Standard Deviation (SD) and Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005]. To compute these features, each text line is resized to 64×256 pixels and a grid of 16×64 pixels is placed on the text window, thus making a total of 16 blocks per text window, as shown in figure 3.13(a). The mean value for each block is computed, giving a vector of 16 components. In order to make it independent to brightness, blocks are weighted by convoluting them with the masks shown in figure 3.13(b) and the absolute value is taken. This is the Mean Difference Feature (MDF). The standard deviation (SD) of each block is also computed, obtaining a 16 dimensional vector. Finally, HOG is computed for each block, resulting in a total of 16 histograms of 8 dimensions. Therefore, the dimensionality of the feature set is 39 features (7 MDF, 16 SD and 16 HOG). The classification is carried out using an AdaBoost approach with 780 weak classifiers based on a Likelihood Ratio Test, which is a statistical test for taking a decision between two hypotheses.



(a) Grid



+1	-1	+1	-1
-1	+1	-1	+1
+1	-1	+1	-1
-1	+1	-1	+1

(i)

-1	-1	-1	-1
-1	-1	-1	-1
+1	+1	+1	+1
+1	+1	+1	+1

(ii)

+1	+1	+1	+1
-1	-1	-1	-1
+1	+1	+1	+1
-1	-1	-1	-1

(iii)

-1	+1	-1	+1
-1	+1	-1	+1
-1	+1	-1	+1
-1	+1	-1	+1

(iv)

-1	-1	+1	+1
-1	-1	+1	+1
-1	-1	+1	+1
-1	-1	+1	+1

(v)

+1	+1	+1	+1
-1	-1	-1	-1
-1	-1	-1	-1
+1	+1	+1	+1

(vi)

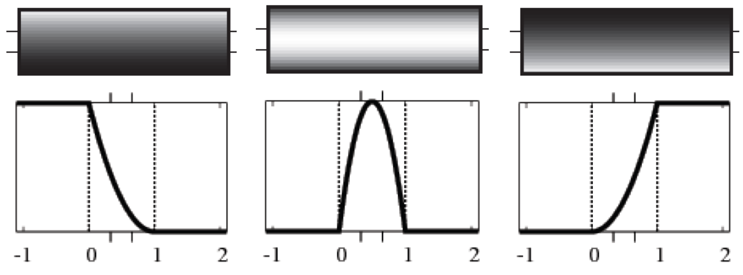
+1	-1	-1	+1
+1	-1	-1	+1
+1	-1	-1	+1
+1	-1	-1	+1

(vii)

(b) MDF weights

Figure 3.13: MDF classifier.

On the other hand, [Minetto et al., 2011] propose to use Fuzzy HOG (F-HOG) features. This descriptor is based on the idea that the distribution of the directions of the image gradient is different in the top, the middle and the bottom parts of text lines. Each text line is scaled to a fixed height of 21 pixels, maintaining its original aspect ratio, and HOG features are computed, with 18 bins, over the top, middle and bottom part of the text, using Bernstein weight functions as shown in figure 3.14 and computed as in (3.34), and L1 normalization. The feature vector has 54 elements.



(a) w_0 (b) w_1 (c) w_2

Figure 3.14: Bernstein weight functions for F-HOG classifier.

$$w_k(z) = \begin{cases} 1 & \text{if } k = 0 \textit{ and } z \leq 0 \\ 1 & \text{if } k = 2 \textit{ and } z \geq 0 \\ \frac{\beta_k^n(z)}{\beta_k^n(\frac{k}{n})} & \text{if } 0 < z < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.34}$$

where

$$\beta_k^n(z) = \binom{n}{k} z^k (1-z)^{n-k} \tag{3.35}$$

for $k = 0, 1, ..., n$ and

$$z = \frac{y - y_{top}}{y_{bot} - y_{top}} \tag{3.36}$$

In (3.36), y is the vertical coordinate of the pixel, and y_{top} and y_{bot} are the estimated y coordinates of the top and bottom contours of the text in the image.

In this thesis, we test three different classifiers and the comparative results are shown in section 3.3. All the classifiers have been trained with the training set of the ICDAR 2003 dataset. We have labelled manually a set of positive and negative samples, in a proportion of 1:2 approximately. Each classifier is based on SVM with linear kernel.

First classifier consists of standard HOG features. Each text line is resized to a window of 64×256 pixels and HOG features are extracted using a cell size of 8×8 pixels, 9 bins and a range of 180°, for each color channel red, green and blue. A text line candidate is accepted if it is classified as text for at least one of the color channels.

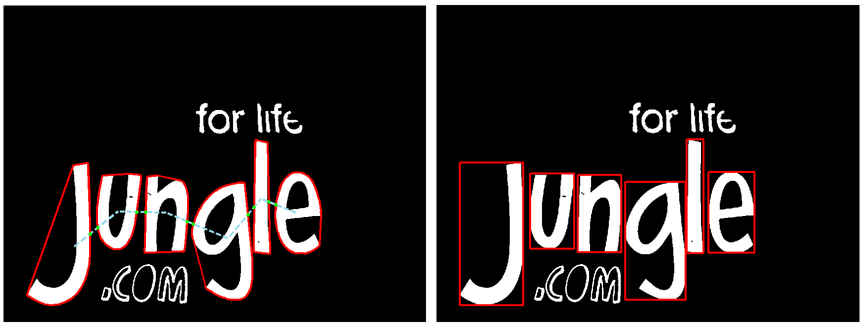
The second tested classifier is based on the features proposed by [Hanif and Prevost, 2009]: MDF, SD and HOG. All features are normalized by their euclidean norm in order to make the classifier independent to contrast.

The third classifier is based on the F-HOG features proposed by [Minetto et al., 2011].

Experimental results will show that the best classifier is the second one based on HOG, MDF and SD features.

3.2.5 Word separation

Finally, text lines are split into words. We compute the distance between each pair of adjacent letters and classify them into one of these two classes: intra-word separation or inter-word separation. Character separation inside a word tends to be constant, while separation of two consecutive characters that belong to two different words is consistently higher than the distance between two characters of the same word. The proposed method to separate words combines two ways of computing the gap between two adjacent letters. The first one consists of computing the euclidean distance between the intersection points of the convex hull of each character and the line that joins the centroids of both letters (see figure 3.15(a)). The second one consists of computing the horizontal distance between the bounding boxes of both letters (figure 3.15(b)). In first place, the two types of distances are computed for each pair of adjacent characters. Then, the mean values of all the distances are computed for both metrics separatedly. Two adjacent characters are separated into different words if both distances are higher than 1.8 times their means. This thresholds have been computed heuristically. This method minimizes the disadvantages of using both metrics separatedly. The first metric can erroneously separate two adjacent characters that belong to the same word but their centroids have a big vertical separation. An example is given by letters 'g' and 'l' in figure 3.15(a). On the other hand, the second metric gives negative or zero values for adjacent components whose bounding boxes overlap.



(a) Convex hulls (b) Bounding boxes

Figure 3.15: Word separation.

3.3 Experimental results

We evaluate the proposed method by running it on several public datasets and comparing to the state of the art. The chosen datasets have been used as a benchmark for most of researchers working in the field of text location in the last decade. However, each dataset has its own characteristics and its own way of measuring the performance of an algorithm tested on it. In this section, we show the results obtained with each dataset in order to compare our method with other state-of-the-art works. Before that, we display some partial results obtained with one of the datasets in order to show the importance of the contributions of the proposed framework.

3.3.1 Partial experiments

In this section, we show some experiments that prove the contribution of the proposed segmentation method, the importance of the CC restoration process and the usefulness of the text line classifier. These results have been obtained using the test set of the ICDAR 2003 Robust Reading Competition dataset. Therefore, the evaluation metrics proposed for that competition have been used. These metrics are defined in [Lucas et al., 2003]. The precision and recall are computed using (3.37) and (3.38) respectively.

$$p = \frac{\sum\limits_{r_e \in E} m(r_e, T)}{|E|} \tag{3.37}$$

$$r = \frac{\sum\limits_{r_t \in T} m(r_t, E)}{|T|} \tag{3.38}$$

T and E are the sets of ground-truth and estimated rectangles in the test image respectively, while $m(r, R)$ is the best match for a rectangle r in a set of rectangles R, defined in (3.39).

$$m(r, R) = \max m_a(r, r') \mid r' \in R \tag{3.39}$$

where

$$m_a(r_1, r_2) = \frac{2a(r_1 \cup r_2)}{a(r_1) + a(r_2)} \tag{3.40}$$

and $a(r)$ is the area of rectangle r.

The precision measure is strongly affected by the number of false positives (the larger the number of false positives is, the smaller this magnitude is), while the recall tries to measure the accuracy of the detection. An f metric is used to combine both precision and recall into one single measure. It is defined as in (3.41). It is used to compare the performance of different methods.

$$f = \frac{1}{\frac{\alpha}{p} + \frac{\alpha}{r}} \tag{3.41}$$

In order to give equal weight to precision and recall, α is set to 0.5. It must be taken into account that the f measure usually varies from 0.75 to 1.0 even when all text is correctly located, because it is unlikely to give estimated rectangles as output which exactly align to the manually labelled ground-truth.

In the case of reporting the results for a set of images, as it is the case, p, r and f are computed for each single image and averaged over the images in the test set. Therefore, precision, recall and f measure are computed using (3.42)-(3.44) respectively, in case of N images.

$$p_N = \frac{\sum\limits_{k \in N} p_k}{N} \tag{3.42}$$

$$r_N = \frac{\sum\limits_{k \in N} r_k}{N} \tag{3.43}$$

$$f_N = \frac{\sum\limits_{k \in N} f_k}{N} \tag{3.44}$$

As it was explained in section 3.2, the segmentation method is based on a combination of MSER and Yao's algorithm. Table 3.17 shows the improvement of using both methods instead of using only MSER and only Yao's method separatedly.

Algorithm	Precision	Recall	f
MSER [Matas et al., 2002]	0.85	0.53	0.65
Yao [Yao et al., 2007]	0.67	0.57	0.62
MSER and Yao	0.81	0.57	0.67

Table 3.17: Comparison of segmentation algorithms.

The precision is higher when only MSER is used, but the recall is much lower. On the contrary, Yao's precision is lower but its recall is higher. It means that the second method is able to segment most of the characters in the image, but it also generates a much bigger number of false positives. The combination of both methods helps to maintain a high recall while the precision is hardly affected.

We have used the Vedaldi's implementation for the MSER algorithm [Vedaldi and Fulkerson, 2008]. The MSER algorithm has several parameters that can be modified. The most significant are: Delta (it defines the stability of a region, which is the relative variation of the region area when the intensity is changed $\pm Delta/2$), MinDiversity (when the relative area variation of two nested regions is below this threshold, only the most stable one is selected) and MaxVariation (a threshold that sets the maximum variation of the regions). Figure 3.16 shows how varying these parameters affects the results. The optimum results are achieved with the values Delta=12.5, MinDiversity=0.9 and MaxVariation=0.3.

In section 3.2, it was stated that those lines with less than 3 characters are rejected. This threshold has been found to be optimum, as it is shown in figure 3.17. It can be seen that the lower this threshold is, the higher the recall is, but the precision and the f-measure are optimum for 3 characters.

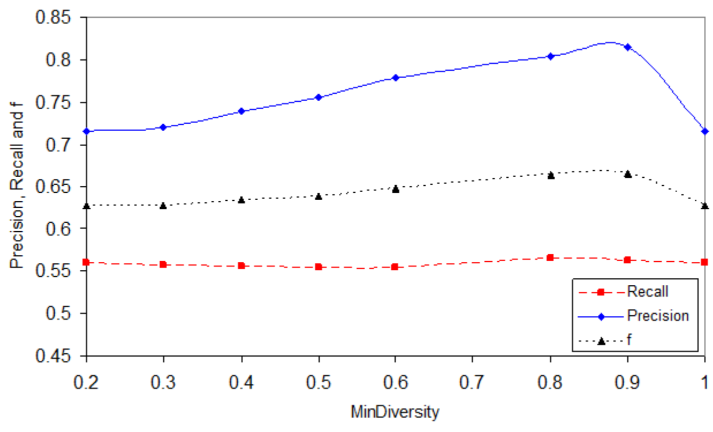
It was also explained in section 3.2 that the text features can be approached to Gaussian distributions and that we are using the range $(\mu - 2 \cdot \sigma, \mu + 2 \cdot \sigma)$ for each feature to discard between text and non-text objects. Figure 3.18 shows how varying this range affects to the results. It is clearly seen that we get the optimum results for $k = 2$ in $(\mu - k \cdot \sigma, \mu + k \cdot \sigma)$.

The use of the range $(\mu - 2 \cdot \sigma, \mu + 2 \cdot \sigma)$ for the text features is optimum, thus a high number of true letters is accepted by keeping low the number of false positives. However, some character candidates can be erroneously rejected, thus leading to a partial text detection. In order to bring back these mistakenly removed characters, a method to restore them is applied. Table 3.18 shows the importance of this block.

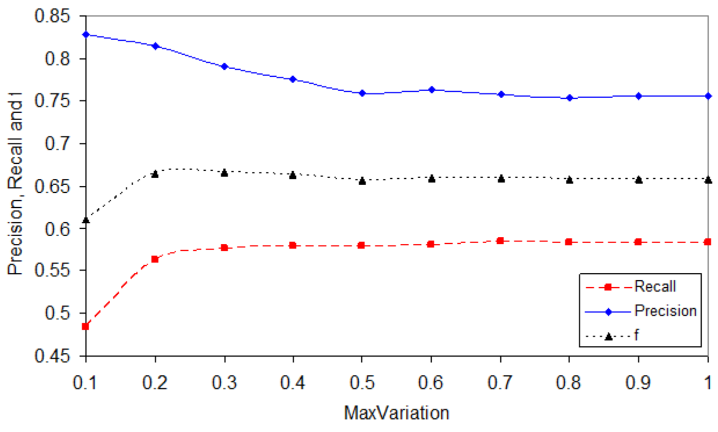
Algorithm	Precision	Recall	f
No restoration	0.69	0.33	0.45
Restoration	0.81	0.57	0.67

Table 3.18: Effect of using the character restoration.

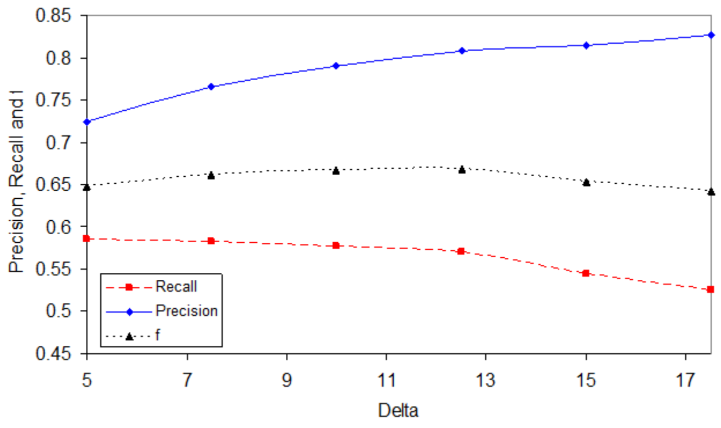
Finally, we have tested the three different classifiers explained in subsection 3.2.4. Table 3.19 shows the comparison between each classifier and the improvement of using a



(a) Effect of MinDiversity



(b) Effect of MaxVariation



(c) Effect of Delta

Figure 3.16: Effect of varying the MSER parameters.

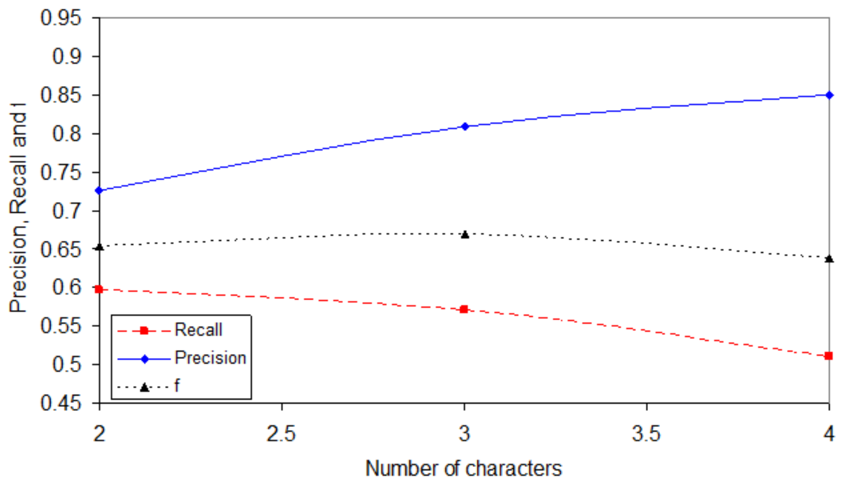


Figure 3.17: Effect of varying the minimum number of characters per text line.

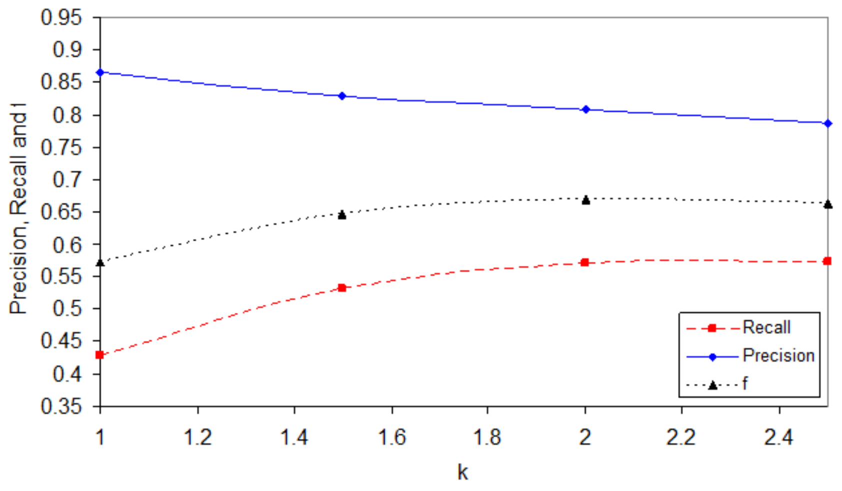


Figure 3.18: Effect of varying the ranges of the features $(\mu - k \cdot \sigma, \mu + k \cdot \sigma)$.

classifier against not using it. If a classifier is not used, the recall is slightly higher but the precision is much lower compared to the case when a classifier is used. It means that the classifier discards many false positives and a few number of true positives. The f-measure turns out to be higher when a classifier is used. The highest f-measure is obtained when HOG on the one hand and MDF, SD and HOG features on the other hand are used, but the precision is a bit higher in the second case. That is the reason why we have chosen to use MDF, SD and HOG features.

Classifier	Precision	Recall	f
No classifier	0.71	0.59	0.65
HOG	0.80	0.57	0.67
MDF+SD+HOG	0.81	0.57	0.67
F-HOG	0.83	0.55	0.66

Table 3.19: Comparison of text line classifiers.

3.3.2 ICDAR 2003/2005 dataset

A Robust Reading Competition was organized at the International Conference on Document Analysis and Recognition (ICDAR) in 2003 [Lucas et al., 2003, 2005]. The competition was divided into three sub-problems: text location, character recognition and word recognition. In this section, we show the results for the first one. It received five entries. A new competition was held at ICDAR 2005 [Lucas, 2005] using the same dataset. Again, only five participants took part on the competition. The dataset released for these competitions was divided into two sections: a training set that contains 258 images and 1157 words and a test set with 251 images and 1111 words. The images were obtained by the organizers of the first competition with different digital cameras, which were used with a range of resolution and other settings, with the particular settings chosen at the discretion of the photographer. The images correspond to different outdoor and indoor scenarios with a wide variety of text appearance in terms of font, thickness, color, size, texture, lighting conditions, blur, viewpoints and occlusions. The metrics to evaluate the performance of the text location have been explained in the previous subsection.

Table 3.20 shows the performance of our algorithm on the ICDAR 2003/2005 dataset, together with the performance of the winners of the Robust Reading competitions at ICDAR 2003 and ICDAR 2005 and some other methods that have used this dataset as a benchmark in the last decade. It can be seen that we score third in the global ranking, although we outperform the results obtained in the framework of ICDAR 2003 and 2005 competitions. Actually, our approach scores first in terms of precision. It means that the number of false positives that our algorithm produces is the smallest one.

Some output samples are shown in figures 3.19, 3.20, 3.21 and 3.22. The results of the detection are shown in yellow or blue colors. The reason to use different colors is just to display the results better. Most of the images present correct detections and separation into words. It can be seen that the proposed algorithm achieves optimum results in many different difficult situations. Figure 3.19(a) shows an example of text whose characters are very difficult to segmentate because they are very close and the tonality of the color varies even at each character. Text with certain orientation in the image due to non-perpendicular shooting angle is also detected, as it is displayed in figures 3.19(b), 3.20(c), 3.20(d), 3.20(f), 3.21(b), 3.22(a) or 3.22(e), among others. Text embedded in complex background or present in images with low contrast is also detected by the proposed framework, as it can be seen in figures 3.19(e), 3.20(f) and 3.20(g). Moreover, small text or blurred text can be also detected. Some examples are figures 3.19(f), 3.20(f), 3.21(b) and 3.21(h).

On the other hand, the text detection can fail or can be not precise enough due to different reasons. For instance, characters in a word which have different characteristics respect to the other components in the same word, may not be correctly extracted. An example is the letter ‘W’ in “WRIGLEY’S” in figure 3.19(d), whose stroke width is bigger than the other letters in the same word, or the letter ‘T’ in “TEETH” in figure 3.21(c). Another example is the letter ‘j’ in “jungle”, whose height is much bigger than the adjacent letters. Another source of errors is due to groups of characters, words or text lines whose number of elements is less than 3, because of the condition imposed in our algorithm, such as words “9” and “12” in figure 3.19(h). Isolated characters that have a long aspect ratio can be also wrongly rejected, such as “1” and “11” in figures 3.21(d) and 3.22(d), respectively. The segmentation can be also difficult when shining is present in the image, like the word “GAS” in figure 3.22(e). The text line classifier can erroneously reject some words, like “PLEASE” in figure 3.21(b). However, we have shown in the previous section that its function is very important as it removes many false positives while it hardly rejects true positives. Actually, the number of false positives is very low. Some examples of images where true positives appear are figures 3.19(c), 3.20(e), 3.21(b), 3.21(c) and 3.22(b).

Algorithm	Precision	Recall	*f*
[Zhang and Lai, 2012]	0.80	0.76	0.78
[Pan et al., 2011]	0.67	0.70	0.69
Our system	**0.81**	**0.57**	**0.67**
[Fehli et al., 2012]	0.75	0.61	0.67
[Phan et al., 2012]	0.68	0.66	0.67
[Epshtein et al., 2010]	0.73	0.60	0.66
[Chen et al., 2011]	0.73	0.60	0.66
[Liu et al., 2012]	0.63	0.65	0.64
[Lee et al., 2010]	0.69	0.60	0.64
[Zhao et al., 2011]	0.69	0.58	0.63
1st ICDAR’05 [Lucas, 2005]	0.62	0.67	0.62
[Yao et al., 2007]	0.64	0.60	0.61
[Minetto et al., 2010]	0.63	0.61	0.61
[Li and Lu, 2012]	0.59	0.59	0.59
Alex Chen [Lucas, 2005]	0.60	0.60	0.58
[Chowdhury et al., 2012]	0.57	0.59	0.58
[Neumann and Matas, 2012]	0.59	0.55	0.57
[Merino et al., 2011]	0.51	0.67	0.55
[Zhang and Kasturi, 2010]	0.67	0.46	0.55
[Liu and Sarkar, 2008]	0.66	0.46	0.54
1st ICDAR’03 [Lucas et al., 2005]	0.55	0.46	0.50
[Hanif and Prevost, 2009]	0.25	0.35	0.29

Table 3.20: Text location ICDAR’03 dataset.

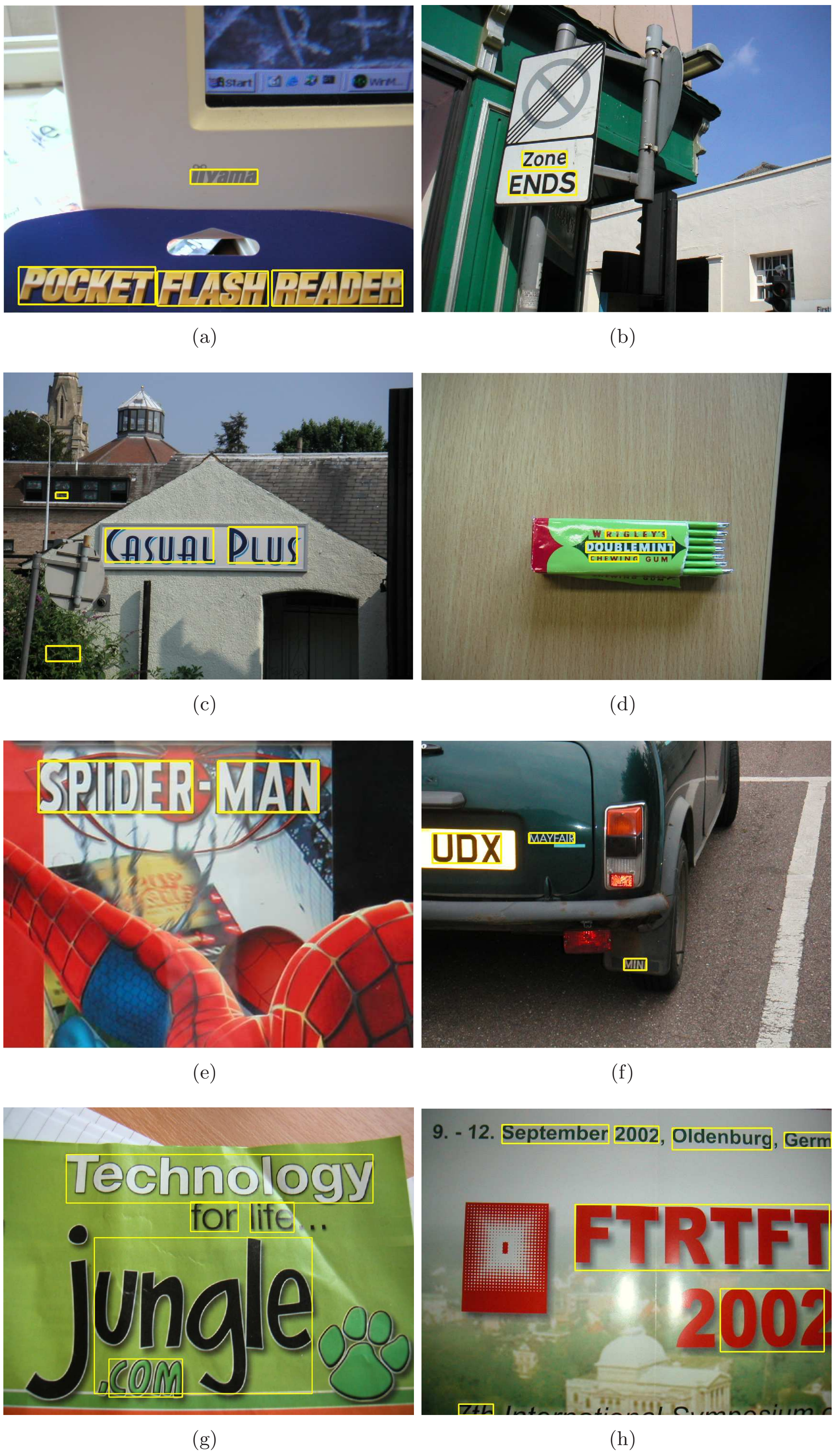


Figure 3.19: Text location results on some images from the ICDAR 2003/2005 dataset.

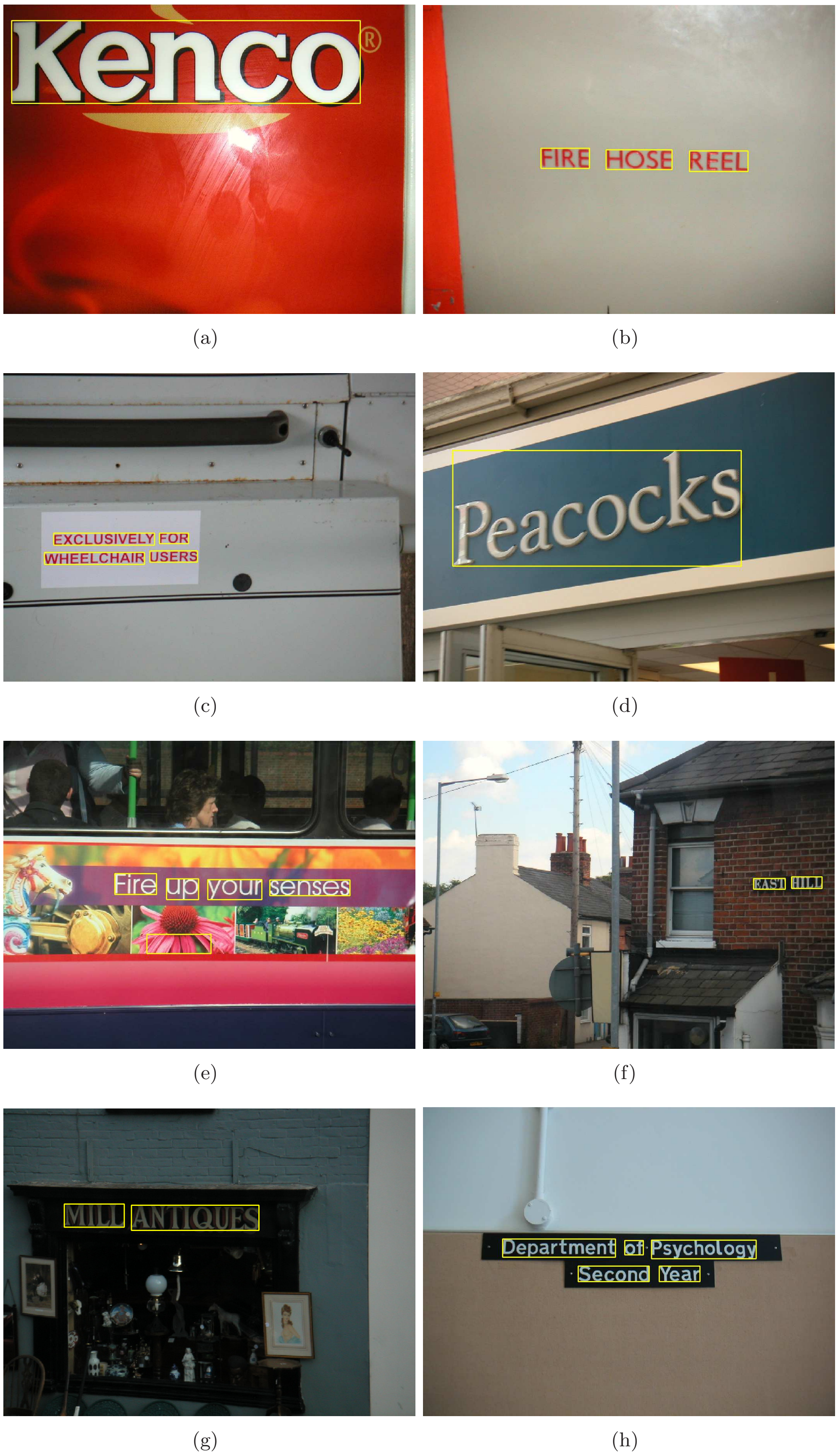


Figure 3.20: Text location results on some images from the ICDAR 2003/2005 dataset.

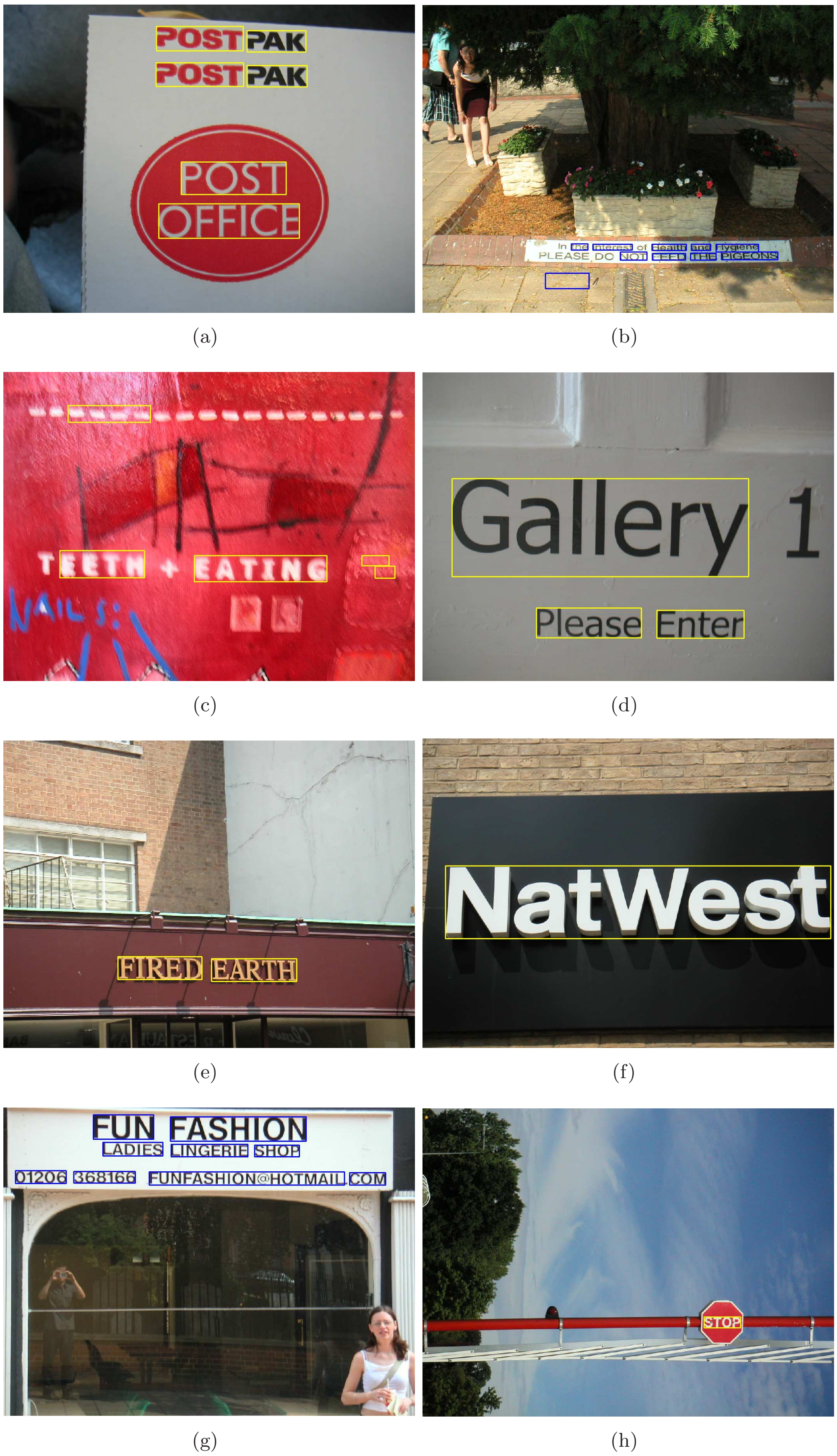


Figure 3.21: Text location results on some images from the ICDAR 2003/2005 dataset.

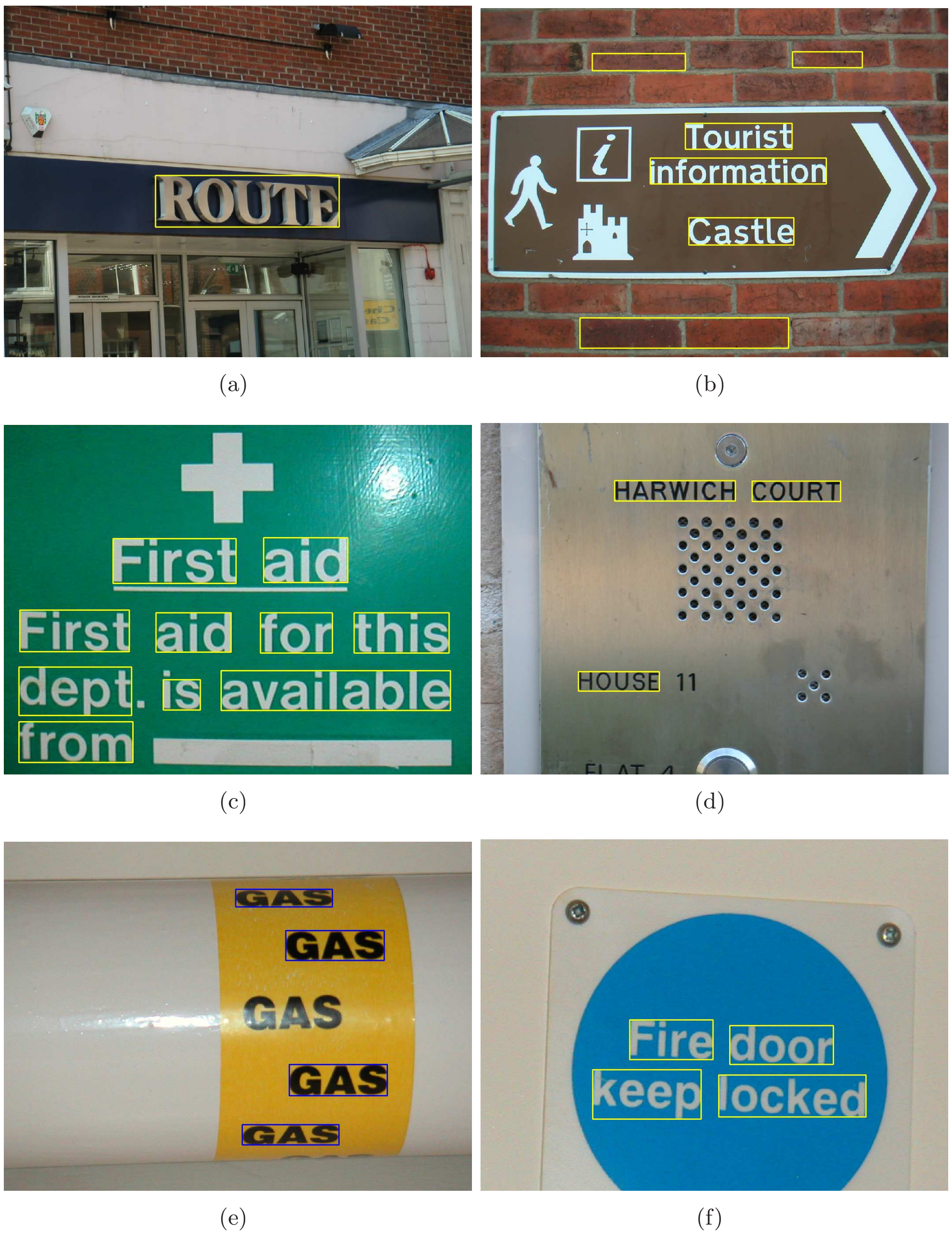


Figure 3.22: Text location results on some images from the ICDAR 2003/2005 dataset.

3.3.3 ICDAR 2011 dataset

A new Robust Reading Competition has been recently held at ICDAR 2011. In this case, the competition was composed of two different challenges. The first challenge deals with born-digital images (web, email), while the second one deals with natural images. The first challenge [Karatzas et al., 2011] is organized over three tasks: text location, text segmentation and word recognition. In this section, we show our results for the first task. The training set for this challenge is composed of 420 images containing 3583 words, while the test set has 102 images containing 918 words. The fact that most of the images in this dataset have a very low resolution and the text is blurred in most of cases presents a further difficulty. On the other hand, the second challenge [Shahab et al., 2011] consists of two tasks: text location and word recognition. We show our results for the first one in this section. The training set is composed of 229 images and 848 words, whereas the test set has 255 images and 716 words. Most of the images for this challenge have been taken for the ICDAR 2003 Robust Reading competition.

In this competition, the organizers propose to use a different way of evaluating the text location performance than the one used in the 2003 and 2005 editions. Many authors have claimed that the precision, recall and f metrics explained in the previous subsection are unfair. The reason for this complaint comes from the equation (3.39), which implies that only one-to-one matches are considered. However, in reality sometimes one ground-truth rectangle is split into several objects or several ground-truth rectangles are merged into a single object. Many authors claim that it is more interesting to evaluate the solution of a detection problem but the ground-truth is specified as the correct and only solution of a big problem. Therefore, an over or undersegmented solution may be a correct detection but it is not considered correct by this way of evaluation.

The text location task is evaluated in ICDAR 2011 datasets using the methodology proposed by Wolf et al [Wolf and Jolion, 2006], who have developed a software to evaluate object detection algorithms [1]. This method improves the evaluation approach used in ICDAR 2003 competition with the precision and recall measures.

If G is the set of all ground-truth rectangles and D is the set of all detected rectangles, recall and precision are defined as in (3.45)-(3.46).

$$P = \frac{\sum\limits_{i \in D} Match_D(D_i, G)}{|D|} \tag{3.45}$$

$$R = \frac{\sum\limits_{j \in G} Match_G(G_j, D)}{|G|} \tag{3.46}$$

where $Match_D$ and $Match_G$ are functions which take into account different types of matches and which evaluate the quality of the match. They are defined in (3.47)-(3.48).

$$Match_D(D_i, G) = \begin{cases} 1 & \text{if } D_i \textit{ matches against a single detected rectangle} \\ 0 & \text{if } D_i \textit{ does not match against any detected rectangle} \\ 0.8 & \text{if } D_i \textit{ matches against several detected rectangles} \end{cases} \tag{3.47}$$

[1] `http://liris.cnrs.fr/christian.wolf/software/deteval/index.html`

$$Match_G(G_j, D) = \begin{cases} 1 & \text{if } G_j \textit{ matches against a single detected rectangle} \\ 0 & \text{if } G_j \textit{ does not match against any detected rectangle} \\ 0.8 & \text{if } G_j \textit{ matches against several detected rectangles} \end{cases} \tag{3.48}$$

In case of N images, precision and recall are not accumulated by averaging the precision and recall values of the single images, but they are weighted by the number of rectangles in each image, as in (3.49)-(3.50).

$$P_N = \frac{\sum\limits_{k \in N} \sum\limits_{i \in D} Match_D(D_i^k, G^k)}{\sum\limits_{k \in N} |D^k|} \tag{3.49}$$

$$R_N = \frac{\sum\limits_{k \in N} \sum\limits_{j \in G} Match_G(G_j^k, D^k)}{\sum\limits_{k \in N} |G^k|} \tag{3.50}$$

The final performance value is the harmonic mean of the two measures (3.51).

$$Hmean = 2\frac{P_N \cdot R_N}{P_N + R_N} \tag{3.51}$$

Table 3.21 and table 3.22 show the comparison of the proposed method in this thesis with the participants in Challenge 1 and Challenge 2 at ICDAR 2011, respectively. It can be seen that our method scores first in Challenge 1 and fourth in Challenge 2. Some detection samples are shown in figures 3.23, 3.24, 3.25, 3.26 and 3.27 and figures 3.28, 3.29, 3.30 and 3.30, respectively, in blue, yellow, cyan and black colors. Again, different colors have been necessary to be used in order to show the results more clear. All the output images for Challenge 1 can be seen in the website of the challenge [2].

In general, the same conclusions have been drawn from these datasets than from the ICDAR 2003 dataset. The proposed text location algorithm achieves a very high performance and the same sources of errors have been detected. A main drawback comes from images of small resolution or images whose text is impossible to be separated into single characters by the segmentation process. An example of this problem is the word "bada" in figure 3.24(h). However, the proposed segmentation is able even to detect small resolution text, like the word "Informal" in figure 3.29(b) or most of the text that appears in the images corresponding to Challenge 1, whose resolution is less than 200-by-200 pixels in most of the cases. More than once, some CCs are rejected in the initial step of the algorithm because of the Gaussian classifier and they are not restored later. For instance, the words "to" and "with" in figure 3.23(a), "Laptops" in figure 3.24(b) or "of" and "a" in figure 3.26(b). The text line classifier may erroneously reject some words or groups of words, such as "4311"in figure 3.29(b), "HERE" in figure 3.29(e), "STAFF" in figure 3.29(f) or "SHOE" in figure 3.31(b). Nevertheless, the text line classifier removes most of the false positives, but some might remain, like it happens in figures 3.23(b), 3.28(f) and 3.30(a).

[2] `http://www.cvc.uab.es/icdar2011competition/?com=results&action=images_list&id_submit=407`

Algorithm	Precision	Recall	H. Mean
Our system	**89.23**	**70.08**	**78.51**
Textorter	85.83	69.62	76.88
TH-TextLoc	80.51	73.08	76.62
TDM IACAS	84.64	69.16	76.12
OTCYMIST	64.05	75.91	69.48
SASA	67.82	65.62	66.70
Text Hunter	75.52	57.76	65.46

Table 3.21: Text location on ICDAR'11 Chall. 1 (%).

Algorithm	Precision	Recall	H. Mean
Kim's method	82.98	62.47	71.28
[Yin et al., 2012]	81.53	62.22	70.58
[Fehli et al., 2012]	84.00	60.00	70.00
Our system	**72.67**	**56.00**	**63.25**
Yi's method	67.22	58.09	62.32
TH-TextLoc	66.97	57.68	61.98
[Li and Lu, 2012]	59.00	62.00	61.00
Neumann's method	68.93	52.54	59.63
TDM IACS	63.52	53.52	58.09
LIP6-Retin	62.97	50.07	55.78
KAIST AIPR System	59.67	44.57	51.03
ECNU-CCG method	35.01	38.32	36.59
Text Hunter	50.05	25.96	34.19

Table 3.22: Text location on ICDAR'11 Chall. 2 (%).

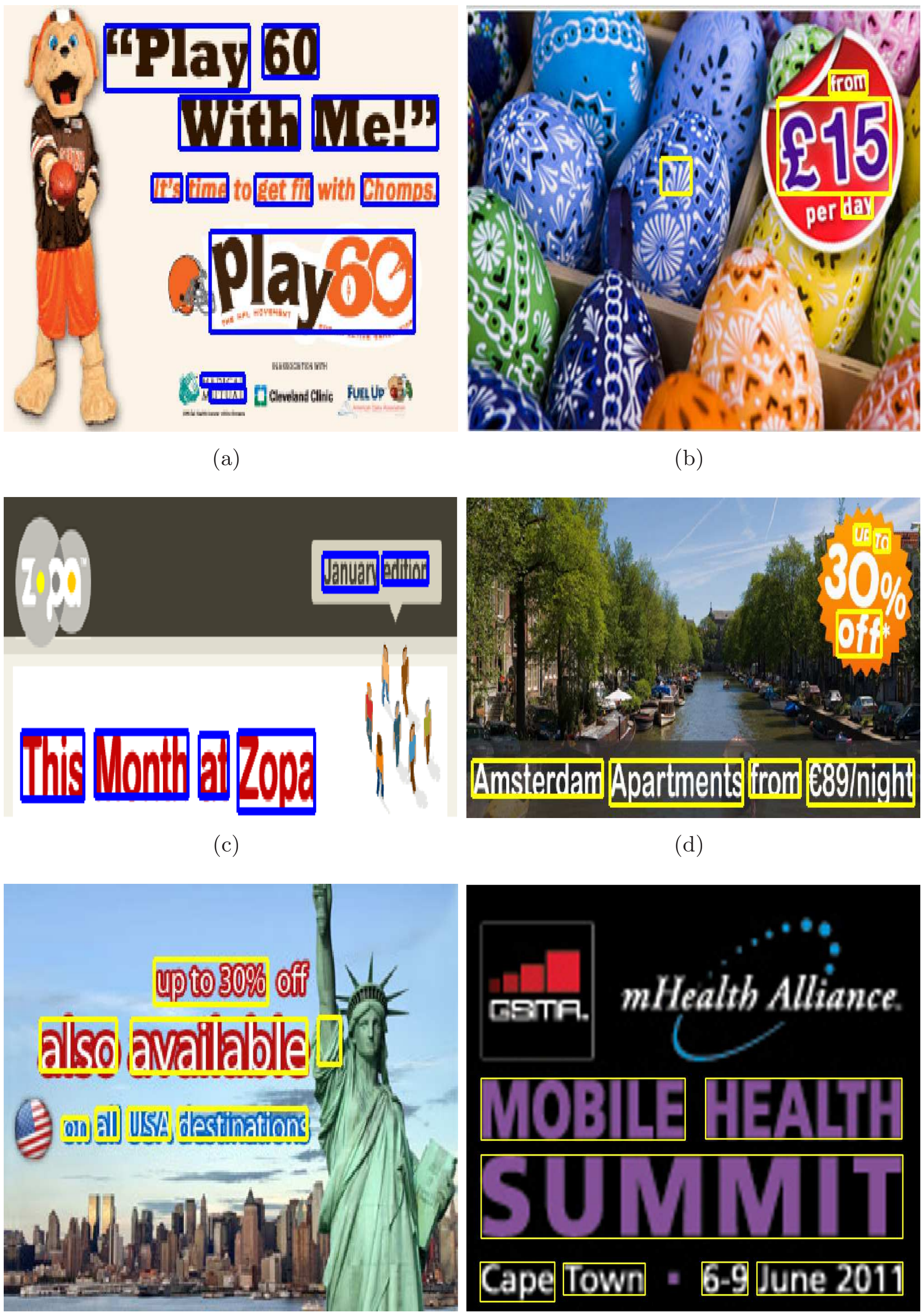
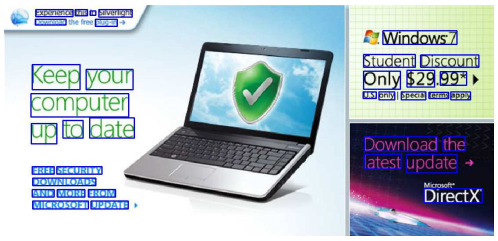
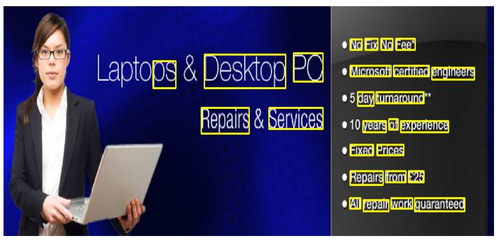


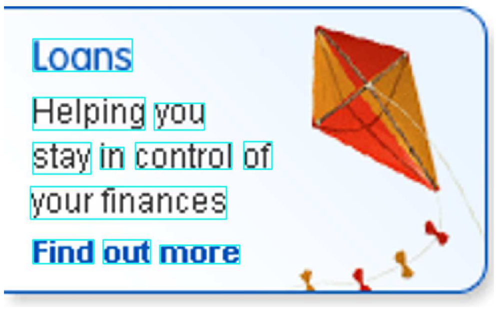
Figure 3.23: Text location results on some images from the ICDAR 2011 Challenge 1 dataset.


Keep your computer up to date
FREE SECURITY DOWNLOADS AND MORE FROM MICROSOFT UPDATE
Windows 7
Student Discount
Only $29.99*
Download the latest update
Microsoft DirectX

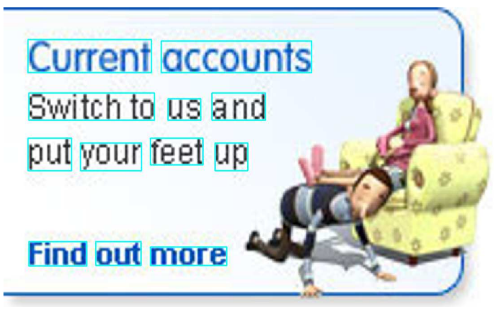
(a)



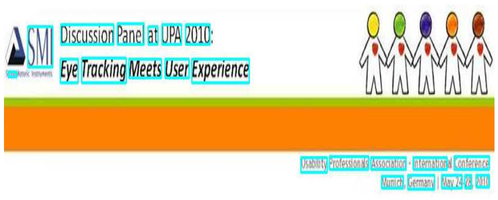
(b)



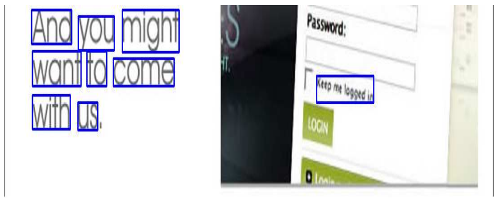
(c)



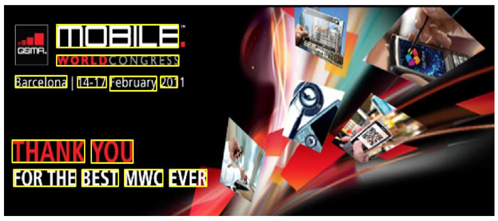
(d)


SMI
Discussion Panel at UPA 2010:
Eye Tracking Meets User Experience
Usability Professionals Association - International Conference
Munich, Germany | May 24-28, 2010

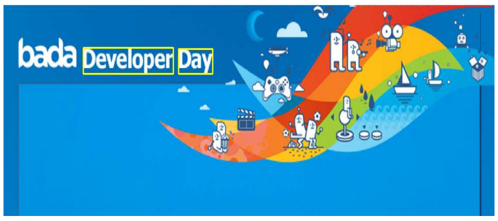
(e)



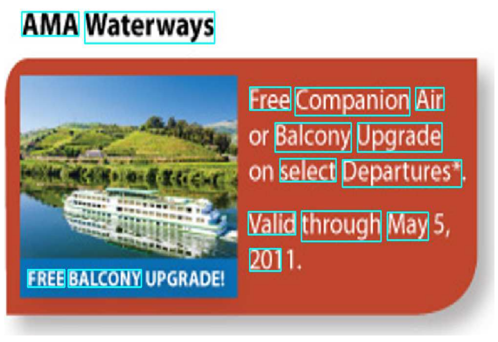
(f)



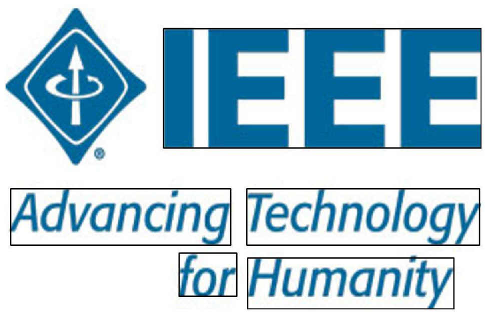
(g)



(h)

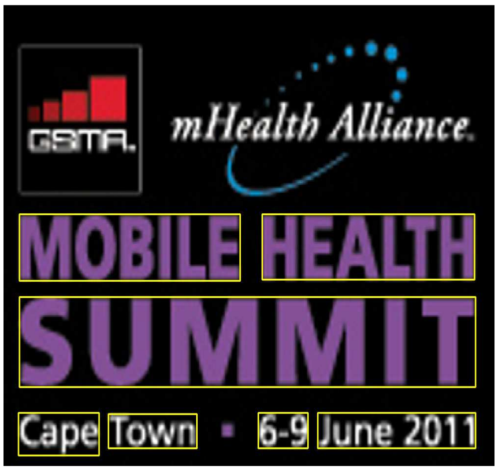

AMA Waterways
Free Companion Air or Balcony Upgrade on select Departures*.
Valid through May 5, 2011.
FREE BALCONY UPGRADE!

(i)

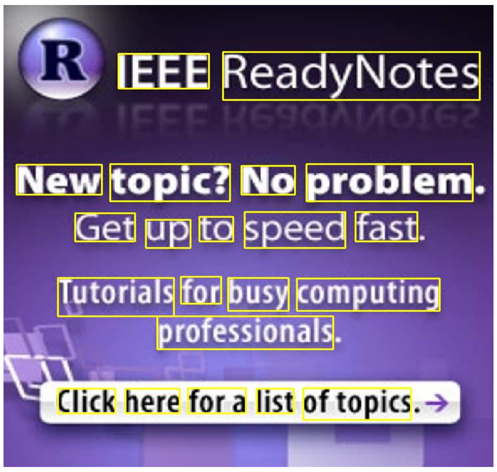


(j)

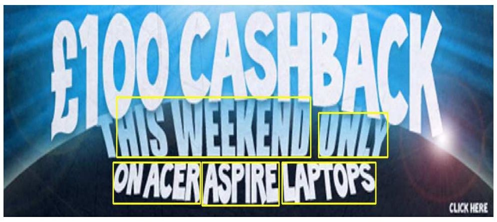
Figure 3.24: Text location results on some images from the ICDAR 2011 Challenge 1 dataset.



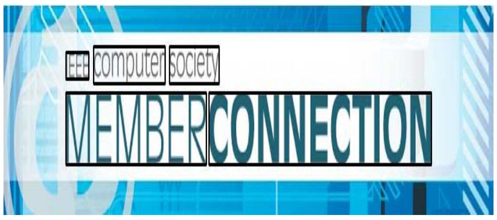
(a)


R IEEE ReadyNotes
New topic? No problem.
Get up to speed fast.
Tutorials for busy computing professionals.
Click here for a list of topics.

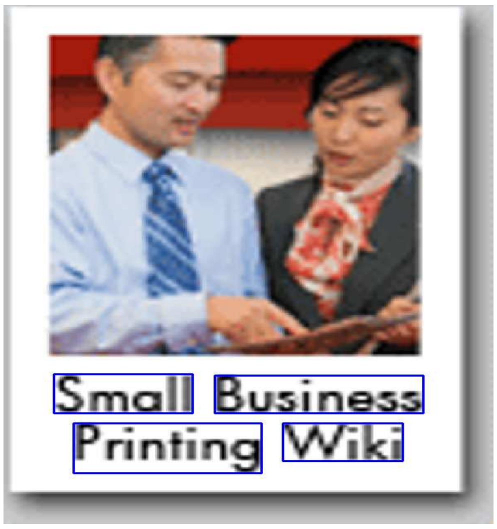
(b)



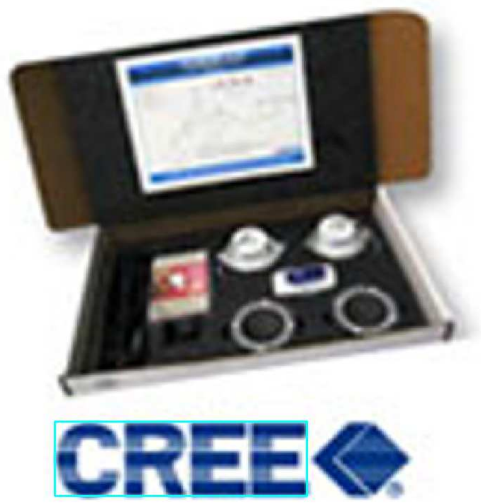
(c)



(d)

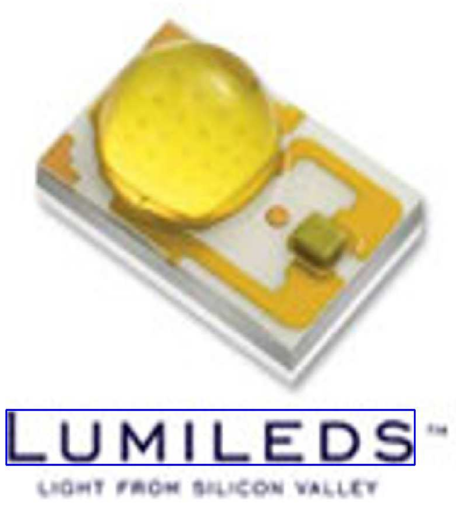


(e)

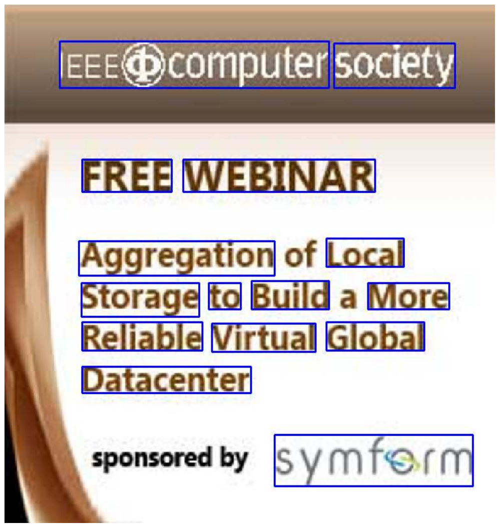


(f)

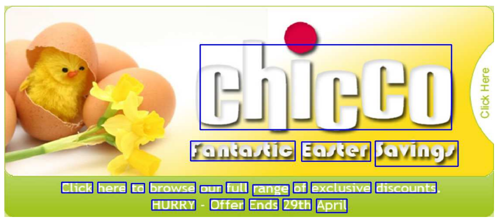
Figure 3.25: Text location results on some images from the ICDAR 2011 Challenge 1 dataset.



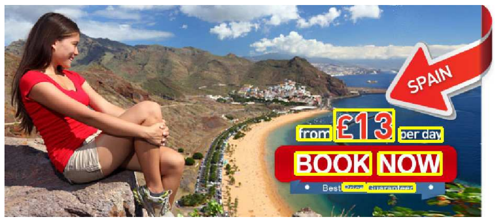
(a)



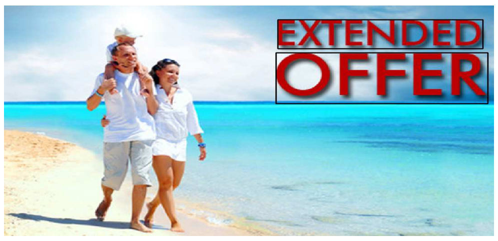
(b)


chicco
Fantastic Easter Savings
Click Here
Click here to browse our full range of exclusive discounts
HURRY - Offer Ends 29th April

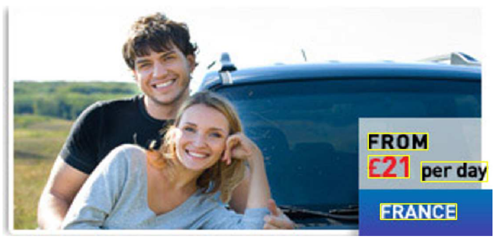
(c)



(d)

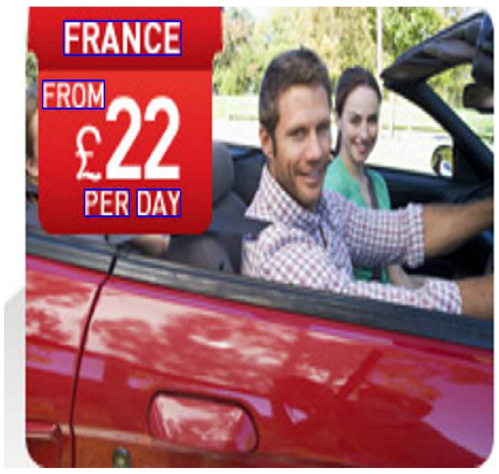


(e)

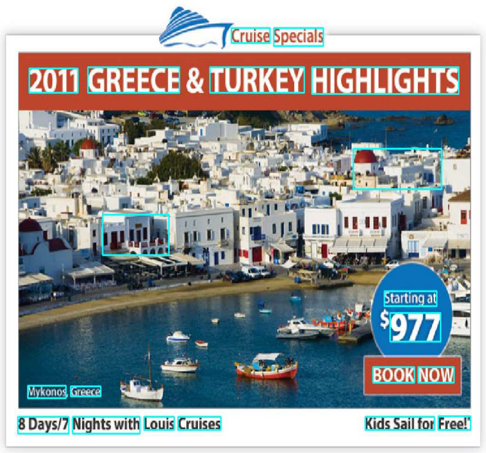


(f)

Figure 3.26: Text location results on some images from the ICDAR 2011 Challenge 1 dataset.



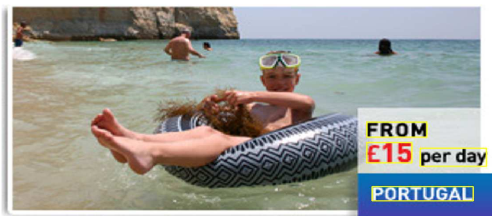
(a)



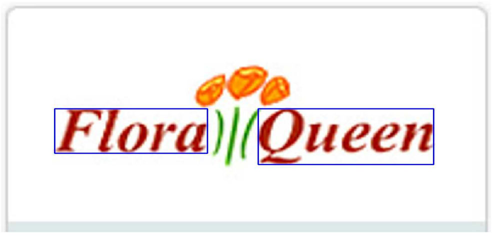
(b)

bambinodirect
the nursery goods specialists

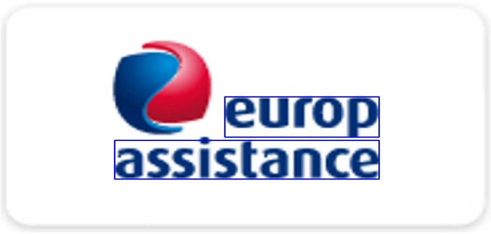
(c)



(d)

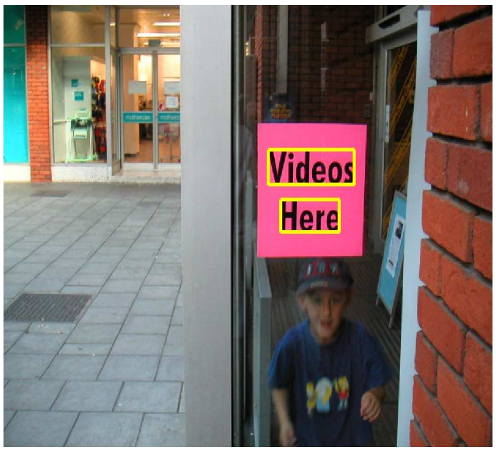


(e)

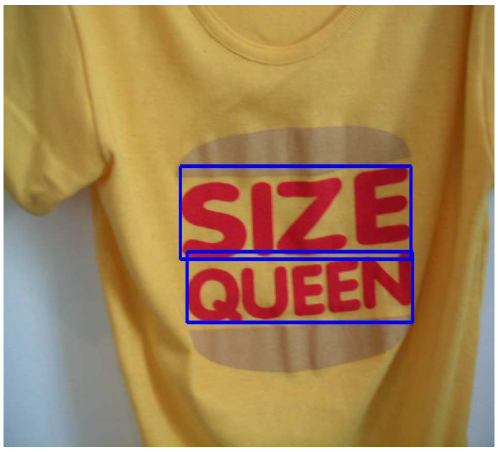


(f)

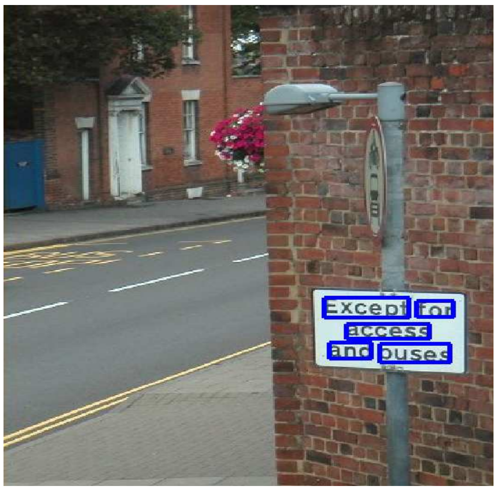
Figure 3.27: Text location results on some images from the ICDAR 2011 Challenge 1 dataset.



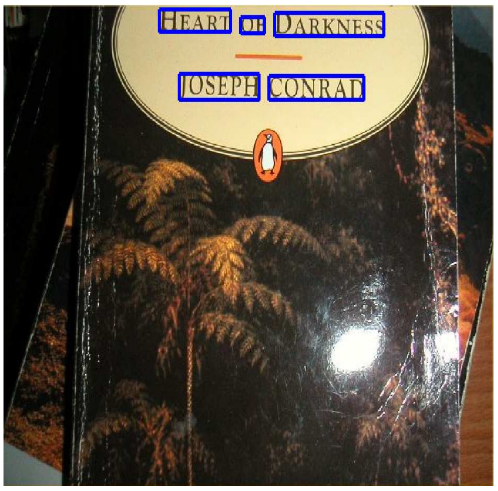
(a)



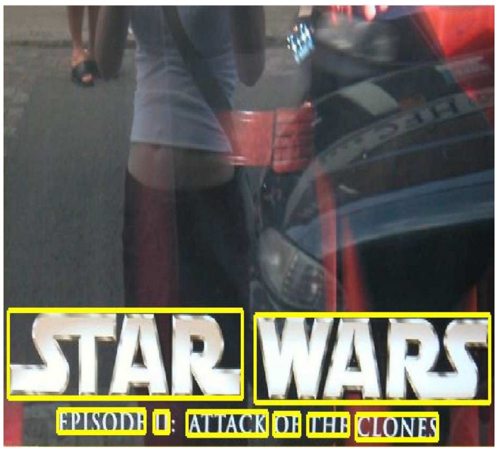
(b)



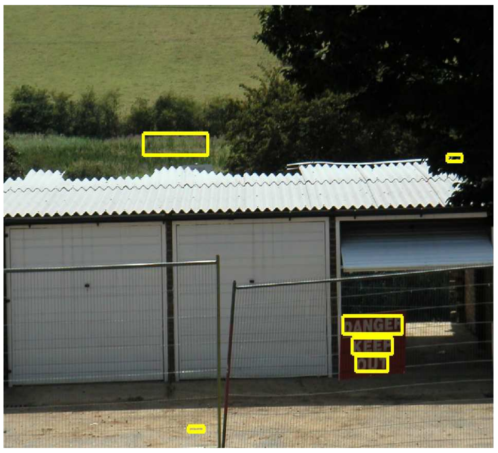
(c)



(d)

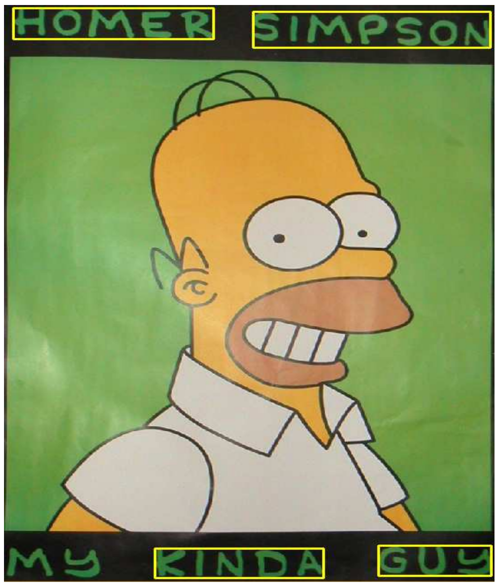


(e)

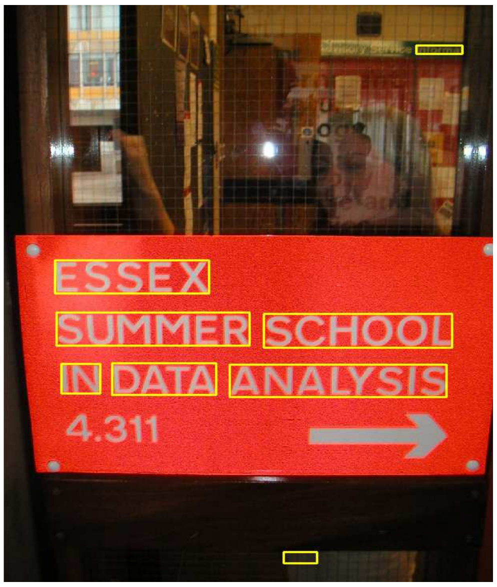


(f)

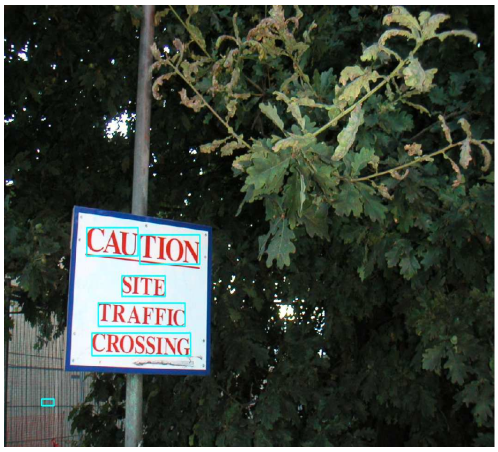
Figure 3.28: Text location results on some images from the ICDAR 2011 Challenge 2 dataset.



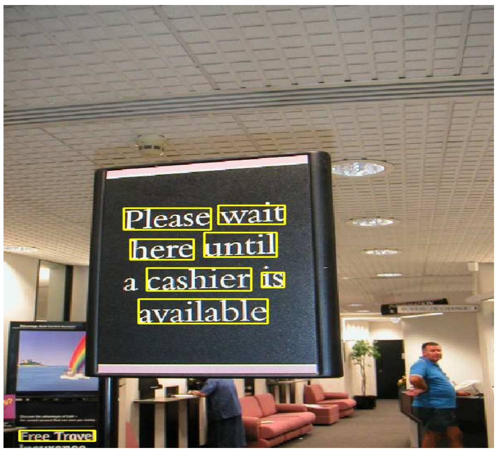
(a)



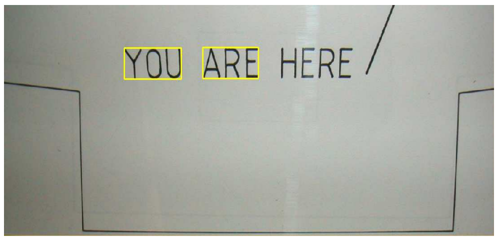
(b)



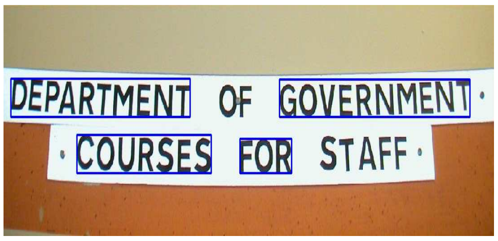
(c)



(d)



(e)



(f)

Figure 3.29: Text location results on some images from the ICDAR 2011 Challenge 2 dataset.

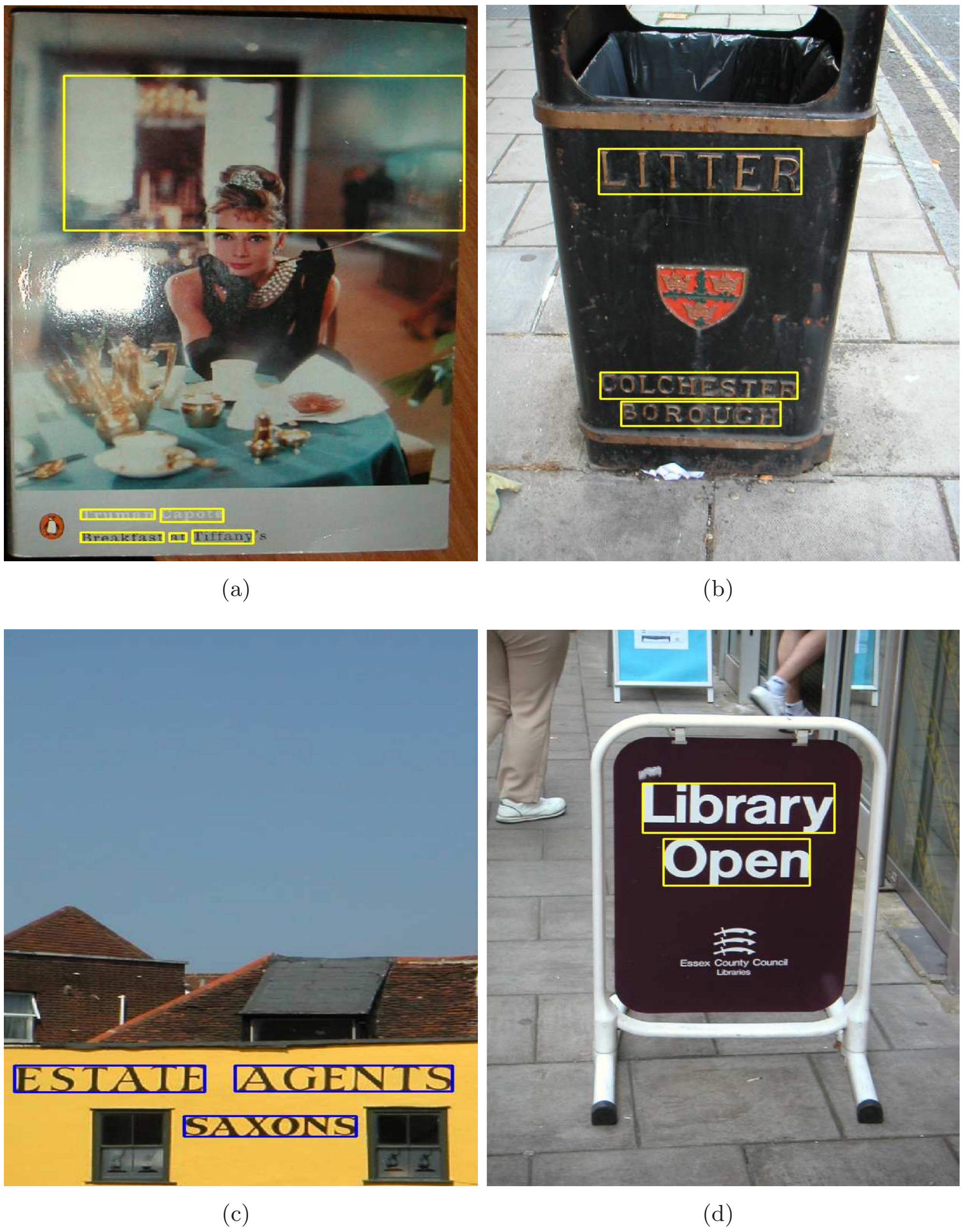


Figure 3.30: Text location results on some images from the ICDAR 2011 Challenge 2 dataset.

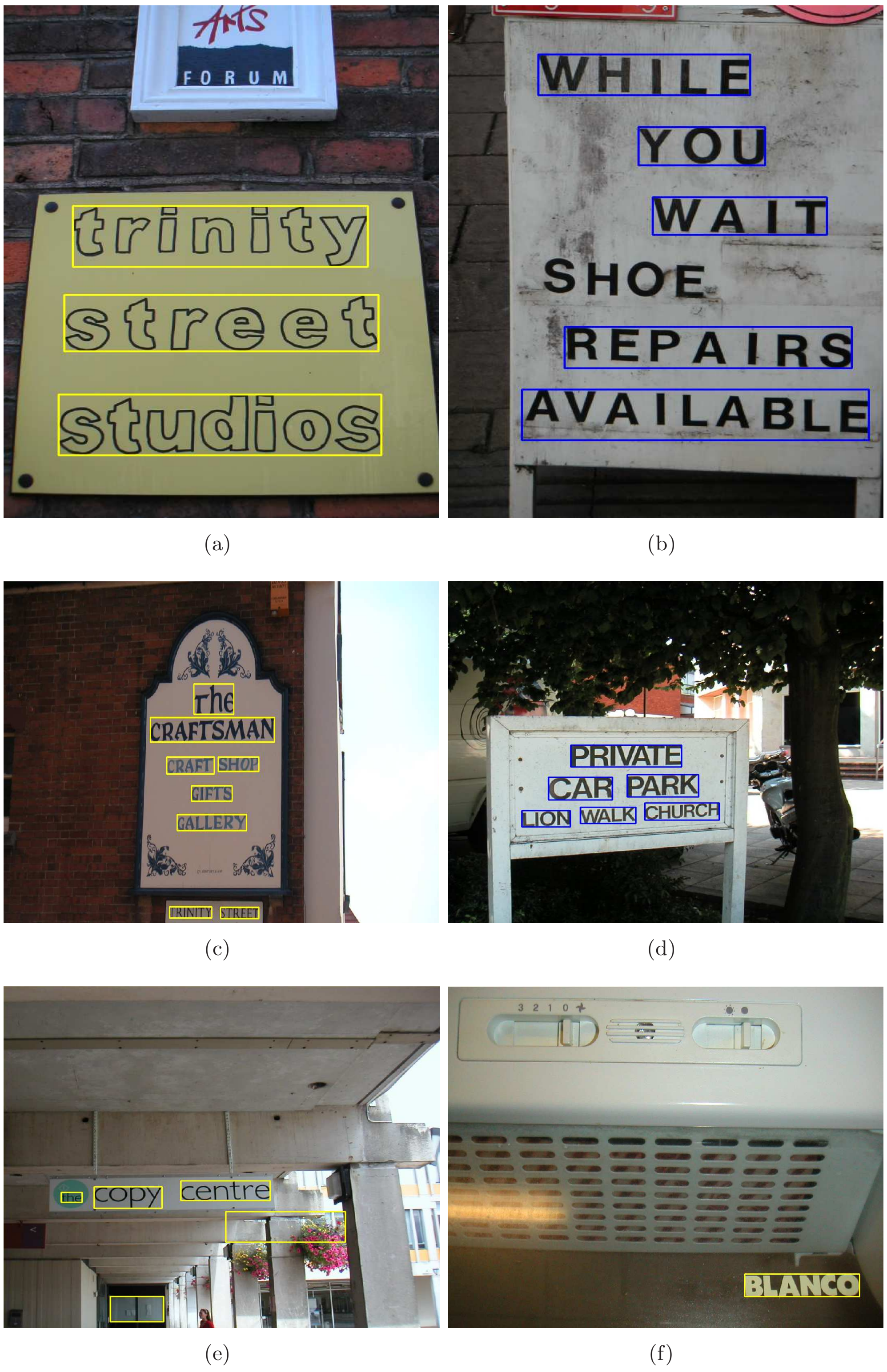


Figure 3.31: Text location results on some images from the ICDAR 2011 Challenge 2 dataset.

3.3.4 CoverDB dataset

An additional test has been carried out with the CoverDB dataset [Escalera et al., 2009], which is composed of CD/DVD cover images. The term "cover" refers to the front-facing panel of a CD/DVD package, and the primary image accompanying a digital download of the album or of its individual tracks. CD covers represent an interesting challenge related to several computer vision and pattern recognition problems. The CoverDB dataset is composed of 663 training images and 300 test images. Most of the images in this dataset have a very low resolution and the text size is very small, thus the text is blurred in the most of the cases.

The evaluation on this dataset is performed using the metric proposed by the creators of the dataset. It consists of comparing the area of the detected text region with the area of the ground truth bounding box, as in (3.52). Table 3.23 shows the comparison of our method to other methods that have tested their performance on this dataset. It is clearly seen that we outperform all the methods. Some examples of the performance of our approach on this dataset are shown in appendix A.

$$Performance = \frac{(Detected\ text\ area) \bigcap (Ground\ truth\ text\ area)}{(Detected\ text\ area) \bigcup (Ground\ truth\ text\ area)} \tag{3.52}$$

Algorithm	Performance
Our system	**0.45**
[Escalera et al., 2009]	0.28
[Cano and Perez-Cortes, 2003]	0.16

Table 3.23: Text location on CoverDB test set.

3.4 Conclusions and future works

In this chapter we proposed a method to detect text in natural images. Our approach is based on connected component analysis, but the idea used on texture-based methods consisting of extracting certain features on large areas and applying machine learning techniques to classify these areas has been also used in our approach to validate the detected text line candidates. We have compared different features and shown the benefits of combining gradient features (HOG) with simple features such as mean and standard deviation computed over blocks in the image.

We have also carried out an analysis of the most suitable features that characterize letters. We have studied the most commonly used in the state of the art and have added new features. This has been done not only for unary component features, but also for binary component features that characterize the similarities between neighboring components. All the parameters have been optimized using genetic algorithms.

In this thesis, it has been proposed a segmentation method that combines the benefits of using MSER, which globally extracts stable regions in the image, and a locally adaptive segmentation method, which is able to separate small characters in blurred images.

The proposed method has been trained using only one dataset and it has been tested on four different datasets showing that we achieve state-of-the-art performance in two of them in terms of f-measure, although we outperform other methods in terms of precision, thus meaning that our algorithm is the one which gets the lowest number of false positives. Moreover, in the other two datasets we outperform other state-of-the-art approaches. All this means that the study explained in section 3.1 and the proposed text location framework can be extrapolated to any situation and to any other dataset.

As future work, we are interested in improving the recall value of the detection by keeping the precision high. The addition of new features will be studied for this purpose. We also aimed at improving the segmentation method in order to reduce the number of character candidates and thus reducing the computational time of the algorithm, so it can be embedded into a mobile device such as a smartphone or a Tablet PC.

Chapter 4

Text Recognition in Natural Images

As it was stated in the previous chapter, the algorithm proposed in this thesis can be split into two main parts. The first one, aimed at detecting and locating text in natural images, was described in the previous chapter. The second part, which is described in this chapter, deals with the problem of recognizing the text detected in the first stage. Most of the state-of-the-art works in the field of text recognition in natural images are based on applying commercial OCRs over the bounding boxes where text has been detected. Commercial OCRs achieve an excellent performance when reading text of scanned images of documents, which are usually very contrasted images between the foreground and the background. However, natural images present a further difficulty, because the text is usually embedded in complex backgrounds, the contrast between foreground and background is very low and the text can have many different appearances in terms of size, style, color and layout. Some approaches have tried to face this problem, but most of them use their own datasets or are restricted to a limited number of font styles or even to only recognizing digits.

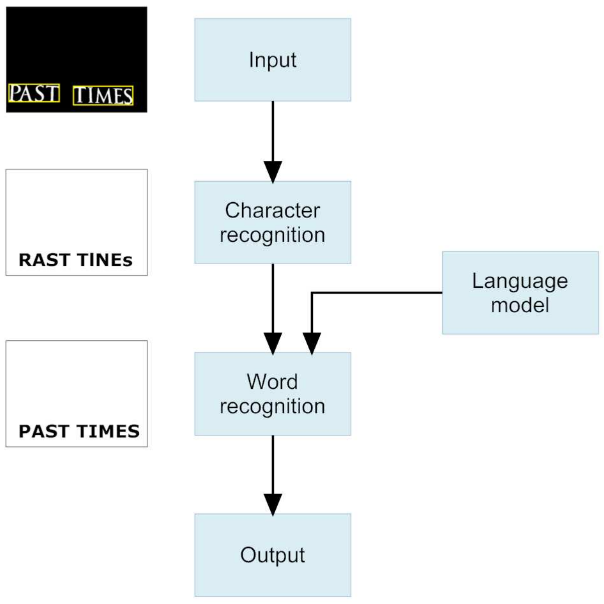


Figure 4.1: The flowchart of the text recognition algorithm

In this chapter, we present a text recognition algorithm for natural images that has been tested on the challenging ICDAR 2003 and ICDAR 2011 datasets. We simply restrict to recognize machine-printed text (not handwritten) in English language.

The flowchart of the text recognition algorithm is shown in figure 4.1. Initially, single characters are recognized using a classification approach based on K-Nearest Neighbors (KNN) and gradient direction features. Later, a unigram language model is applied in order to correct misspelled words.

The rest of the chapter is structured as follows. In first place, section 4.1 describes the approach developed to recognize single characters, with its corresponding experimental results and partial experiments. On the other hand, section 4.2 explains the method to recognize whole words and the corresponding experimental results. Finally, section 4.3 concludes the chapter.

4.1 Character recognition

4.1.1 Proposed approach

In this section we present the proposed approach to recognize single characters, which coincides with the typical framework used in the state of the art, as shown in figure 4.2. Given an input object, firstly some features are extracted on it and then these features are classified in order to identify the input object. Our main contributions to the state of the art for this process of recognizing single characters are, in first place, a new feature based on histogramming the gradient directions only at the boundary pixels of the input object, called Direction Histogram (DH), and, secondly, a fuzzy KNN-based approach to classify the extracted feature into different classes, providing a degree of membership for each class.

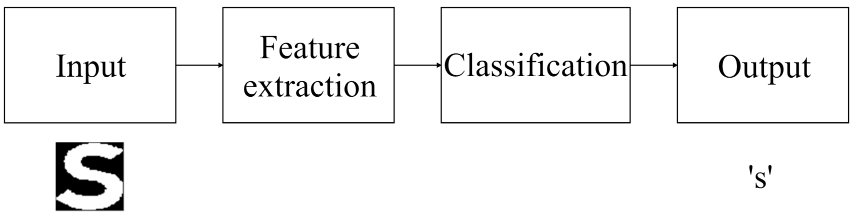


Figure 4.2: Character recognition flowchart

Characters in a word usually have the same texture properties and very similar geometrical characteristics. Therefore, features like those used in the previous chapter to distinguish between character and non-character components, such as aspect ratio, compactness, solidity or stroke width, among others (see table 3.5 in the previous chapter), are not a good solution to identify letters, as many of them have very similar values for all these features. Five examples are shown in tables 4.1 and 4.2. It can be seen that each pair of letters ‘B’-‘D’, ‘M’-‘N’, ‘O’-‘P’, ‘R’-‘S’ and ‘6’-‘g’ have very similar values of the mean for all the features, so, in principle, it would be impossible to distinguish the letters in each pair using the features proposed in the previous chapter.

The most distinctive feature to distinguish letters is the shape of the object. This idea has been proved by many authors. The work by [de Campos et al., 2009] is very relevant on this matter. They perform a comparison between two kind of descriptors: shape descriptors (Shape Context, Geometric Blur) and appearance descriptors (SIFT, Spin Image, Maximum Response of filters and Patch descriptor). The experimental results show that the two first descriptors (Shape Context and Geometric Blur) outperform the appearance descriptors when recognizing single characters in natural images.

Feature	Measure	‘B’	‘D’	‘M’	‘N’
Occupy rate	Mean	0.5804	0.5378	0.5077	0.5210
	Std	0.1271	0.1160	0.1389	0.1440
Compactness	Mean	0.0369	0.0391	0.0118	0.0151
	Std	0.0095	0.0105	0.0062	0.0066
Solidity	Mean	0.6374	0.6107	0.5426	0.5602
	Std	0.1308	0.1288	0.1443	0.1410
Occupy rate convex area	Mean	0.9089	0.8807	0.9361	0.9277
	Std	0.0365	0.0393	0.0543	0.0726
Stroke width size ratio	Mean	0.0947	0.1024	0.0958	0.1085
	Std	0.0294	0.0340	0.0399	0.0390
Max stroke width size ratio	Mean	0.1257	0.1324	0.1306	0.1510
	Std	0.0381	0.0455	0.0517	0.0472
Aspect ratio	Mean	1.391	1.3100	1.2001	1.2489
	Std	0.3993	0.2945	0.1747	0.2673
Stroke width variance ratio	Mean	0.2885	0.2519	0.3333	0.4228
	Std	0.2916	0.2432	0.2714	0.3161

Table 4.1: Mean and standard deviation of each feature per letter.

Feature	Measure	‘O’	‘P’	‘R’	‘S’	‘g’	‘6’
Occupy rate	Mean	0.4636	0.4946	0.5326	0.4967	0.5228	0.5080
	Std	0.1292	0.1222	0.1389	0.1101	0.1473	0.1261
Compactness	Mean	0.0391	0.0341	0.0277	0.0160	0.0233	0.0234
	Std	0.0112	0.0124	0.0094	0.0078	0.0092	0.0074
Solidity	Mean	0.5637	0.6470	0.5854	0.5722	0.5990	0.5934
	Std	0.1478	0.1418	0.1416	0.1229	0.1503	0.1599
Occupy rate convex area	Mean	0.8199	0.7608	0.9068	0.8684	0.8666	0.8599
	Std	0.0327	0.0499	0.0577	0.0481	0.0705	0.0369
Stroke width size ratio	Mean	0.0925	0.1064	0.0970	0.0953	0.0855	0.0771
	Std	0.0285	0.0368	0.0378	0.0321	0.0347	0.0187
Max stroke width size ratio	Mean	0.1183	0.1356	0.1244	0.1186	0.1147	0.1230
	Std	0.0388	0.0480	0.0466	0.0371	0.0510	0.0429
Aspect ratio	Mean	1.2611	1.4074	1.3783	1.5446	1.4885	1.5763
	Std	0.3324	0.3638	0.3272	0.3508	0.2553	0.3518
Stroke width variance ratio	Mean	0.2746	0.2431	0.2209	0.2792	0.2760	0.2633
	Std	0.2946	0.3042	0.2066	0.2419	0.2769	0.2029

Table 4.2: Mean and standard deviation of each feature per letter.

Typically, there are two tendencies to represent the shape of objects and classify them. The first one [Belongie et al., 2002; Tu et al., 2006] consists of representing the contour of an object as a set of points and performing shape matching with respect to a set of templates. The main problem of this kind of techniques is that a huge number of templates is needed to model each class, especially for natural images in which the number of different fonts and styles of writing can be very high, in order to achieve a reliable classification. The second trend [Jin and Geman, 2006; Neumann and Matas, 2010; Newell and Griffin, 2011] consists of extracting the orientation of the boundary points of the objects and representing the local symmetry, which closed contours are supposed to have, using histogramming or other kind of representation. The classification is typically carried out using machine learning techniques or nearest neighboring. In this thesis, we follow the second trend, as it seems to achieve better performance for natural images. However, instead of representing the local symmetry of the objects, which can be highly affected by the noise in the image, we propose to represent the local similarity between objects of the same class. For this purpose, we have defined a new feature, which we have named as Direction Histogram (DH), that is based on simple and fast-to-compute features. An overview can be seen in figure 4.3. We detect the boundary pixels of the connected components obtained in the detection stage (see previous chapter), after being resized to 64×64 pixels, and then we compute the direction of the gradient for each boundary pixel. As it is a binarised image, there is only gradient on the boundary pixels, so it is faster to compute. Later, we quantize the direction of the gradients in the boundary pixels into 8 bins: $\{-135°, -90°, -45°, 0°, 45°, 90°, 135°, 180°\}$, and we compute the histogram for each bin. The image is divided into 16 blocks in order to have spatial information, and the histograms for each block are concatenated into a 128-dimensional vector. It is important to remark that, as this method is based exclusively on the direction of the boundary pixels, it is not affected by color neither intensity. Later in this section we will show the robustness of the proposed feature compared to other widely used features.

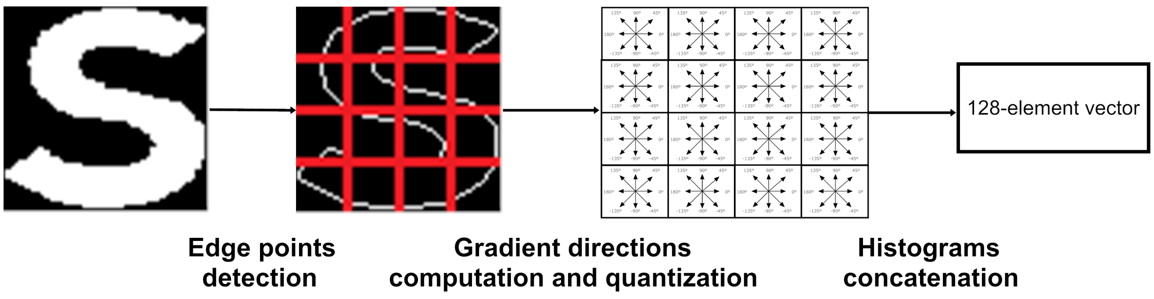


Figure 4.3: Feature computation

The classification is based on a KNN approach, but some contributions have been proposed, as it will be explained in the following paragraphs. We have chosen to use KNN as the basis of our proposal because of its simplicity compared to more complex machine learning technique such as SVM or neural networks. The training dataset is composed of 5438 character samples extracted from the training set of the ICDAR 2003 Robust Reading Competition dataset, which has a wide diversity of fonts. This dataset is asymmetric. In other words, there are different number of samples per letter. Table 4.3 shows the number of samples per letter.

Unlike a typical KNN approach that classifies the input sample into a single class, we suggest to use a fuzzy KNN-based classifier. What we propose is to give different solutions with output probabilities (that is, a membership value for each class), instead

Letter	Samples	Letter	Samples	Letter	Samples	Letter	Samples
A	197	Q	1	g	62	w	41
B	47	R	175	h	84	x	21
C	137	S	222	i	210	y	42
D	89	T	177	j	6	z	0
E	301	U	56	k	25	0	24
F	51	V	24	l	123	1	25
G	69	W	35	m	88	2	33
H	77	X	13	n	228	3	13
I	152	Y	41	o	246	4	19
J	14	Z	4	p	55	5	15
K	24	a	217	q	2	6	5
L	142	b	34	r	199	7	6
M	69	c	89	s	185	8	9
N	153	d	84	t	214	9	5
O	167	e	346	u	87	-	-
P	76	f	52	v	31	-	-

Table 4.3: Number of samples per letter in the training set.

of giving only one solution (that is, instead of predicting only one class for the input sample). This will be of interest to separate characters that are not possible to segment in the text detection stage, as it will be explained later in this section. We have discarded to use a Bayesian classifier, which could be another option to compute the likelihood of belonging to a certain class, because a Bayesian classifier would require to know the prior probabilities of each class, which can be different depending on the application and on the language, and the classifier should need to be adapted for each case. The classifier explained in this section is trained using the ICDAR 2003 dataset and the experimental results obtained with other datasets, shown later in this chapter, will prove that the proposed solution is valid and robust to many situations.

The classification is carried out as follows. Firstly, the nearest K neighbors in the training dataset of the character to be classified are extracted as a function of the distance between their feature 128-dimensional descriptors. Each neighbor belongs to a class, *i.e.* each neighbor votes for a certain candidate $S = \{s_1, s_2, \ldots, s_K\}$, where $s_i \in$ {'A', 'B', ..., 'Z', 'a', 'b', ..., 'z', '0', ..., '9'} (62 classes). The set of distances from the object to each neighbor is $D = \{d_1, d_2, \ldots, d_K\}$. Figure 4.4 shows all these definitions. We define the ratio between each distance to the minimum one as in (4.1).

$$R = \{r_1, r_2, \ldots, r_K\} = \{1, \frac{d_1}{d_2}, \ldots, \frac{d_1}{d_K}\} \tag{4.1}$$

As d_1 is the shortest distance, each ratio $\frac{d_1}{d_i}, i = 2 \ldots K$ are less or equal to 1. We define p_1 as the output probability of the nearest neighbor. We assume that the output probabilities of the following $K-1$ nearest neighbors are related to p_1 by the distance ratios defined in (4.1). Therefore, it must be fulfilled that the sum of the probabilities given by each neighbor is 1, as shown in (4.2).

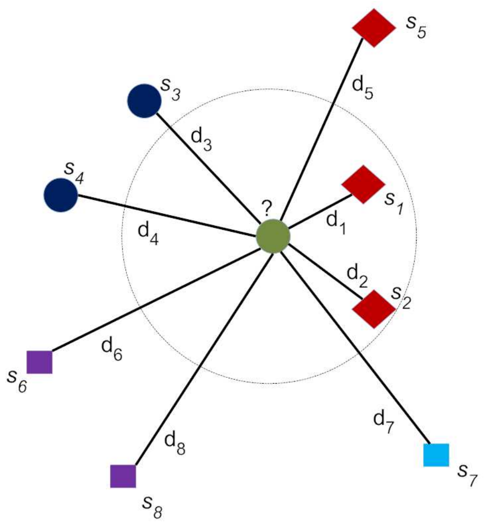


Figure 4.4: Classification method

$$\sum_{i=1}^{K} r_i \cdot p_1 = p_1 + \frac{d_1}{d_2} \cdot p_1 + \ldots + \frac{d_1}{d_K} \cdot p_1 = 1 \tag{4.2}$$

The value of p_1 can be easily computed from (4.2). The output probabilities of the object for every class can be computed using (4.3). Equation (4.3) means that the probability of the object of belonging to class 'A' is computed only from the neighbors that correspond to this class. The same is done for class 'B', 'C' and so on.

$$\begin{aligned} p_A &= \sum_{j=1}^{K} r_j \cdot p_1 \qquad \forall j \ s.t. \ s_j = A \\ p_B &= \sum_{j=1}^{K} r_j \cdot p_1 \qquad \forall j \ s.t. \ s_j = B \\ &\vdots \\ p_9 &= \sum_{j=1}^{K} r_j \cdot p_1 \qquad \forall j \ s.t. \ s_j = 9 \end{aligned} \tag{4.3}$$

With this method, when the object to be recognized is clearly a certain letter, there are many minima that vote for the same class, thus it will have a high output probability for that class. When it is not a clear case, the highest output probability tends to be low, and the worst case would be when each neighbor is at the same distance and votes for different classes, thus there would be K outputs with comparable probability. Therefore, a compromise in the value of K must be found. A low value for K could be insufficient to have reliable output probabilities, but a high value could lead to errors, as the solutions with highest output probabilities would tend to those classes with a bigger number of samples. In our case, in which the training dataset is asymmetric, *i.e.* there are classes with a number of elements much higher than other classes, the number of nearest neighbors K has been set empirically to 25.

As the feature proposed is a distribution represented by histograms, it is natural to use the χ^2 test statistic, as it is shown in [Belongie et al., 2002]. Therefore, the distances in the classification are computed using (4.4), where h and h_i denote the 128-bin normalized histogram for the object to be recognized and the object i in the training set respectively.

$$d_i = \frac{1}{2}\sum_{k=1}^{128} \frac{[h(k) - h_i(k)]^2}{h(k) + h_i(k)} \tag{4.4}$$

4.1.2 Experimental results

In order to state the robustness of the proposed feature DH, the main local features of the state of the art are evaluated:

- Shape Context (SC) [Belongie et al., 2002] is a descriptor for point sets and binary images. Points in the boundary of the objects are extracted using the Sobel edge detector. The descriptor is a log-polar histogram, which gives a $\theta \times n$ vector, where θ is the angular resolution and n is the radial resolution. We use $\theta = 15$ and $n = 4$, as other authors have proposed for character recognition [de Campos et al., 2009].

- Geometric Blur (GB) [Berg et al., 2005] is a feature extractor with a sampling method similar to that of SC, but instead of histogramming points, the region around an interest point is blurred according to the distance from this point. For each region, the edge orientations are counted with a different blur factor.

- Scale Invariant Feature Transform (SIFT) [Lowe, 1999] are extracted on keypoints located by the Harris Hessian-Laplace detector, which gives affine transform parameters. The feature descriptor is computed as a set of orientation histograms on 4×4 pixel neighborhoods. The orientation histograms are relative to the keypoint orientation. The histograms contain each 8 bins, and each descriptor contains a 4×4 array of 16 histograms around the keypoint. This leads to feature vector with 128 elements.

- Speeded Up Robust Feature (SURF) [Bay et al., 2008] is inspired by the SIFT descriptor but it uses the sum of the Haar wavelet response around the points of interest with the aid of integral images.

- Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] counts occurrences of gradient orientation in blocks of an image.

- Local Binary Patterns (LBP) [Ojala et al., 2002] is a texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

Table 4.4 shows the character recognition rate using each kind of feature. Three cases have been analysed. The first one only takes into account the hit rate for the output class with highest probability. The second analysis computes the hit rate for those cases in which the recognition succeeds for either the first or the second solution. Similarly it is done for the first, second and third candidates. It can be clearly seen that DH is the best feature and this method successfully recognizes more than 90% of characters as first or second solution. For the subsequent word recognition applied after the character recognizer, which we will explain the following section, we are only taking into account the first candidate, which is correct for the 76.3% of the cases using the proposed feature. As future work, we intend to use the different candidates with their corresponding probabilities given by the character recognizer in order to make a more robust word recognizer that takes into account these likelihood values. That is the reason why we wanted to show in this section the importance of taking the two or even the first three candidates instead of choosing only the first one.

On the other hand, figures 4.5-4.7 show the character recognition rate as a function of the training dataset size. It can be seen that the hit rate for DH feature tends to an asymptote for a training dataset size of 2000 samples, while the asymptote for other features is reached for a major number of samples. This means that the DH feature requires a lower number of training samples than other features to achieve the same performance.

Features	Hit rate 1st candidate	Hit rate 1st/2nd candidate	Hit rate 1st/2nd/3rd candidate
DH	**76.3%**	**91.4%**	**95.6%**
LBP	67.5%	82.7%	90.0%
SC	59.6%	77.0%	83.4%
SIFT	58.9%	66.8%	68.4%
GB	56.1%	70.1%	75.4%
SURF	52.2%	64.0%	70.2%
HOG	48.8%	66.8%	75.4%

Table 4.4: Individual character recognition on ICDAR 2003 dataset.

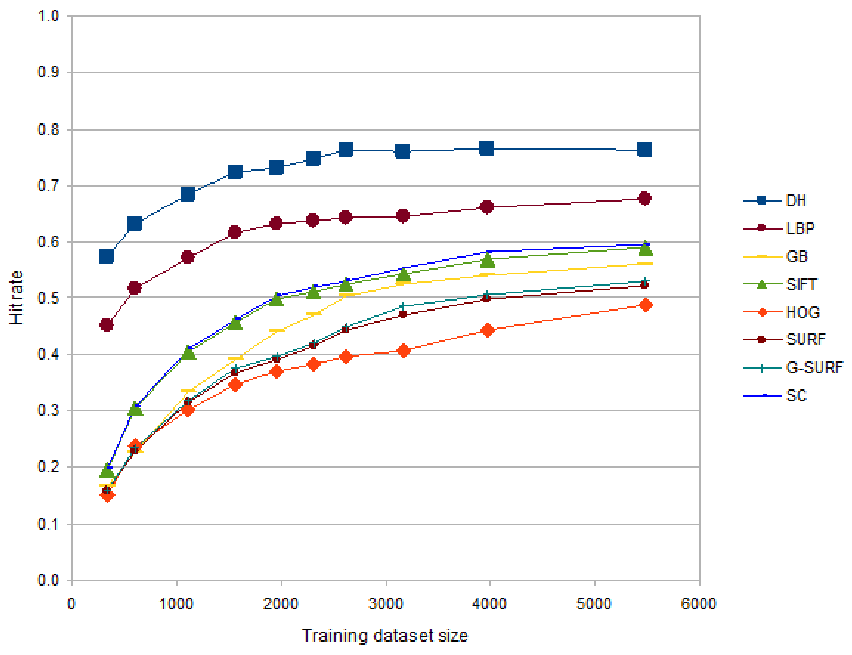


Figure 4.5: Recognition rate vs Training dataset size (1^{st}*cand.*)

The proposed method has been evaluated on the ICDAR 2003 test dataset, which contains more than 5000 letters in 250 pictures. We compare our approach to [Coates

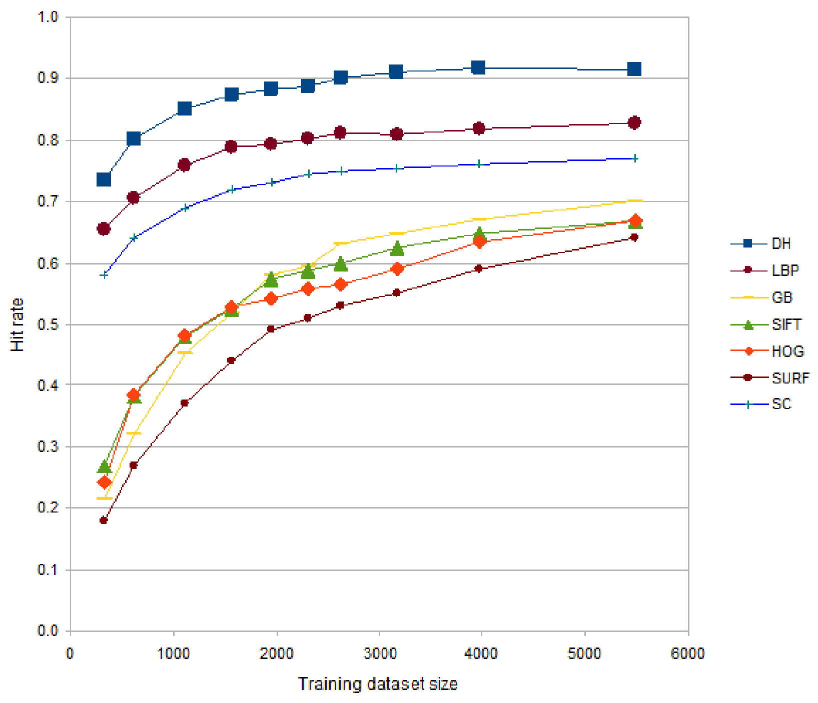


Figure 4.6: Recognition rate vs Training dataset size ($1^{st}/2^{nd}cand.$)

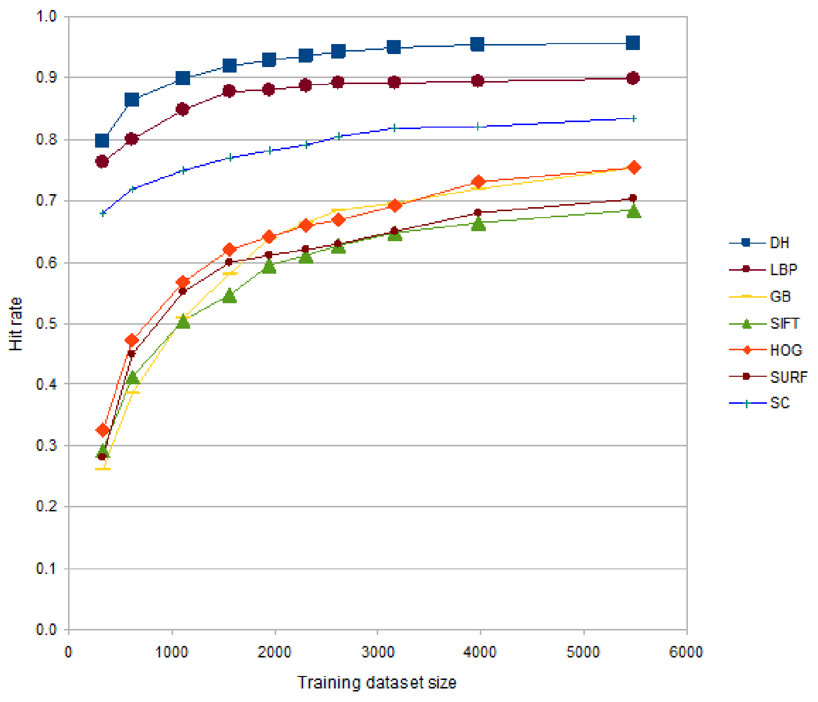


Figure 4.7: Recognition rate vs Training dataset size ($1^{st}/2^{nd}/3^{rd}cand.$)

et al., 2011; Chan and Pun, 2011; Newell and Griffin, 2011; Neumann and Matas, 2012] methods, which were tested with the same dataset. The approach proposed by [Zhu et al., 2012] was also tested with the same dataset, but only with a subset of the whole set, so their results are not comparable. Table 4.5 shows the comparison of our method to the state-of-the-art approaches. Since table 4.4 does not take into account the number of non-detected objects, we have incorporated the non-detection rate in table 4.5 in order to make a fair comparison with [Neumann and Matas, 2012], because there is not any information available on the mismatched and non-detection rates for the other techniques. It can be seen that we perform in second place, but we get the same hit rate than [Coates et al., 2011] if we take into account the second candidate for this analysis. However, the work by [Coates et al., 2011] uses a feature vector of 1500 elements, while the method proposed in this thesis only needs 128 elements. The mismatched rate for the first two candidates is reduced almost to one third of the mismatched rate with only one candidate and it is much lower than the Neumann's mismatched percentage. Actually, it has been observed that there is a set of pairs and threes of letters that cannot be differentiated between upper-case and lower-case: {'Cc', 'Iil', 'Jj', 'Oo', 'Pp', 'Ss', 'Uu', 'Vv', 'Ww, 'Xx', 'Zz'}. The only way to distinguish these letters in their upper-case and lower-case variants is to use as reference the height of the other unambiguous letters in the same line. In principle, we are just interested in character recognition in a raw way, but if we compute the character recognition rate joining both classes of the undistinguishable letters as only one class for each pair, we get the results shown in table 4.6. It can be clearly noticed that the hit rate for the first candidate greatly increases, as it achieves a matched rate higher than 80% and the mismatched rate reduces to 9%, achieving state-of-the-art performance.

Algorithm	Matched	Mismatched	Not found
[Coates et al., 2011]	81.7%	N/A	N/A
[Neumann and Matas, 2012]	67.0%	12.9%	20.1%
[Chan and Pun, 2011]	56.0%	N/A	N/A
[Newell and Griffin, 2011]	52.7%	N/A	N/A
Our method (1st candidate)	68.2%	21.2%	10.6%
Our method (1st/2nd cand.)	81.7%	7.7%	10.6%
Our method (1st/2nd/3rd candidate)	85.4%	4.0%	10.6%

Table 4.5: Individual character recognition on ICDAR 2003 dataset.

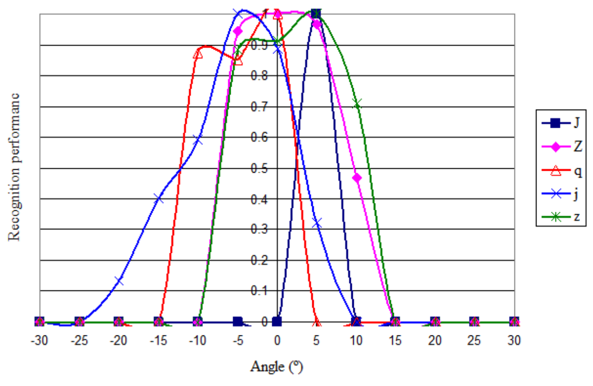
4.1.3 Study on rotations

The approach proposed in this paper aims at detecting horizontal text, as it is the most common layout in English and the rest of western languages. However, our approach is able to detect text even when slight deviations and rotations respect to the horizontal axis take place. In terms of text detection, it has been seen that the proposed approach is able to correctly detect words that have up to 30° of deviation respect to the horizontal axis. However, in terms of character recognition, the performance depends on the class (letter or number). A study on how rotations affect the accuracy of the character recognizer

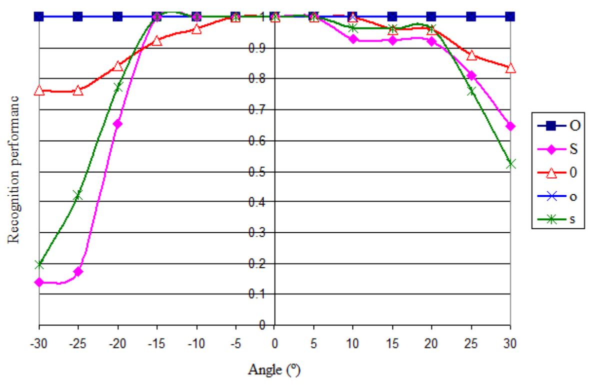
Algorithm	Matched	Mismatched	Not found
Our method (1st candidate)	80.4%	9.0%	10.6%
Our method (1st/2nd cand.)	84.1%	5.3%	10.6%
Our method (1st/2nd/3rd candidate)	85.8%	3.6%	10.6%

Table 4.6: Individual character recognition on ICDAR 2003 dataset, taking indistinguishable pairs of letters as one class for each pair.

is presented below. In general the error committed when recognizing letters increases as the rotation grows up. However, the degradation is bigger for some letters than others. Specifically, among the 62 classes (52 letters and 10 numbers), those that suffer a faster degradation are the following: ‘J’, ‘Z’, ‘q’, ‘j’ and ‘z’. As it is shown in figure 4.8(a), the accuracy of the recognition for these letters falls below 30% from 5° or 10° upwards. On the other hand, those classes that hardly suffer degradation are ‘O’, ‘S’, ‘0’, ‘o’ and ‘s’. The recognition accuracy remains above 80% even when the rotation angle reaches 25° or 30°, as shown in figure 4.8(b).



(a) Classes with more degradation

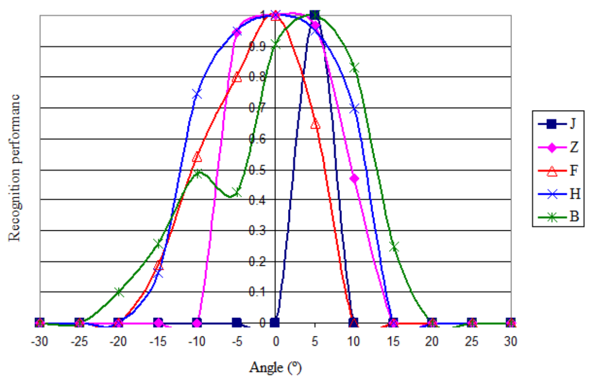


(b) Classes with less degradation

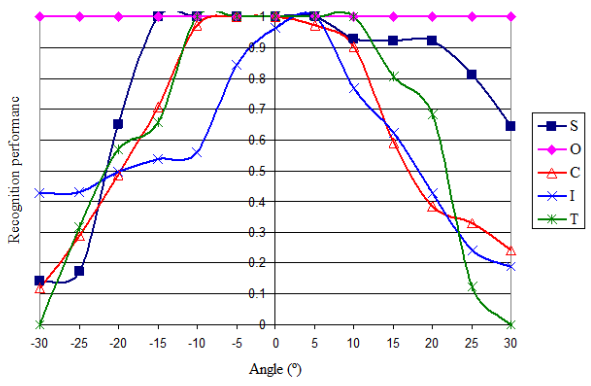
Figure 4.8: Effect of rotations on character recognition

Figure 4.9 shows how rotations affect the 5 upper-case letters that suffer more degradation (‘J’, ‘Z’, ‘F’, ‘H’ and ‘B’) and the 5 upper-case letters that have less degradation (‘S’, ‘O’, ‘C’, ‘I’ and ‘T’). Similarly, figure 4.10 shows the 5 more (‘j’, ‘n’, ‘u’, ‘q’ and ‘z’) and less degradated (‘o’, ‘s’, ‘c’, ‘i’ and ‘l’) lower-case letters when rotations are present.

Finally, figure 4.11 shows how rotations affect the recognition for the 10 digits. It can be seen that ‘0’ and ‘1’ are the numbers whose recognition performance is less affected by rotations. On the other hand, the degradation of the performance for the rest of the numbers depend on the direction of the rotation (clockwise or counterclockwise). If the rotation is clockwise (positive angle), classes ‘5’ and ’8’ suffer less degradation than if the rotation is on the opposite direction (negative angle). However, classes ‘2’, ‘3’, ‘4’, ‘7’ and ‘9’ are less affected by rotations on the counterclockwise direction (negative angle).

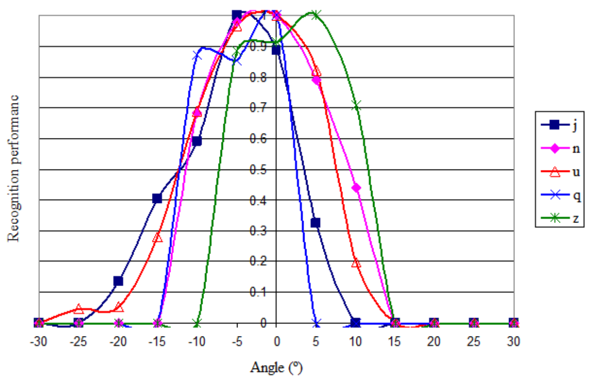


(a) Upper-case letters with more degradation

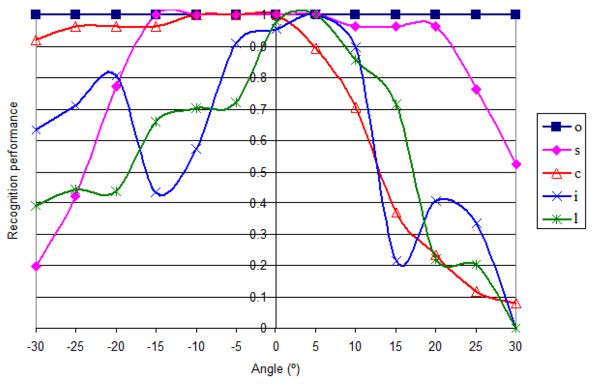


(b) Upper-case letters with less degradation

Figure 4.9: Effect of rotations on upper-case letter recognition



(a) Lower-case letters with more degradation



(b) Lower-case letters with less degradation

Figure 4.10: Effect of rotations on lower-case letter recognition

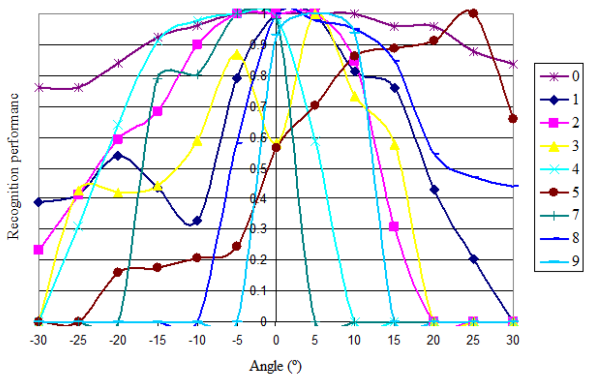
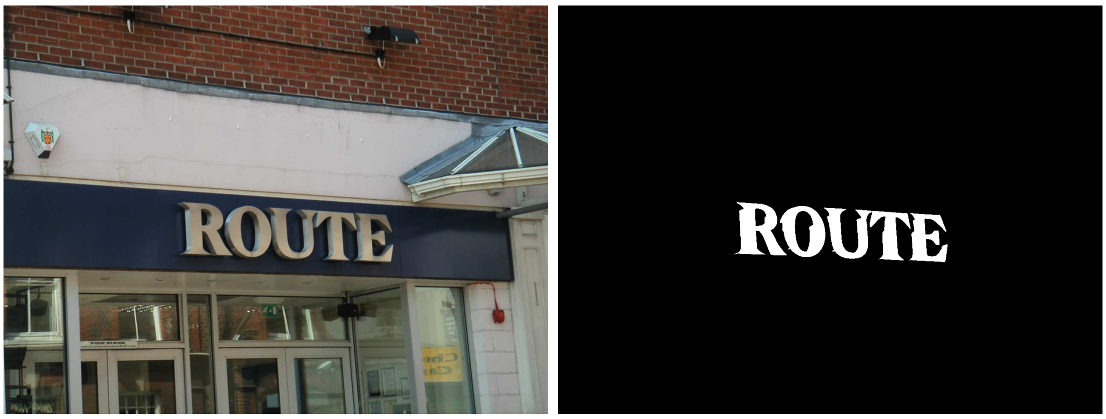


Figure 4.11: Effect of rotations on number recognition

4.1.4 Character separation

We have seen in the previous subsections that the proposed character recognition approach is based on a fuzzy classifier that, given a certain input object to be recognized, is able to classify the input sample into different letters with a certain membership value for each class. We have mentioned before that a possible application of these membership values is in the subsequent word recognizer that will be explained in the next section. However, firstly we show here an alternative application of these values. We propose to use the output probabilities to split those characters that are not possible to separate in the segmentation step. From out knowledge, there is not any work in the state of the art that has dealt with this problem using a similar approach like the one proposed here using the probabilities of the objects of belonging to each class.

Figure 4.12 shows an example where the objects U and T are not separated in the segmentation step because they are 8-connected in the binary image. Therefore, initially only 4 objects (R, O, UT, E) are detected. The first candidates and the output probabilities of the first candidate for each object are, respectively, 'R' with $p = 0.97$, 'O' with $p = 1.0$, 'M' with $p = 0.42$ and 'E' with $p = 0.74$. It can be clearly seen that the third object, which has not been correctly segmentated, has a lower probability respect to the others. It suggests that something is wrong with it. We have developed an algorithm that uses this evidence together with others, in order to deal with this kind of situations by separating the objects that are highly likely to be wrongly connected.



(a) Source image (b) Segmented image

Figure 4.12: Segmented image.

The following conditions must be fulfilled to split up an object (we recommend to see the whole algorithm summarized in algorithm 4.1):

1. The projection on the horizontal axis of all the pixels that belong to the object is computed. An example of two join characters in an 8-connected object is shown in figure 4.13(a), while the projection on the horizontal axis is displayed in blue in figure 4.13(b). The maximum max_{ver} and the minimum min_{ver} values of the projection in the area delimited by the red dashed lines in figure 4.13(b), which correspond to the 35% and 65% of the width of the object, are computed. The minimum falls in a horizontal coordinate x_{min}. If the minimum value is equal or lower than the 35% of the maximum one ($min_{ver} \leq 0.35 max_{ver}$) and there is one only stroke that joins the object pixels to the left and to the right of x_{min} (the object is made by two

components joined with only one stroke), then the next condition is checked. If not, the object cannot be separated.

2. The object is compared to the rest of characters of the same word in the following way. If the output probability of the first candidate P_i of the object under review is less than 0.7 times the output probability of the first candidate P_j ($P_i \leq 0.7P_j \; \forall j \neq i$) and the width W_i is greater than 1.82 times the width of the other character in the same word W_j ($W_i \geq 1.82W_j \; \forall j \neq i$), only if this other character is not a thin letter, such as 'i', 'j', 'f', 'l', 'I', 'r' and 't', and these conditions are fulfilled for at least the half of the total number of characters in the word, then the object is split into two parts. The separation is carried out at the coordinate x_{min} previously computed. The classes and the output probabilities P_{i1} and P_{i2} of the new two objects are computed. The mean output probability P_m of the probabilities of the first candidates of the rest of characters in the same word, is also computed. The division is validated if $P_{i1} > 0.7P_m$ or $P_{i2} > 0.7P_m$. If not, the separation is not carried out.

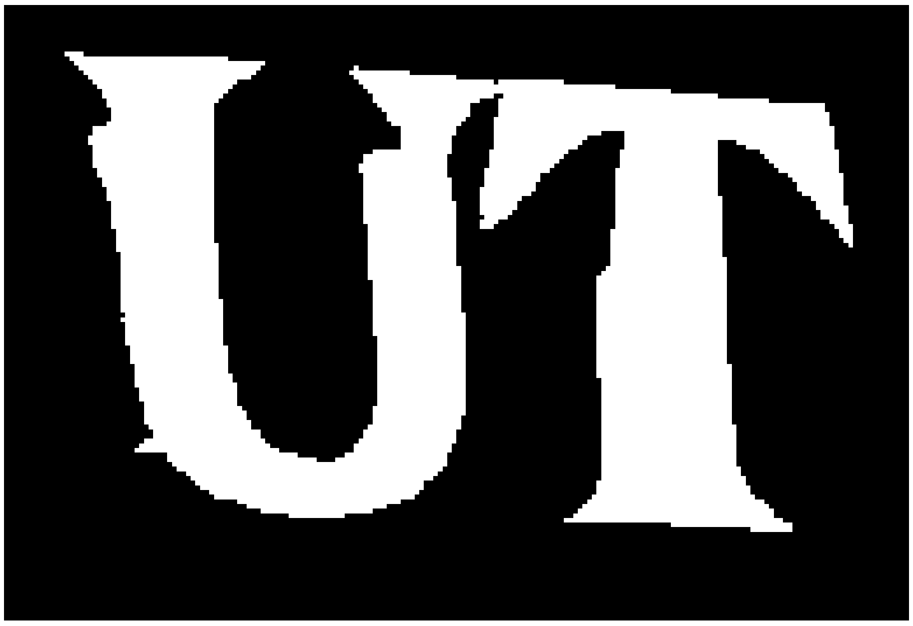
These steps are done iteratively for all the objects in a word until no more objects can be split up.

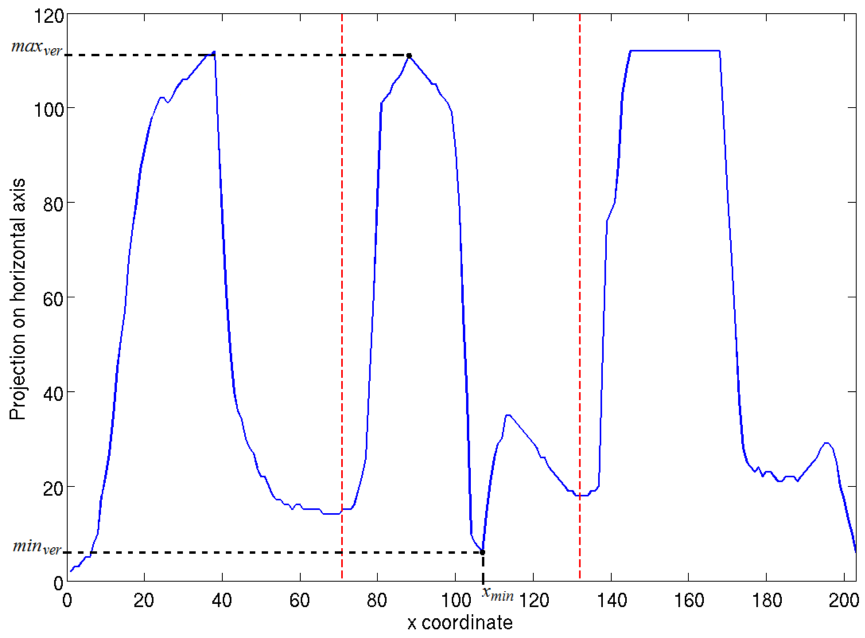
After applying this method, we are able to solve situations like the one shown in figure 4.12, where the first candidate for each object would be: 'R' with $p = 0.97$, 'O' with $p = 1.0$, 'U' with $p = 0.61$, 'T' with $p = 1.0$ and 'E' with $p = 0.74$, respectively. However, the problem of having erroneously split characters is also dealt in the word recognition step, where hypothesis of having wrongly separated objects are made and a set of candidates for each word is served as input into the word recognizer, as it will be explained in section 4.2.

Some other examples are shown in figures 4.14 and 4.15. Figures 4.14(c)-4.14(d) show how the developed algorithm is able to separate the letters 'R' and 'E' in "FIRED" and 'A' and 'R' in "EARTH". Similarly in figures 4.14(e)-4.14(f), 'R' and 'A' are split up in the word "NATURAL". Three letters are separated in the word "COLCHESTER" in figures 4.15(a)-4.15(b) and two letters in "BOROUGH". A very interesting example is shown in figures 4.15(c)-4.15(d), where letters 'S' and 'T' in "STAR" and 'R' and 'S' in "WARS" are impossible to be segmented as they are written as if they were one letter, but the developed method is able to split them up.

Algorithm 4.1 Character separation

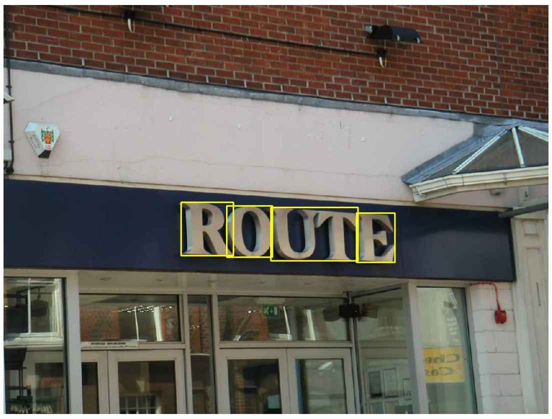
```
1: Let $N$ be the number of character components in a word $Z$, $O = \{O_1, \ldots, O_N\}$ be the set
   of objects in $Z$, $W_i$ be the width of $O_i$ and $P_i$ be the probability of the first candidate of $O_i$

2: procedure CHARACTER SEPARATION
3:     for all object $O_i$ in $Z$ do
4:         repeat
5:             Compute projection of the pixels of $O_i$ on the horizontal axis
6:             Compute maximum $max_{ver}$ and minimum $min_{ver}$ in $0.35W_i \leq x \leq 0.65W_i$.
   Minimum is in $x_{min}$
7:             if $min_{ver} \leq 0.35max_{ver}$ and only one stroke joins the object pixels to the left and
   to the right of $x_{min}$ then
8:                 for all $O_j$ in $Z$, $j \neq i$ do
9:                     Compute $\frac{P_i}{P_j}$ and $\frac{W_i}{W_j}$
10:                end for
11:                if $\frac{P_i}{P_j} \leq 0.7$ and $\frac{W_i}{W_j} \geq 1.82$ for at least $\frac{N}{2}$ objects then
12:                    Divide $O_i$ in two parts at $x_{min}$
13:                    Compute the classes and probabilities $P_{i1}$ and $P_{i2}$ of the new objects
14:                    Compute $P_m = mean\{P_j\}, \forall j \neq i$
15:
16:                    if $P_{i1} > 0.7P_m$ or $P_{i2} > 0.7P_m$ then
17:                        Validate separation
18:                    else
19:                        Do not separate
20:                    end if
21:                end if
22:            else
23:                Do not separate
24:            end if
25:        until $O_i$ cannot be separated any more
26:    end for
27: end procedure
```


(a)

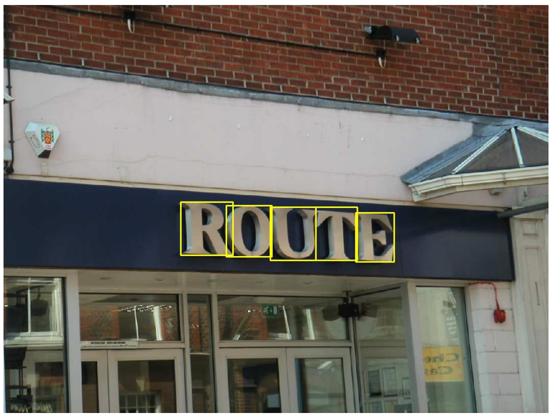


(b)

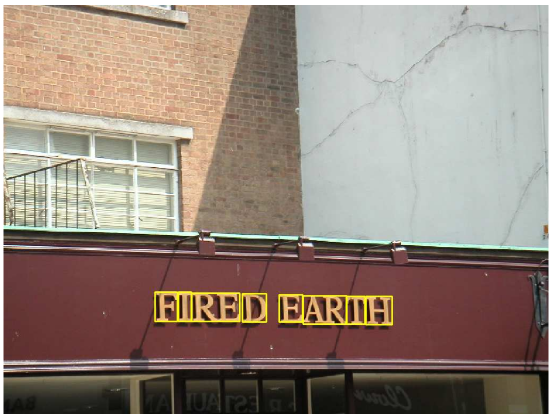
Figure 4.13: Projection of the object pixels on the horizontal axis (continuous)



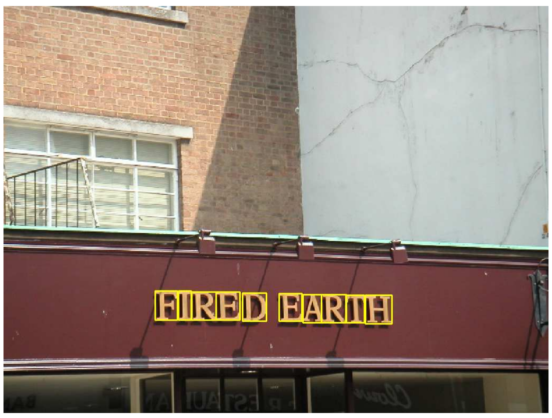
(a) Before character separation



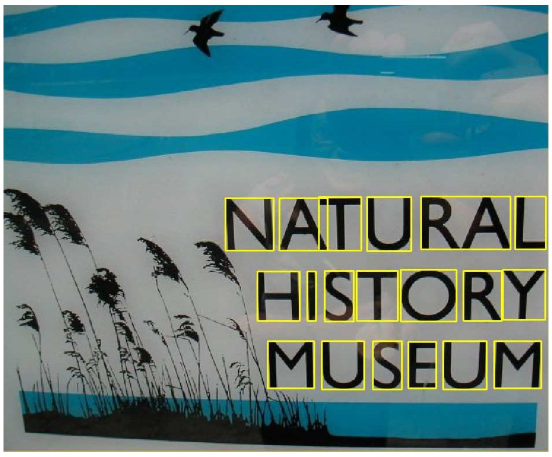
(b) After character separation



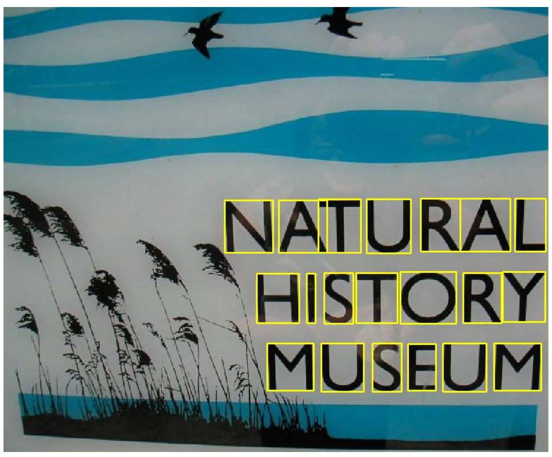
(c) Before character separation



(d) After character separation

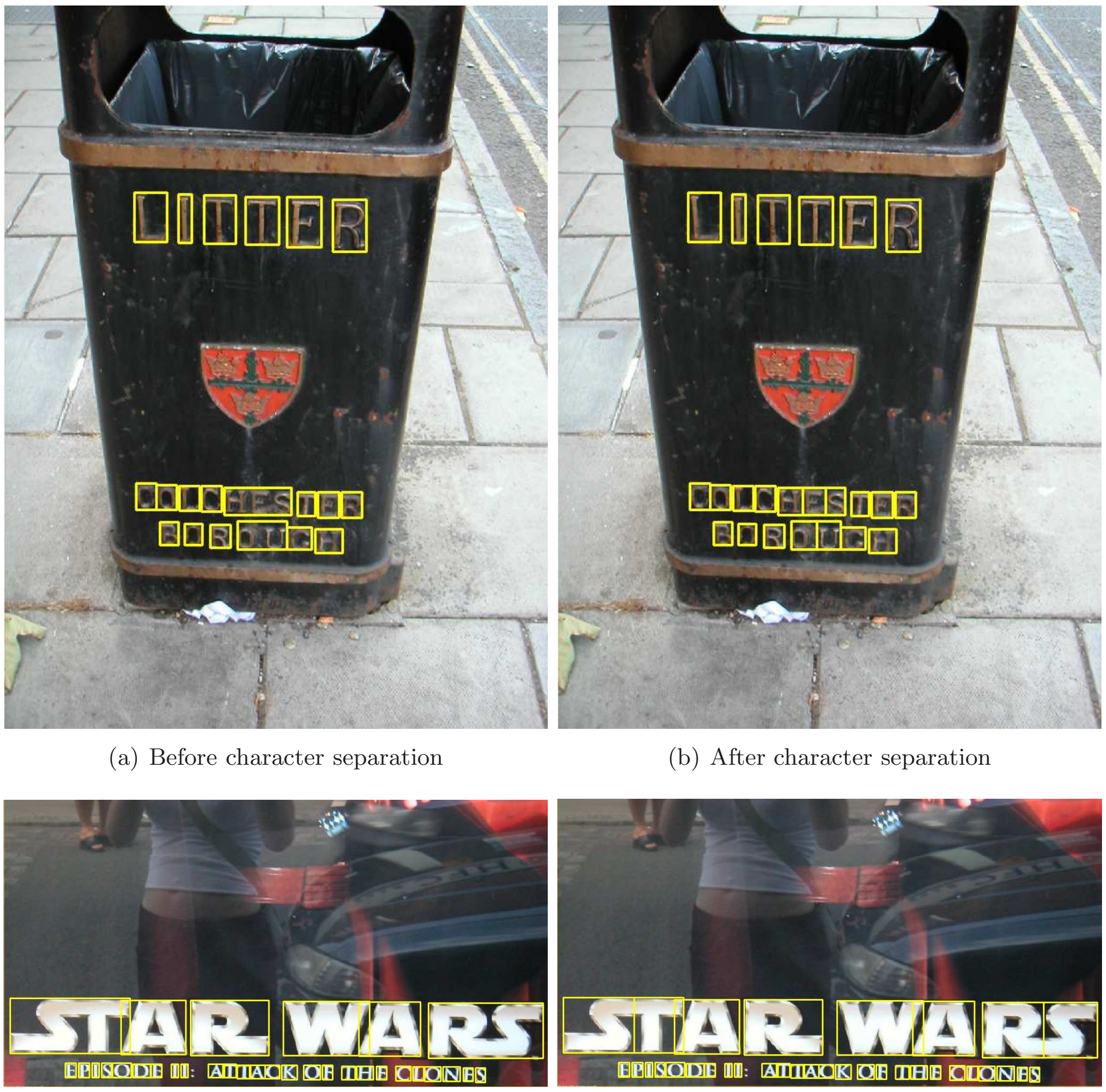


(e) Before character separation



(f) After character separation

Figure 4.14: Character separation.



(a) Before character separation

(b) After character separation

(c) Before character separation

(d) After character separation

Figure 4.15: Character separation.

4.2 Word recognition

4.2.1 Proposed approach

The character recognizer can erroneously classify some objects, thus leading to words that do not exist or are unlikely to be found under certain circumstances. Some authors have tried to solve this problem using different approaches. [Weinman and Learned-Miller, 2006] assume certain a priori knowledge of the language with a bigram model, which takes into account the likelihood of appearing two certain letters together, and letter case to improve recognition accuracy in context as English language rarely switches case in the middle of a word. This method is able to correct typos at syllable level but it does not correct whole words, thus non-existing words may occur.

On the other hand, [Neumann and Matas, 2010] use a typographic model to correctly differentiate between upper-case and lower-case variants of certain letters, such as 'C' and 'c' or 'P' and 'p', which are ambiguous without knowing the heights of other letters in the text line. In addition, they propose to use a language model at text line level that computes the most likely sequence in a text line. Given an alphabet A, a word $w = a_1a_2a_3\dots a_n, a_i \in A$ and a set of words in a dictionary W, a word score $s(w)$ is defined as in (4.5).

$$s(w) = \begin{cases} 1 & w \in W \\ \sqrt[n]{\prod_{i=1}^{n-1} P(a_i, a_{i+1})} & w \notin W \end{cases} \tag{4.5}$$

The probability $P(a_i, a_{i+1})$ is estimated using the relative frequency of the sequence (a_i, a_{i+1}) in the dictionary W. Given a text line $t = w_1w_2\dots w_n$, the text line score $S(t)$ is defined as in (4.6).

$$S(t) = \sqrt[n]{\prod_{i=1}^{n} s(w_i)} \tag{4.6}$$

Given a set T of hypothesis, the one with the highest score $S(t)$ is selected.

However, this approach has many unclear points. First of all, trying to find the most likely sequence in a text line should be equivalent to use a N-gram model, which assigns a probability to observe the word w_N in the context history of the preceding $N-1$ words. However, what the authors propose is to find the sequence of words that maximize (4.6), which is not the same as they do not use any model that takes into account the relative frequencies of observing N words consecutively. Therefore, it makes no sense to find the sequence of words that maximizes (4.6) if each word is considered as an independent observation. For this reason, it is not assured that finding the optimum solution as the maximization of the text line score $S(t)$ implies the maximization of each word score $s(w)$ independently, thus incorrect solutions can be given at word level.

In this thesis, we propose to use an unigram language model which simply constrains the output of the OCR to a set of meaningful words, because we aim at recognizing words instead of sequences in a text line, unlike [Neumann and Matas, 2010], as in our context we only have isolated words. An unigram language model only calculates the probability of hitting an isolated word, without considering any influence from the words before or after the target. The model that we propose is based on the British National Corpus (BNC) [Oxford, 2007], which is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of British English from the later part of the 20th century, both spoken and written. The latest edition was updated in 2007. The written part of the BNC (90%) includes extracts from newspapers, journals for all ages and interests, academic books and fiction, published and unpublished letters and memoranda, school and university essays, among many other kinds of text. On the other hand, the spoken part (10%) consists of ortographic transcriptions of unscripted informal conversations (recorded by volunteers selected from different age, region and social classes in a demographically balanced way) and spoken language collected in different contexts, ranging from formal business or government meetings to radio shows and phone-ins. Therefore, the corpus includes many different styles and varieties, and it is not limited to any particular subject field, genre or register. Moreover, it is updated as it covers English of the late 20th century, rather than the historical development which produced it. It deals with modern British English, but non-British English and foreign language words occur in the corpus. A key aspect is that the corpus includes the number of times that each word occurs, which is essential to compute the prior probability for each word and to build our language model.

Let denote z as a noisy detection of some unknown word w, which $w \in W$, where W is the set of all possible terms (in our case, the BNC). The most likely word w_{MAP} that could have generated z is the one that maximizes the a posteriori probability $p(w|z)$, as shown in (4.7).

$$w_{MAP} = \arg\max_{w \in W} p(w|z) \tag{4.7}$$

Applying the Bayes rule, (4.7) can be expressed as in (4.8).

$$w_{MAP} = \arg\max_{w \in W} \frac{p(z|w)p(w)}{p(z)} \tag{4.8}$$

In (4.8), $p(z)$ is identical for all words w in the set W. Therefore, the denominator can be dropped as we are trying to find the maximum, thus (4.8) reduces to (4.9).

$$w_{MAP} = \arg\max_{w \in W} p(z|w)p(w) \tag{4.9}$$

$p(w)$ is the prior probability of the word w occuring in a scene. We use word frequencies obtained from the BNC. On the other hand, $p(z|w)$ is the likelihood of the OCR returning a sequence z when the underlying word is w. This probability is computed using Dynamic Programming (DP), especifically the forward algorithm [Rabiner, 1989]. The idea behind DP consists of transforming one sequence into another one using edit operations that replace, insert or remove an element of the input sequence. Each operation has an associated cost. The goal is to find the optimum sequence, that is, the one with the lowest associated total cost. Here we work with probabilities instead of using costs, thus the optimum sequence would be the one with the highest associated probability.

If we assume that a misspelled word z has a set of of letters $z = z_1, z_2, ..., z_m$ and the correct word w has a set of letters $w = w_1, w_2, ..., w_n$, DP aims at computing the probability $p(z|w)$ of recognizing the sequence z given w. The way of modeling it is by using a confusion matrix, which says, for any given pair of letters, how likely a particular edit is to happen. Therefore, there are three confusion matrices: the substitution matrix, the insertion matrix and the deletion matrix. For instance, for the pair of letters x and y, the substitution matrix $sub[x, y]$ keeps the count of how often x is recognized as y. On the other hand, the insertion matrix $ins[x, y]$ keeps the count of how often y is inserted after x, while the deletion matrix $del[x, y]$ keeps the count of how often y is removed after x. Therefore, insertion and deletion are conditioned on the previous character. In our case, we propose a different approach that, from our knowledge, has never been used. As we do not have enough reliable data to compute the insertion and deletion matrices, we do not use them, but only the substitution matrix. We deal with deletions and insertions in a different way, using a "split and merging" process that will be explained later in this subsection. For the moment, we are going to explain how $p(z|w)$ is computed given an observed sequence z and a correct word w, both of the same length m, using the substitution matrix.

Firstly, the substitution matrix is computed from the OCR confusion matrix $C[x, y]$, which is shown in figure 4.16. This matrix keeps the count of the number of times that character x is recognized as y. In principle, the substitution matrix $sub[x, y]$, which keeps the probability that a certain character x is recognized as y, would be obtained by normalizing each row in $C[x, y]$. However, many elements in $C[x, y]$ are equal to zero, and we do not want to assign zero probabilities, because a zero probability for a certain substitution would mean that the substitution is not possible to happen. In order to avoid this, the substitution matrix is computed by applying Add-1 smoothing to the confusion matrix, as shown in (4.10), where V is the number of classes, *i.e.* the number of characters ($V = 52$).

$$sub[x, y] = \frac{C[x, y] + 1}{\sum_{y=1}^{V} C[x, y] + V} \tag{4.10}$$

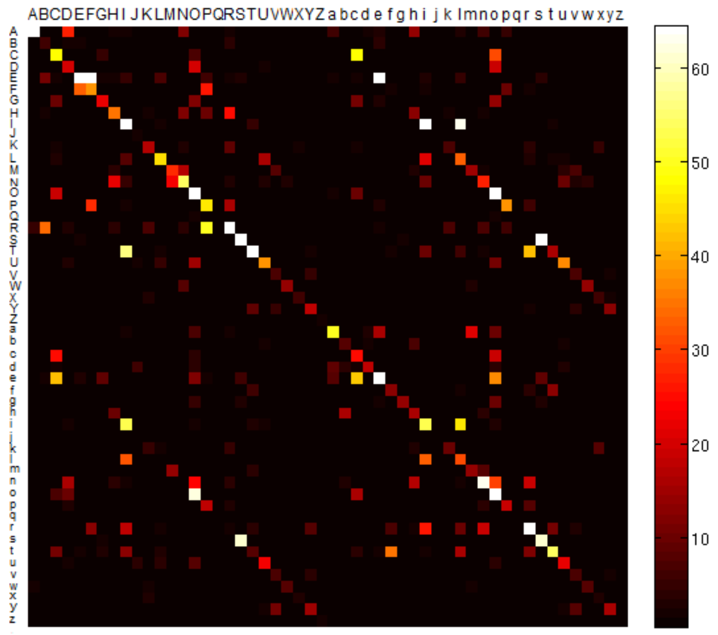


Figure 4.16: Confusion matrix of the character recognizer

Table 4.7 shows a small part of the OCR confusion matrix, while table 4.8 shows the substitution matrix once the Add-1 smoothing has been applied.

Therefore, $p(z|w)$ is computed in the following way. Given an observed word $z = z_1, z_2, ..., z_m$, whose length is m, and a word $w = w_1, w_2, ..., w_m$ of the same length,

-	A	B	C	D	E	F	$\cdots$	z
A	65	0	0	27	0	0	$\cdots$	0
B	0	6	0	2	2	0	$\cdots$	0
C	0	0	48	0	0	0	$\cdots$	0
D	0	0	1	20	0	0	$\cdots$	0
E	1	11	2	1	108	89	$\cdots$	0
F	0	0	0	0	33	38	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
z	0	0	0	0	0	0	$\cdots$	0

Table 4.7: OCR confusion matrix.

-	A	B	C	D	E	F	$\cdots$	z
A	0.3128	0.0047	0.0047	0.1327	0.0047	0.0047	$\cdots$	0.0047
B	0.0127	0.0886	0.0127	0.0380	0.0380	0.0127	$\cdots$	0.0127
C	0.0051	0.0051	0.2500	0.0051	0.0051	0.0051	$\cdots$	0.0051
D	0.0083	0.0083	0.0167	0.1750	0.0083	0.0083	$\cdots$	0.0083
E	0.0052	0.0309	0.0077	0.0052	0.2809	0.2320	$\cdots$	0.0026
F	0.0059	0.0059	0.0059	0.0059	0.2012	0.2308	$\cdots$	0.0059
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
z	0.0169	0.0169	0.0169	0.0169	0.0169	0.0169	$\cdots$	0.0169

Table 4.8: Substitution matrix.

$p(z|w)$ is computed by multiplying, for each pair of letters z_i and w_i in both sequences, the probability of recognizing w_i as z_i, as we assume that the letters in a sequence are independent variables between each other. Therefore, $p(z|w)$ is computed using (4.11).

$$p(z|w) = sub[w_1, z_1] \cdot sub[w_2, z_2] \cdot \ldots \cdot sub[w_m, z_m] \tag{4.11}$$

In order to deal with the problem that each $sub[w_i, z_i]$ can come very close to zero, and the product of them can exceed the precision of double precision floating points, we can work in "log-space", which turns multiplications into addition. Therefore, (4.11) is similar to (4.12).

$$\log(p(z|w)) = \log(sub[w_1, z_1]) + \log(sub[w_2, z_2]) + \ldots + \log(sub[w_m, z_m]) \tag{4.12}$$

Actually, what we are proposing here is a simplification of the forward algorithm for Hidden Markov Models (see appendix B), in which the transition matrix is the identity matrix, the observation matrix is the substitution matrix previously explained and the number of states is the length of the sequence. The forward algorithm computes the posterior probability of an observation sequence, given a model. So, for the approach here proposed, given an observed sequence z and a correct sequence w, we are finding the path with the maximum associated probability, as it is represented in figures 4.17 and 4.18, which show, as an example, how the probability $\log(p(z|w))$ is computed given the observed sequence "lWORD" and two different correct sequences "SWORD" and "SWORE".

	l	W	O	R	D
S	sub[S,l]	sub[S,W]	sub[S,O]	sub[S,R]	sub[S,D]
W	sub[W,l]	sub[W,W]	sub[W,O]	sub[W,R]	sub[W,D]
O	sub[O,l]	sub[O,W]	sub[O,O]	sub[O,R]	sub[O,D]
R	sub[R,l]	sub[R,W]	sub[R,O]	sub[R,R]	sub[R,D]
D	sub[D,l]	sub[D,W]	sub[D,O]	sub[D,R]	sub[D,D]
	log(sub[S,l])+	log(sub[W,W])+	log(sub[O,O])+	log(sub[R,R])+	log(sub[D,D])=
			=-3.166003		

Figure 4.17: Path with maximum associated probability for the observed sequence “lWORD” and the correct sequence “SWORD”.

	l	W	O	R	D
S	sub[S,l]	sub[S,W]	sub[S,O]	sub[S,R]	sub[S,D]
W	sub[W,l]	sub[W,W]	sub[W,O]	sub[W,R]	sub[W,D]
O	sub[O,l]	sub[O,W]	sub[O,O]	sub[O,R]	sub[O,D]
R	sub[R,l]	sub[R,W]	sub[R,O]	sub[R,R]	sub[R,D]
E	sub[E,l]	sub[E,W]	sub[E,O]	sub[E,R]	sub[E,D]
	log(sub[S,l])+	log(sub[W,W])+	log(sub[O,O])+	log(sub[R,R])+	log(sub[E,D])=
			=-4.091594		

Figure 4.18: Path with maximum associated probability for the observed sequence “lWORD” and the correct sequence “SWORE”.

Up to now, we have explained how we compute $p(z|w)$ given an observed sequence z and a word w of the same length in the dictionary. This is done taking into account the different probabilities of recognizing a certain letter as another letter (the substitution matrix). However, we also want to take into account that insertions and deletions of letters are likely to happen in a sequence, but we do not have enough data to compute the corresponding matrices, thus a different approach to deal with insertions and deletions is proposed. As it was stated in the previous section, sometimes characters are erroneously separated, as a character can be wrongly split up into two parts, or two characters may be 8-connected and thus cannot be separated. A method to solve the last problem was developed and explained in the previous section, but it might fail sometimes. Therefore, we make the hypothesis that each character in the sequence may be wrongly segmented and may be formed by two letters in reality. We separate each object into two parts and apply the character recognizer to each part, as shown in figure 4.19. The point of separation is given by the minimum of the projection of the binarised object over the horizontal axis. This separation is equivalent to making an insertion in the sequence. On the other hand, in order to deal with the case of objects that are erroneously broken into two parts, we make the hypothesis that each character could have been wrongly separated, and we join each pair of adjacent binarised objects and apply the character recognizer, as shown in figure 4.20. This is equivalent to making a deletion in the sequence.

As a result of this “split and merging” process, we generate a set of candidates $Z = z^1, z^2, ..., z^L$ for each observed sequence z, in which z^1 is the candidate sequence with neither insertions nor deletions, that is, the original observed sequence ($z^1 = z$). Three examples are displayed in figures 4.21 and 4.22. The first column in each subfigure is

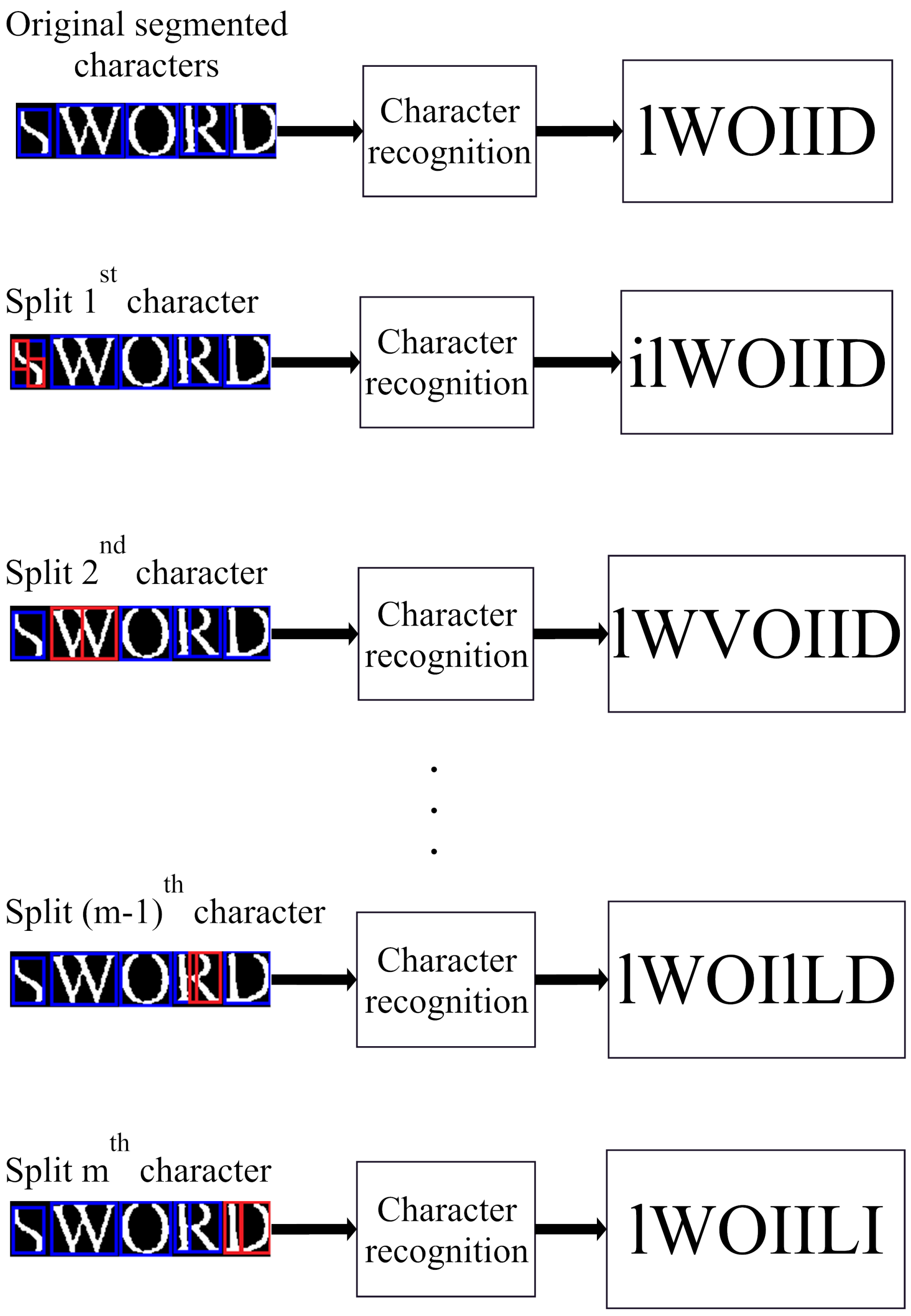


Figure 4.19: Character splitting process.

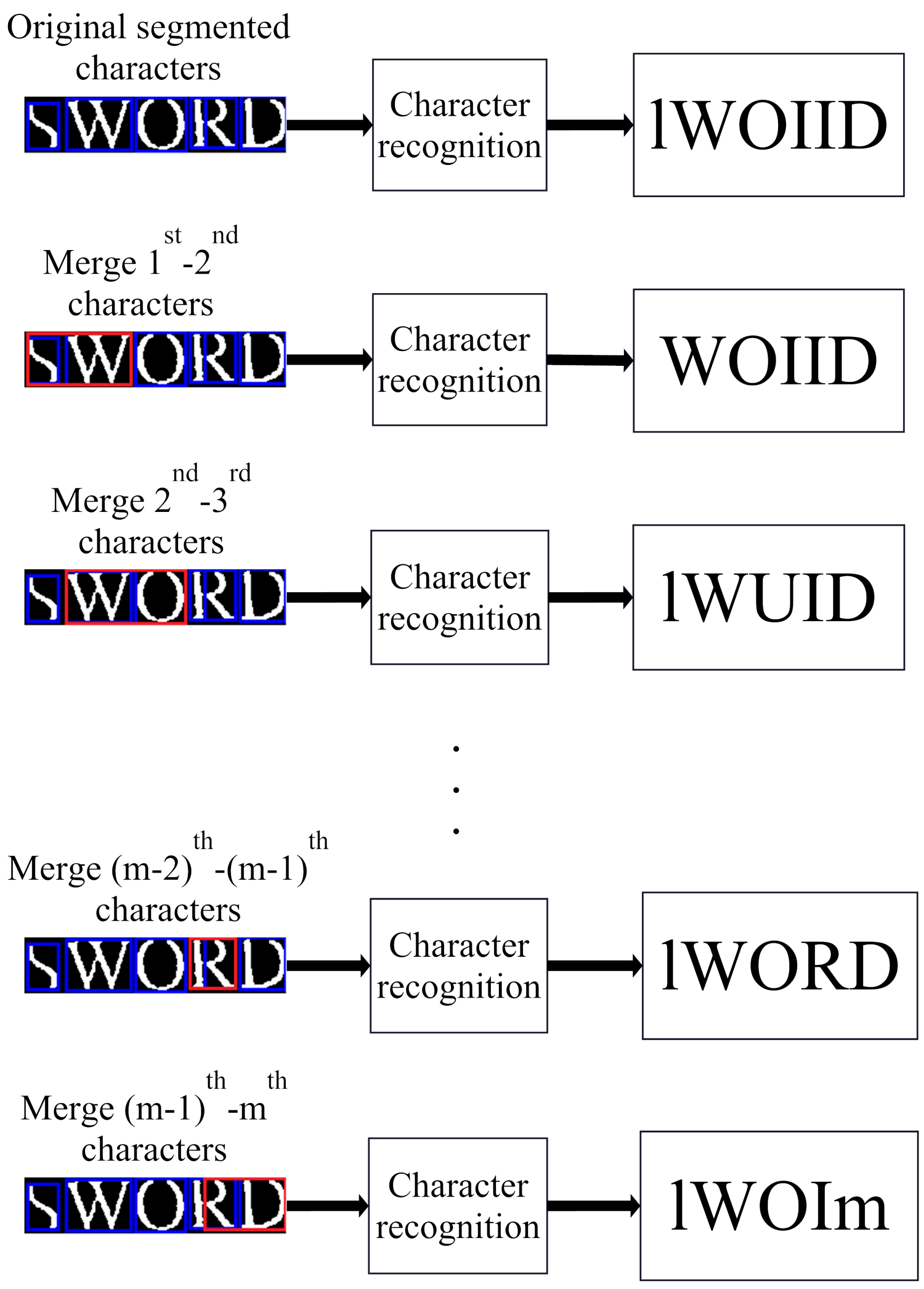


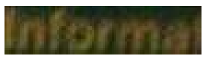
Figure 4.20: Character merging process.

the set Z of candidates. The second column is the most likely word w^i in the dictionary that may have generated each candidate z^i. The third column displays the associated probability p^i for the most likely solution w^i. This probability p^i is computed using (4.9) only for all the words in the dictionary that have the same length as z^i, taking into account their prior probabilities. Therefore, each candidate z^i of a determined observed sequence z has its own most likely solution w^i with an output probability p^i. The output w of the word recognizer for the observed sequence z, is the word w^i with the highest output probability p^i. In figure 4.21(a), the letter 'R' in "SWORD" is initially wrongly separated into two objects. On the other hand, the words "stands" in figure 4.21(b) and "informal" in figure 4.22(a) are correctly segmented.

Observed sequence z: lWOIID			Observed sequence z: StandS		
Candidate z^i	*Output* w^i	$\log(p^i)$	*Candidate* z^i	*Output* w^i	$\log(p^i)$
lWOIID	INOTTA	-3.050655	**StandS**	**stands**	**-1.757635**
WOIID	wollo	-3.011815	nandS	hands	-2.102833
lWIID	twain	-3.538945	SbndS	sends	-3.010265
lWUID	INDIA	-3.285688	StMdS	studs	-2.84479
lWORD	**SWORD**	**-2.854841**	StaNS	STARS	-3.213461
lWOIm	lydia	-3.234948	Stank	stank	-2.293556
ilWOIID	ilwiiin	-3.075759	SStandS	errands	-3.172541
lWVOIID	INVALID	-3.160552	SleandS	sceanes	-4.281876
lWCDIID	INERTIA	-3.531318	StEsndS	SIESTAS	-3.515189
lWOjLID	Aworkin	-4.632335	StaT1dS	statues	-4.206196
lWOIIlLD	BADILLA	-3.320809	StandiS	grandis	-2.525651
lWOIILI	INUTILE	-3.31027	StandES	standes	-2.674135
(a) Output: SWORD			(b) Output: stands		

Figure 4.21: Sequence candidates for each observed sequence and output of the word recognizer.

In case the candidate z^1 (the one with neither insertions nor deletions) coincides with a word in the dictionary ($z^1 \in W$), the output w is directly z^1 ($w = z$). Figure 4.21(b) and figure 4.22(a) are two examples.



Observed sequence z: tntormdI

Candidate z^i	*Output* w^i	$\log(p^i)$
tntormdI	**Informal**	**-2.488576**
MtormdI	Nigraha	-3.922166
tmormdI	factual	-3.059271
tnMrmdI	thurman	-3.695055
tntAmdI	torwada	-3.798835
tntomdI	thrombi	-2.824636
tntormI	Immoral	-2.981749
tntormi	Immoral	-2.977353
1rntormdI	nonformal	-3.570819
tt1tormdI	fraternal	-4.025162
tniIormdI	mortarman	-3.852788
tnt4IrmdI	threlfall	-4.731569
tntoIrmdI	threlfall	-3.685633
tntorn1dI	triennal	-4.13146
tntormliI	Interalia	-2.957902
tntormdIl	Informati	-2.889274

(a) Output: Informal

Figure 4.22: Sequence candidates for each observed sequence and output of the word recognizer.

4.2.2 Numbers and punctuation marks recognition

Sometimes the word to be recognized is a number. However, there is not any number in the BNC, as this would imply to have an infinite dictionary of all the possible terms. In this case, if m is the length of the observed sequence z and the number of digits in z is at least $m/4$, then, the objects that do not correspond to digits are classified again but only using the classes '0'-'9'. The output of the word recognizer is directly the observed sequence z. On the other hand, if digits are found but the amount is less than $m/4$, then, the objects that correspond to digits are classified again but leaving out the classes '0'-'9', and the new observed sequence z is recognized using the method explained in the previous subsection, including the split and merging process and so on. The whole algorithm is summarized in figure 4.23.

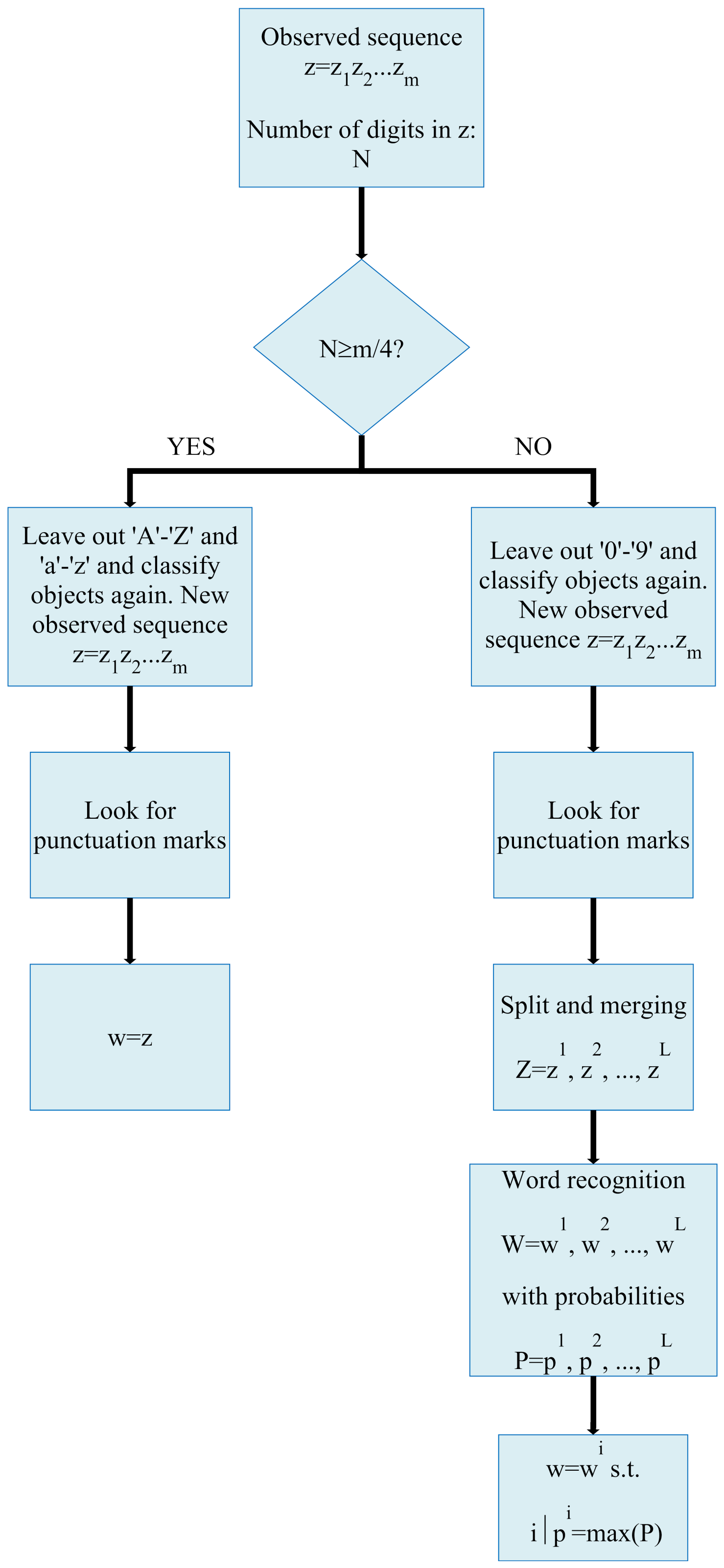
Punctuation marks are also taken into account. Due to the fact that the height of punctuation marks such as full stops, commas, quotation marks or apostrophes is much smaller than the height of the characters of the accompanying word, they are discarded in the CC analysis explained in the previous chapter when grouping objects into words. What we do is to look for these punctuation marks just before applying the word recognizer by searching for smaller objects, which do not belong to any word, in the adjacent areas of the characters of each word. Firstly, we compute the line that divides each word into two halves with (4.13).

$$y_{half} = y_{min} + \frac{y_{max} - y_{min}}{2} \tag{4.13}$$

where y_{max} and y_{min} are the maximum and minimum values of the y coordinate of the bounding box of the word.

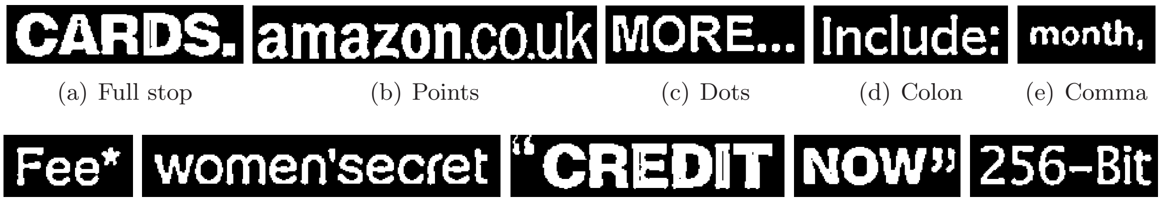
- If the object is located below y_{half} and the eccentricity of the ellipse that contains the object tends to 0 (the object tends to be circular), then the object is a full stop (figure 4.24(a)), a point in the middle of the word (figure 4.24(b)) or dots (figure 4.24(c)). Two points, one above the other, are a colon (figure 4.24(d)).
- If the object is located below y_{half} and the eccentricity of the ellipse that contains the object tends to 1 (the object tends to be lengthy in vertical direction), then the object is a comma (figure 4.24(e)). A point above a comma is a semicolon.
- If the object is located above y_{half} and the eccentricity of the ellipse that contains the object tends to 0 (the object tends to be circular), then the object is an asterisk (figure 4.24(f)).
- If the object is located above y_{half} and the eccentricity of the ellipse that contains the object tends to 1 (the object tends to be lengthy in vertical direction), then the object is an apostrophe (figure 4.24(g)). In case there is a similar object besides it in terms of eccentricity and height, two apostrophes are together and that is the case of a quotation mark (figure 4.24(h)).
- If the object is located around y_{half} and the eccentricity of the ellipse that contains the object tends to 1 (the object tends to be length in the horizontal direction), then the object is a hyphen (figure 4.24(i)).

If a punctuation mark is found between two characters of a same word, then we split the word into two different words and apply the word recognizer separatedly for each new generated word.



Observed sequence $z=z_1z_2...z_m$
Number of digits in z: N
N≥m/4?
YES
NO
Leave out 'A'-'Z' and 'a'-'z' and classify objects again. New observed sequence $z=z_1z_2...z_m$
Leave out '0'-'9' and classify objects again. New observed sequence $z=z_1z_2...z_m$
Look for punctuation marks
Look for punctuation marks
w=z
Split and merging $Z=z^1, z^2, ..., z^L$
Word recognition $W=w^1, w^2, ..., w^L$ with probabilities $P=p^1, p^2, ..., p^L$
$w=w^i$ s.t. $i \mid p^i=\max(P)$

Figure 4.23: Word recognition flowchart



(a) Full stop (b) Points (c) Dots (d) Colon (e) Comma

(f) Asterisk (g) Apostrophe (h) Quotation mark (i) Quotation mark (j) Hyphen

Figure 4.24: Punctuation marks.

4.2.3 Experimental results

Table 4.9 shows the recognition rate achieved for the ICDAR 2003 word recognition task. There were no participants in this contest in the 2003 and 2005 editions. From our knowledge, no one has used this dataset as benchmark in terms of word recognition in the last decade. Therefore, we cannot compare our results to any other methods. On the other hand, table 4.11 and table 4.13 show the performance of our algorithm using the datasets released for both challenges in the ICDAR 2011 competition. In this case, there were several entries. We compare our algorithm to them. It can be clearly seen that we score first in both challenges in terms of correct recognition rate. However, we obtain a higher value for the total normalized edit distance. This is due to the fact that we are applying a word recognizer based on probabilistic inference, while other works do not. This can lead to giving outputs which are completely different to the input, *i.e.* they have a high normalized edit distance, close to 1 or even above 1, while other methods do not but the word recognition rate is much lower. This effect can be seen in figures 4.25, 4.26 and 4.32, which show the histogram of normalized edit distances. For the first two figures, the histograms are shown only for the proposed method, as we do not have any data available for other works, but the third figure compares our method to other state-of-the-art algorithms for which we have data available. It is notable that the percentage of cases with a normalized edit distance equal or higher than 1 (corresponding to all characters being changed), is higher for the proposed method than for other methods. However, we firmly think that the best way to assess the performance of a word recognition algorithm is to measure the number of correctly recognized words, as the final task is to recognize single words, as most as possible.

Tables 4.10, 4.14 and 4.12 highlight the importance of using a language model to correct misspelled words. It can be clearly seen that the word recognition rate using a language model triples the rate when a language model is not applied for the three cases.

Figures 4.27-4.31 show some examples on the ICDAR 2003 and ICDAR 2011 (challenge 2) datasets, which are very similar. Below each image, the output of the word recognizer is transcripted. The first three figures show examples of correct recognized words. It can be seen that the proposed method is robust in a large variety of situations including different font styles, heterogeneous illumination, complex background, low image resolution, slanting text and connected characters. On the other hand, figures 4.30-4.31 show some situations where the word recognizer fails. The errors are due to different reasons. In some cases, the words are not included in the BNC (figures 4.30(a)-(n)) or the words are hard to correctly separate into characters (4.30(p)-(r)). In other cases, the observed sequences coincide with words contained in the dictionary and the observations are given as output, or the word is very similar to other word whose prior probability is larger (figures 4.31(a)-(i)). Wrongly recognized numbers are another source of errors, as it is shown in figures 4.31(j)-(r).

Algorithm	Correct recognition (%)	Total Edit Distance
Our system	**47.43**	**616.87**

Table 4.9: Word recognition on ICDAR'03

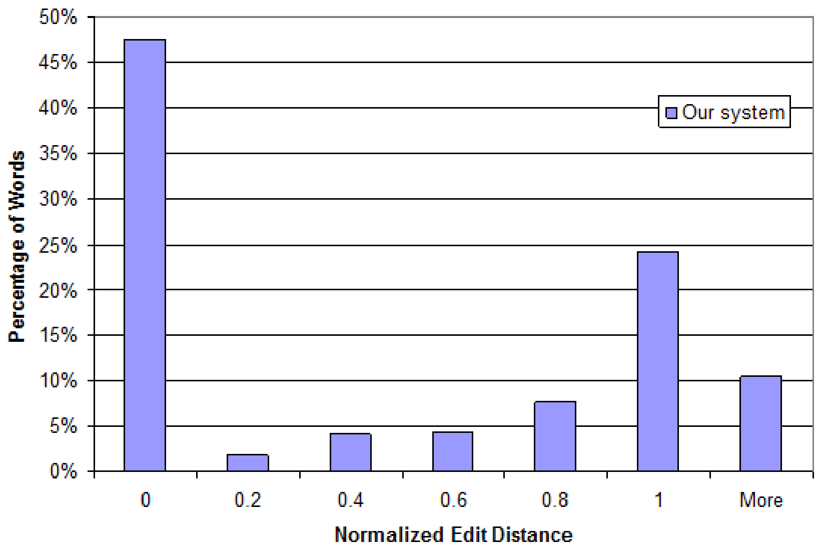


Figure 4.25: Histogram of normalized edit distances for ICDAR 2003

Algorithm	Correct recognition (%)	Total Edit Distance
Our system without language model	16.62	717.80
Our system with language model	47.43	616.87

Table 4.10: Effect of language model on word recognition (ICDAR'03)

Algorithm	Correct recognition (%)	Total Edit Distance
Our system	**46.9**	**639.15**
TH-OCR [Shahab et al., 2011]	41.2	176.23
KAIST AIPR [Shahab et al., 2011]	35.6	318.46
Neumann's method [Shahab et al., 2011]	33.11	429.75

Table 4.11: Word recognition on ICDAR'11 Chall. 2.

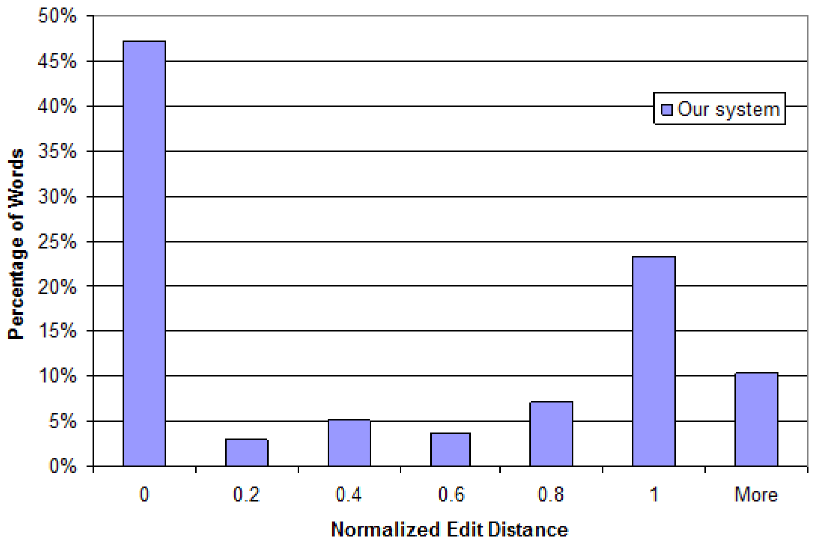
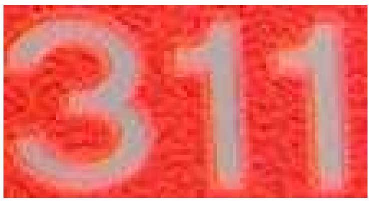


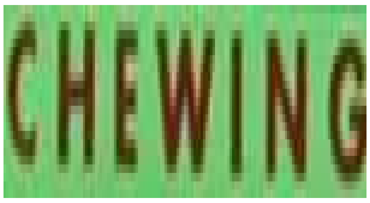
Figure 4.26: Histogram of normalized edit distances for ICDAR 2011 Challenge 2

Algorithm	Correct recognition (%)	Total Edit Distance
Our system without language model	14.9	788.63
Our system with language model	46.9	639.15

Table 4.12: Effect of language model on word recognition (ICDAR'11 Chall. 2.)


(a) 311

(b) SUMMER


(c) CHEWING

(d) 20p

(e) GLASS

(f) NAILS

(g) HUNGRY

(h) COLCHESTER

(i) BRITAIN'S

(j) DEPARTMENT

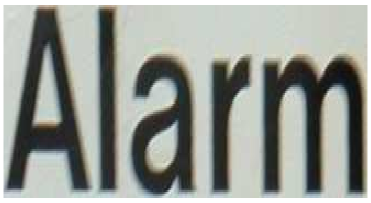
(k) County

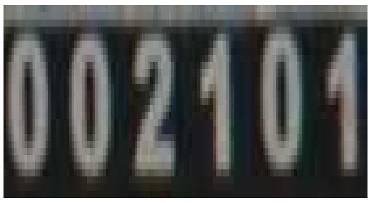
(l) Library

(m) Management

(n) Coach

(o) EXPRESS


(p) Alarm


(q) 002101

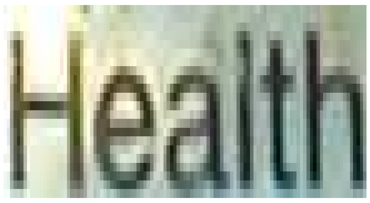

(r) Health

Figure 4.27: Word recognition on ICDAR 2003/ICDAR 2011 (Chall. 2).

(a) price

(b) University

(c) PEUGEOT

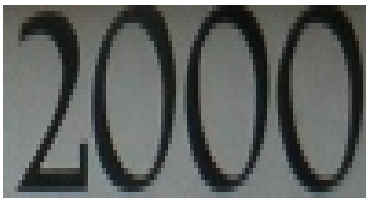
(d) kills

(e) JACKS

(f) alternative

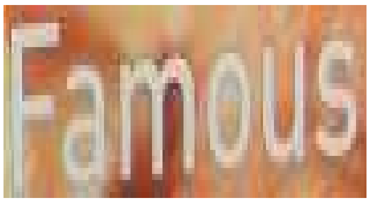
(g) clearance

(h) ESSENCE


(i) 2000

(j) videos

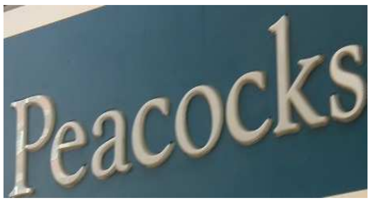
(k) loaned

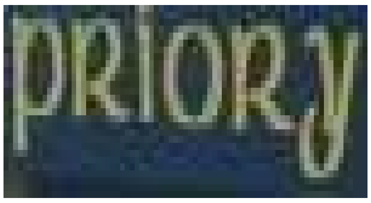

(l) Famous

(m) fish

(n) DRY

(o) CLEANED


(p) Peacocks


(q) priory

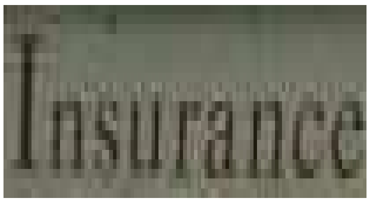
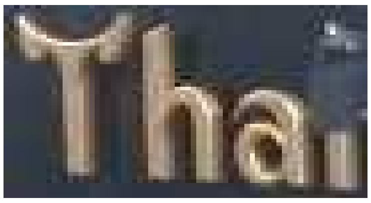

(r) Insurance

Figure 4.28: Word recognition on ICDAR 2003/ICDAR 2011 (Chall. 2).



(a) Thai

(b) MILL

(c) ANTIQUES

(d) TRANSPORT

(e) 794447

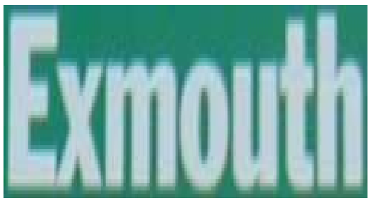
(f) FLOOR

(g) OFFICE

(h) Tetley

(i) Mark

(j) Steinbeck



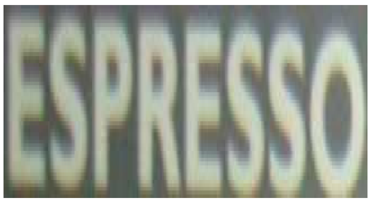
(k) Exmouth

(l) CONTRACTOR’S

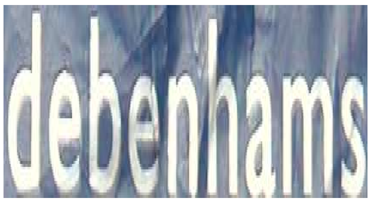
(m) Committee

(n) ENGINEERING

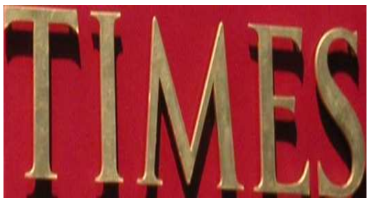
(o) HERE



(p) ESPRESSO

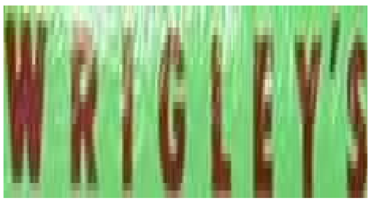


(q) debenhams

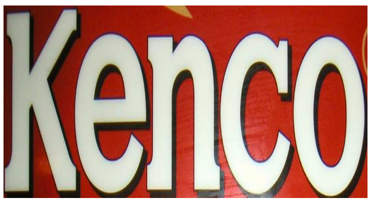


(r) TIMES

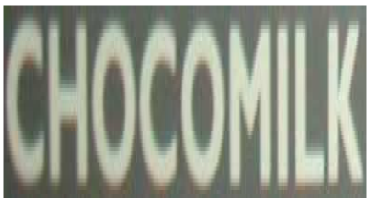
Figure 4.29: Word recognition on ICDAR 2003/ICDAR 2011 (Chall. 2).



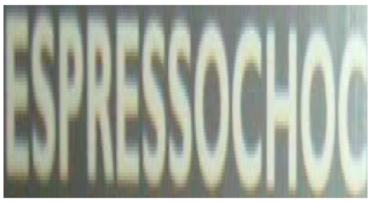
(a) BAILEYS



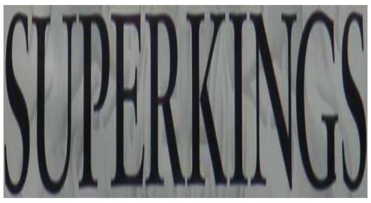
(b) Women



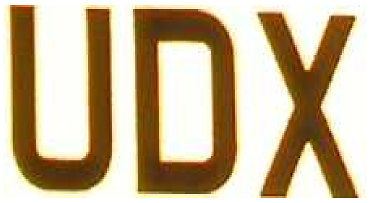
(c) LOCALLY



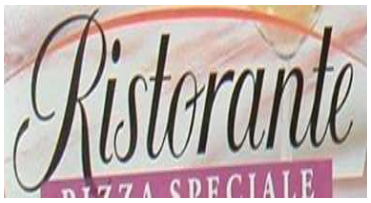
(d) RENOUNCE



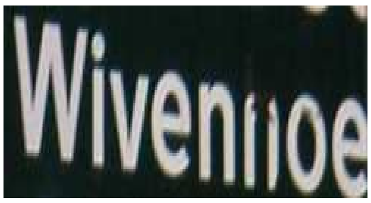
(e) SPELLINGS



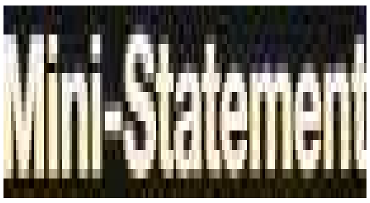
(f) LOOK



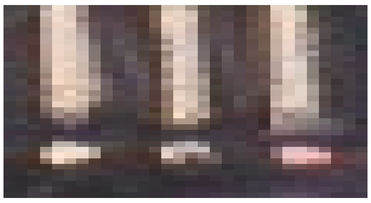
(g) birthmark

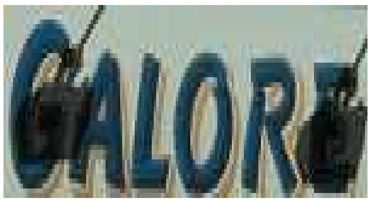


(h) argentine

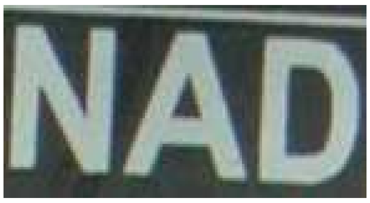


(i) Misstatement

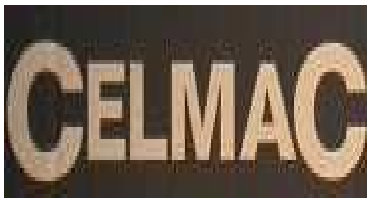

(j) III



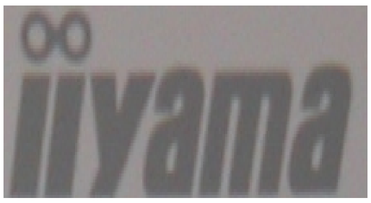
(k) ISIDORA



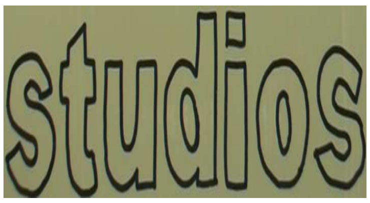
(l) HAD



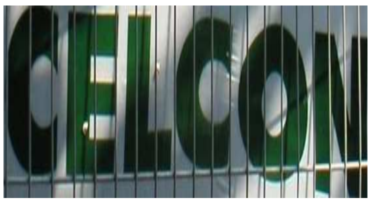
(m) CENAC



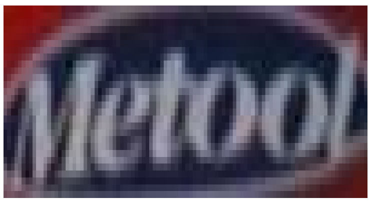
(n) Provide



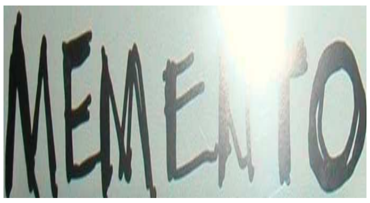
(o) Stages



(p) charlottenstrasse

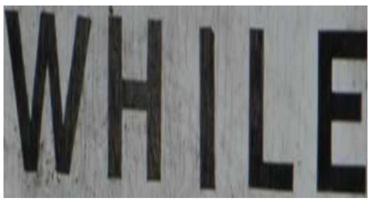


(q) on

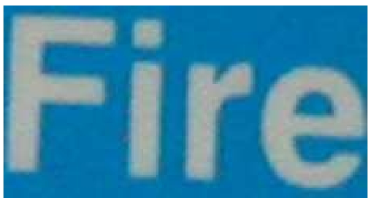


(r) Mmb

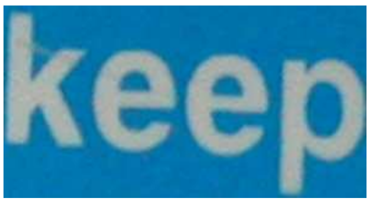
Figure 4.30: Word recognition on ICDAR 2003/ICDAR 2011 (Chall. 2).



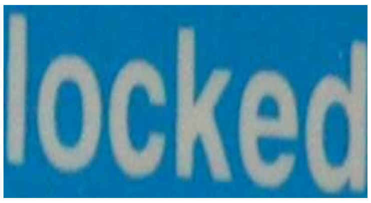
(a) WHOLE



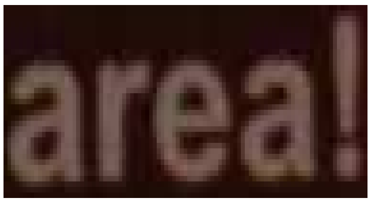
(b) Free



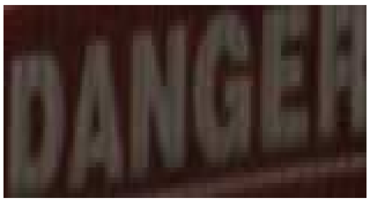
(c) Sleep



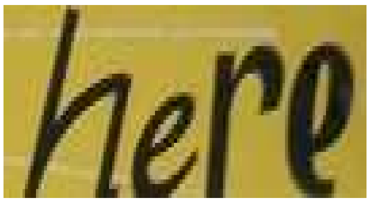
(d) looked



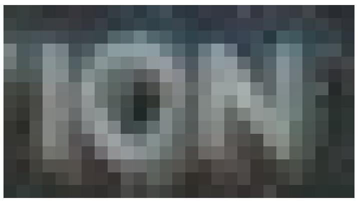
(e) areal



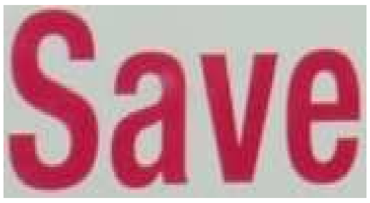
(f) CANCER



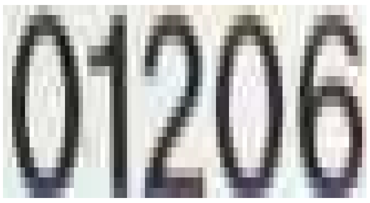
(g) hero



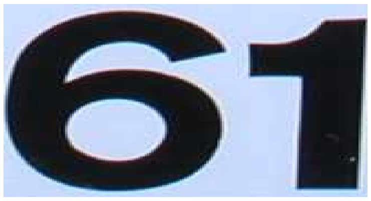
(h) LION



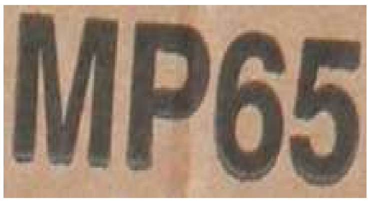
(i) have



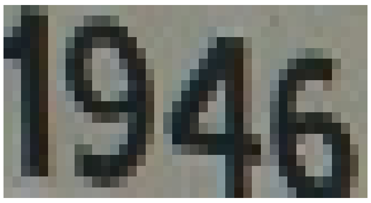
(j) 01205



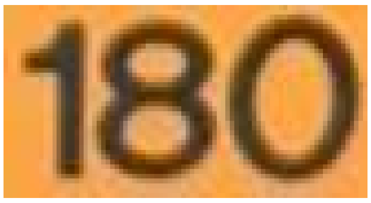
(k) 51



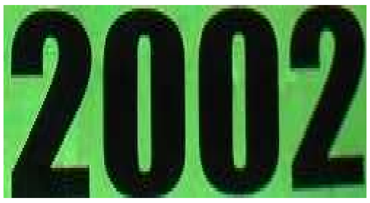
(l) MP66



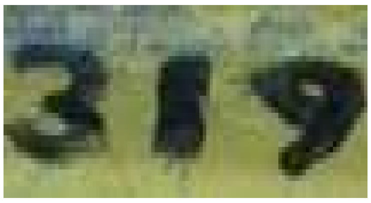
(m) 7946



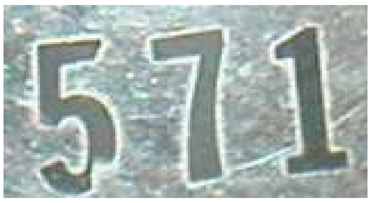
(n) 150



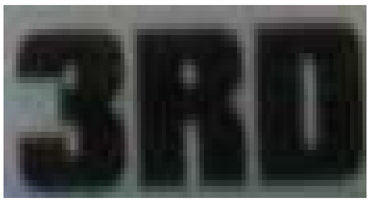
(o) 1112



(p) say



(q) Sell



(r) 3n0

Figure 4.31: Word recognition on ICDAR 2003/ICDAR 2011 (Chall. 2).

Figures 4.33-4.35 show some examples on the ICDAR 2011 challenge 1 dataset. All the results can be seen in the website of the challenge [1]. The first two figures display some cases of correctly recognized words, while the third one shows some errors, which are due to the same reasons explained above for the ICDAR 2003 dataset.

Algorithm	Correct recognition (%)	Total Edit Distance
Our system	**66.88**	**226.8**
Baseline [Karatzas et al., 2011]	63.94	231.2
TH-OCR [Karatzas et al., 2011]	61.98	189.1

Table 4.13: Word recognition on ICDAR'11 Chall. 1.

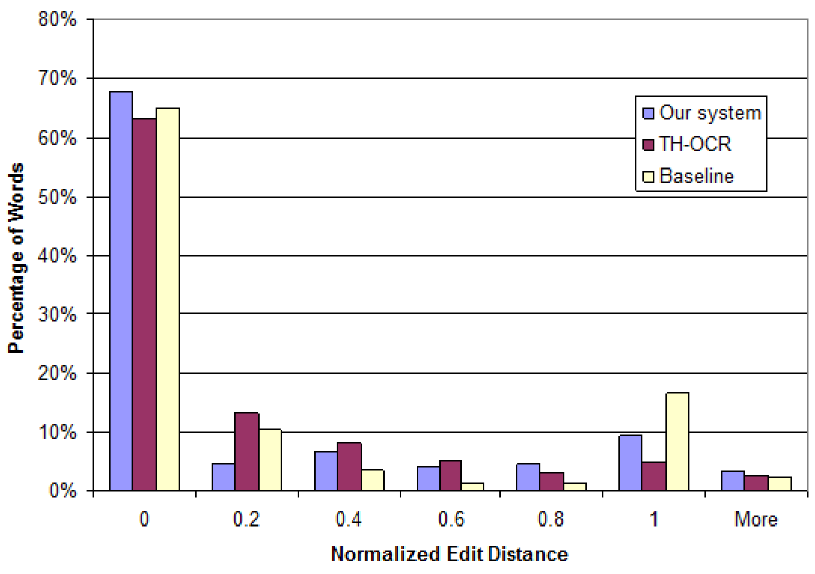
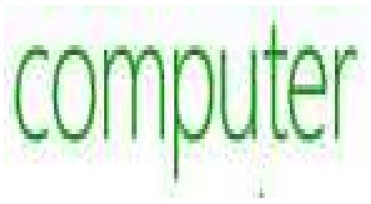


Figure 4.32: Histogram of normalized edit distances for ICDAR 2011 Challenge 1

Algorithm	Correct recognition (%)	Total Edit Distance
Our system without language model	23.64	329.8
Our system with language model	66.88	226.8

Table 4.14: Effect of language model on word recognition (ICDAR'11 Chall. 1.)

[1] http://www.cvc.uab.es/icdar2011competition/?com=results&action=words_list&id_submit=474


(a) computer

(b) Download

(c) 2010

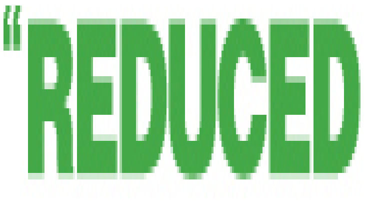
(d) Lufthansa

(e) electronics,

(f) shoes.

(g) Learn

(h) CARDS.


(i) “REDUCED

(j) spend

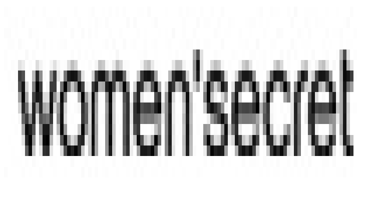
(k) It’s

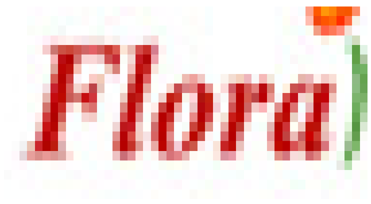
(l) Chomps.

(m) Zopa

(n) publications”

(o) WEBINAR


(p) women’secret


(q) Flora

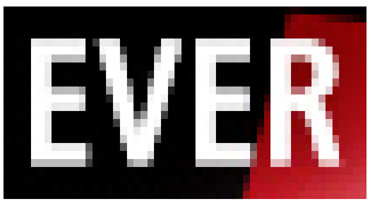
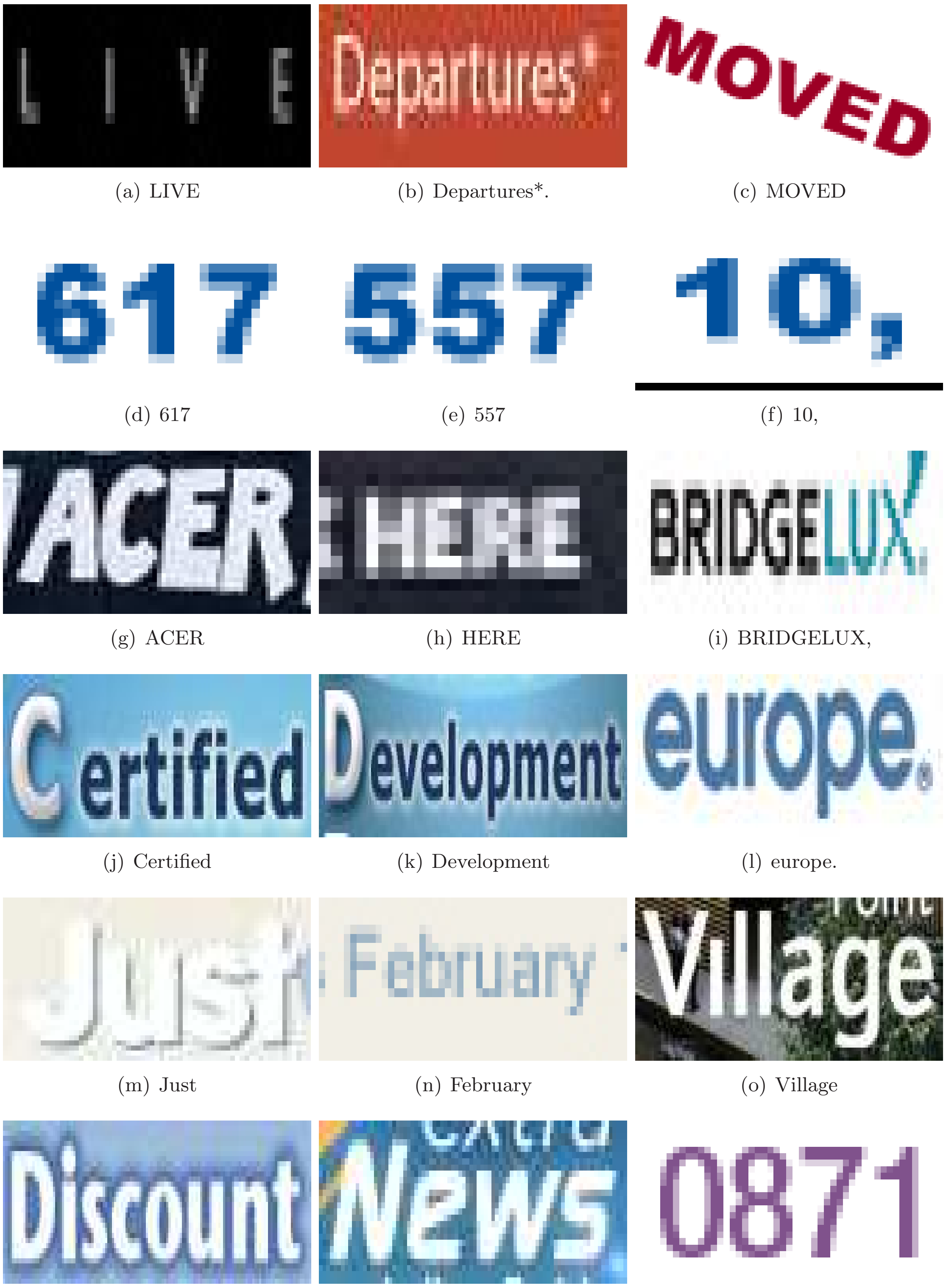

(r) EVER

Figure 4.33: Word recognition on ICDAR 2011 Challenge 1.



(a) LIVE (b) Departures*. (c) MOVED

(d) 617 (e) 557 (f) 10,

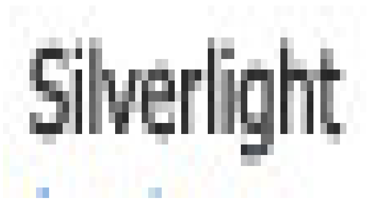
(g) ACER (h) HERE (i) BRIDGELUX,

(j) Certified (k) Development (l) europe.

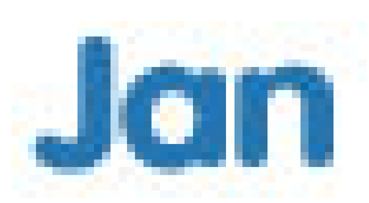
(m) Just (n) February (o) Village

(p) Discount (q) News (r) 0871

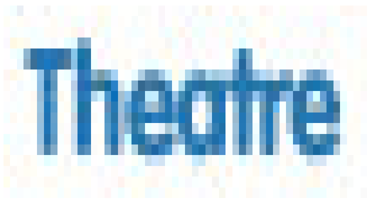
Figure 4.34: Word recognition on ICDAR 2011 Challenge 1.



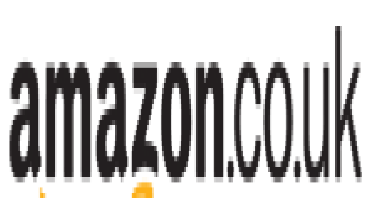
(a) Starlight



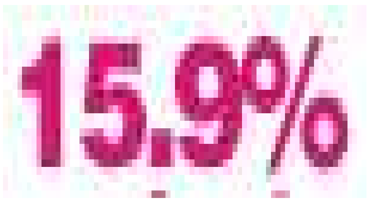
(b) Jon



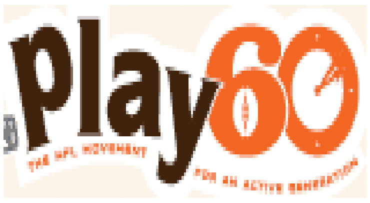
(c) Theme



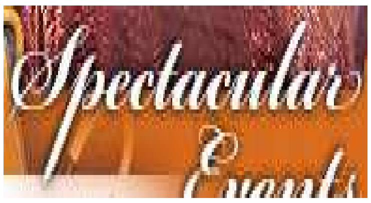
(d) amazon.DO.Uk



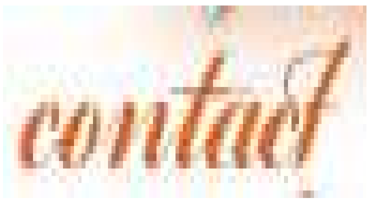
(e) 15.g%



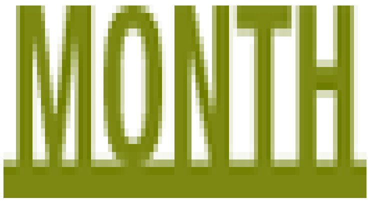
(f) played



(g) OBJECTING

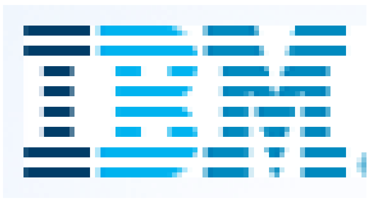


(h) whitnall

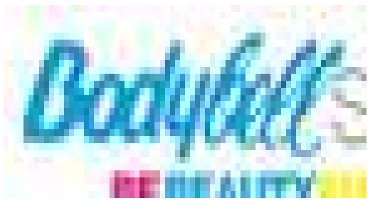


(i) LILL

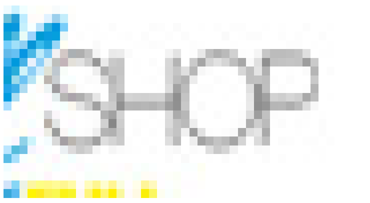
(j) 24-25,



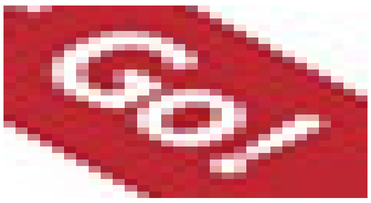
(k) L



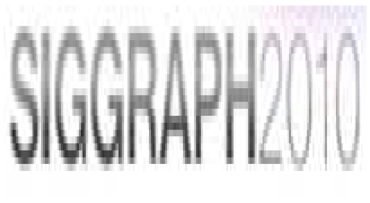
(l) Meryl,



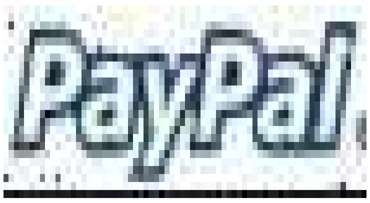
(m) in'



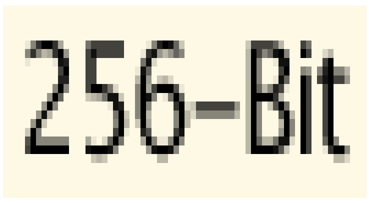
(n) Oil



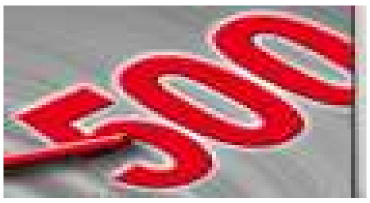
(o) SHOGRAPHICS



(p) WA



(q) 255-Bit



(r) n

Figure 4.35: Word recognition on ICDAR 2011 Challenge 1.

4.3 Conclusions and future works

In this chapter we proposed a novel method for recognizing text in natural images. Our approach is based on identifying single characters and then applying a language model to correct misspelled words, constraining the output to a dictionary of all the possible terms.

A new feature based on gradient direction histogramming has been proposed to characterize single letters. We have named it as Direction Histogram (DH). This new feature has been compared to other state-of-the-art features, such as Shape Context, Local Binary Patterns, SIFT, Geometric Blur or HOG, and the experimental results obtained on a challenging dataset show that the proposed feature is more than adequate as it outperforms the results achieved not only with those features, but also with other methods. In addition, we have developed a fuzzy classification method based on KNN, which gives different solutions for each character, each solution with a certain probability. These probabilities, together with other evidences, have been proved to be useful to separate characters that may have wrongly connected during the segmentation process. However, the proposed method has some heuristic characteristics. Therefore, as future work we intend to carry out the separation of characters using fuzzy inference.

Finally, a language model based on probabilistic inference has been proposed to constrain the output of the character recognizer to a set of meaningful words. The model is based on the British National Corpus, which is a representation of modern British English. The language model finds the most likely word that may have generated an observed sequence. This is carried out using Dynamic Programming, espepcifically the forward algorithm. We propose to model the substitutions using probabilities instead of costs. In order to avoid the assignment of zero probabilities, Add-1 smoothing is used. Deletions and insertions are not modeled using their corresponding matrices, but using a split and merging process in the segmented image. The proposed method is also able to recognize numbers as well as punctuation marks, using geometric characteristics for the punctuation marks. Experimental results with three challenging datasets have been obtained and they show the robustness of the proposed method, as we improve state-of-the-art performance.

Chapter 5

Text Detection and Recognition on Traffic Panels from Street-level Imagery

The previous chapters have presented a method to detect and recognize text in images taken from natural scenarios, as well as in web images. The experimental results have shown the robustness of the proposed technique in many kinds of situations, including indoors and outdoors scenarios, different writing styles, sizes and layout. In this chapter, we present a real application of the proposed method to Intelligent Transportation Systems (ITS) according to one of the main investigation lines of the Robesafe Research Group [1]. The proposed algorithm to locate and recognize text is applied to read the information contained in traffic panels using the images served by Google Street View, which provides panoramic views from positions along many streets and roads in the world. The aim of this chapter is, in first place, to show that the text detection and recognition method proposed in this thesis can be generalized to other scenarios which are completely different to those which have been tested with the used datasets, whithout needing to re-train the system. In second place, we want to develop an application that enables the creation of up-to-date inventories of traffic panels of regions or countries.

Initially, it is necessary to define what a traffic panel is. Typically, the vertical signposting in a road can be classified into three main groups according to the information they depict:

1. Warning signs. They warn of dangerous or unusual conditions ahead such as a curve, turn, dip or sideroad.

2. Regulatory signs. They show the course a driver must follow and an action they are required to take or forbidden to take.

3. Information signs. They show distances and destinations.

The color, the shape or how the information is depicted in the vertical signposting differ from some countries to others. However, what all the countries have in common is that the warning signs and the regulatory signs of the same type must fulfill a restricted set of characteristics. For instance, all the "STOP" signs in a country have the same shape and the same color. The size may differ depending on the category of the road where

[1] http://www.robesafe.com/

they are located, but it is limited by regulation to a small number of discrete values. However, unlike the two first categories, the information signs depict the information in many different ways, using different colors, sizes and showing different messages depending on the road and place where they are located and the category of the information they show, thus the characteristics are not discretized. In other words, there are not two similar information signs, because the information depicted in them is not restricted. Information signs are typically bigger than warning and regulatory signs and, unlike them, can be found not only beside the road but also above it. Figure 5.1 shows the classification of the vertical signposting showing some examples of the Spanish road network. From now on, we will name the warning and the regulatory signs as traffic signs, while the information signs will be known as traffic panels.

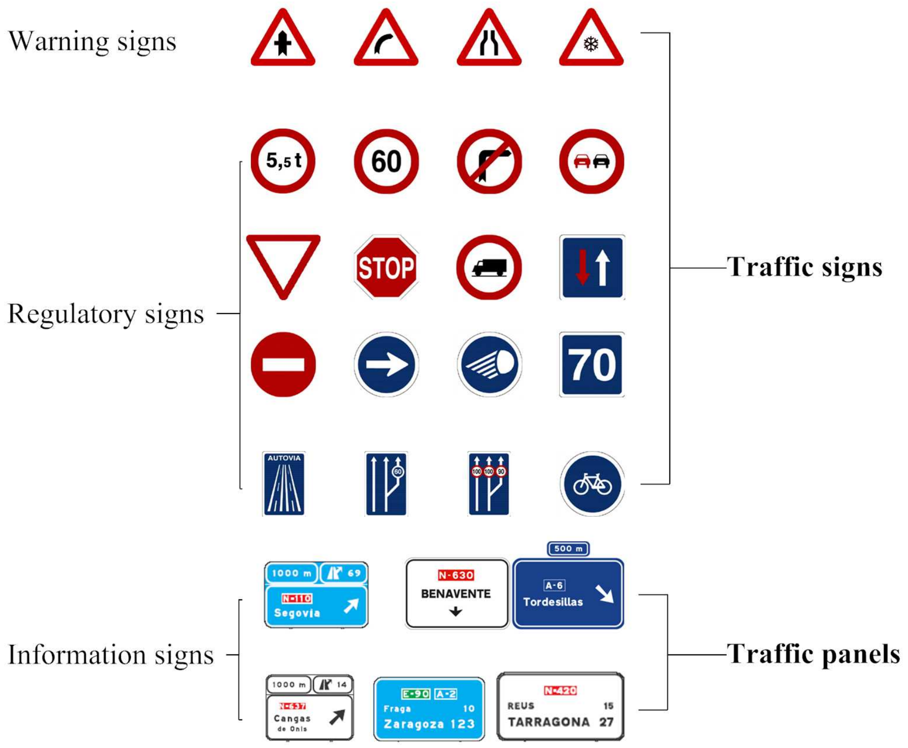


Figure 5.1: Classification of the vertical signposting

In this work, we focus on traffic panels in the Spanish territory for two main reasons. Firstly, unlike other countries, the coverage of Street View in Spain is near complete, thus we can create a huge and diverse dataset of images. Secondly, as far as we know, there is not any official database of all the traffic panels in Spain, thus there are more possibilities that any government or institution responsible for managing the road network is interested in having an up-to-date inventory of the traffic panels in Spain with the method here proposed. The reasons for which these organisations may be interested are various. Traffic signs and panels visibility degrades due to aging and other causes such as vandalism, accidents, pollution or vegetation coverage. Street-level panoramic image recording services, like Street View, which have become very popular in the recent years and have reached a huge coverage of the road network, suppose a potential source to know the state of degradation of the vertical signposting of the road network, just in case the street-level images are updated regularly. Computer vision techniques applied on this kind of images simplify and speed up the creation of traffic signposting inventories, minimizing the human interaction. In addition, these inventories can be useful not only for supporting maintenance, but also for developing future driver assistance systems.

However, traffic panels detection still remains a very challenging problem due to several reasons. As it was explained above, there is a huge variability of traffic panels as each of them depicts different information, varying in size, color and shape. Moreover, there are large viewpoint deviations due to the fact that the images are captured from a driving vehicle. There may also be occlusions due to vegetation or other road users. In addition, weather and illumination conditions are a key problem in any kind of vision-based system. Apart from this, there are many elements in the roads or beside them that can be easily confused with traffic panels, such as advertisement panels or truck bodies.

Traffic signs and panels in the Spanish road network are regulated by the *Norma 8.1-IC sobre Señalización Vertical de la Instrucción de Carreteras* [Ministerio de Fomento, 2000] and the *Reglamento General de Circulación* [Ministerio de Presidencia, 2003]. What we need to know about traffic panels in Spain for this work are the following main aspects:

- The panels can be found above the road or beside it.
- They can have blue or white background. In the first case, the text, borders and pictograms are depicted in white color, while they are displayed in black color in the second case.
- Regardless of the background color of the panel, small text areas of different color can appear. These areas can have either green, blue or red background and white text, or orange or yellow background and black text (see figure 5.2).

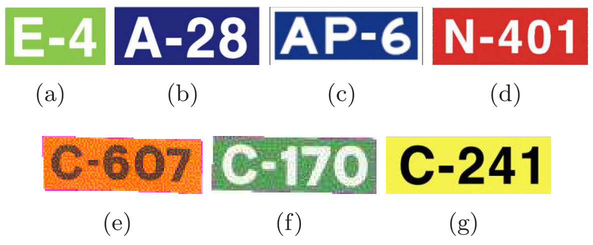


Figure 5.2: Road identification signs

Previous attempts of developing systems to recognize the information depicted in traffic panels are explained in section 5.1. The following sections explained the proposed application (a flowchart is shown in figure 5.3). Firstly, the input images are downloaded from the Street View website using the API provided by Google. This process will be explained in section 5.2. Then, a method that detects the presence of a traffic panel is executed on each input image. This procedure is explained in section 5.3. In case a panel has been detected in the image, the text detection and recognition algorithm is applied on the image, as it is explained in section 5.4, and a geolocalization method is carried out to estimate the geographic coordinates of the panel, as shown in section 5.5. Experimental results and main conclusions are displayed in sections 5.6 and 5.7, respectively.

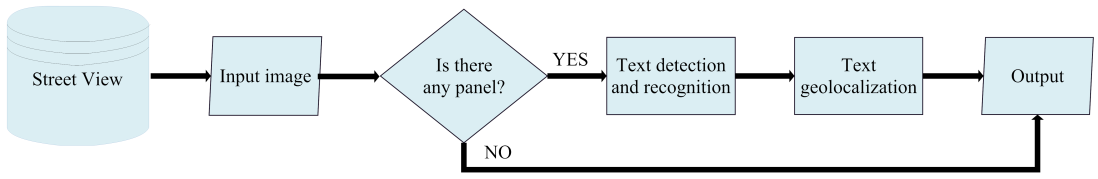


Figure 5.3: The flowchart of the proposed application

5.1 Related work

Because of the wide diversity of the information contained in traffic panels, as well as the usual problems related to outdoor computer vision systems such as occlusions, shadows and non-controlled lighting conditions, to date there has not been much research on automatic visual classification of the information contained in road panels. From our knowledge, only two works have been developed in this matter.

The work proposed by [Reina et al., 2006] extracts candidates to be traffic panels using a method that detects blue and white areas in the image using the Hue and Saturation components of the HSI space. Then, candidates are classified according to their shapes, in order to extract the rectangular blobs. This is done through a method that correlates the radial signature of their FFT (Fast Fourier Transform) with a pattern corresponding to an ideal rectangular shape. Then, panel reorientation is carried out using an homography that aligns the four vertexes of each blob. Once the panels have been detected and reoriented, segmentation of the foreground objects from the background of the panel is done by analysing the chrominance and luminance histograms. Connected components labeling and position clustering is finally done for the arrangement of the different characters on the panels. This algorithm is invariant to traslations, rotations, scaling and projective distortion, but it is severely affected by changing lighting conditions. In addition, there are many parameters and thresholds that are adjusted *ad hoc*. Recognition is applied at character level, but no language model is applied to correct misspelled words. There is not any information on where and how the images are extracted. Moreover, the experimental results provided by the authors do not show any kind of performance evaluation, so it is impossible to know the robustness of their proposal and no comparisons are possible, as they use their own dataset.

On the other hand, [Wu et al., 2005] propose a method to detect text on traffic panels from video. Firstly, regions of the same color are extracted using a k-means algorithm and traffic panels candidates are detected by searching for flat regions perpendicular to the camera axis. The orientation of the candidate planes are estimated using three or more points in two successive frames, so this method needs an accurate tracking method to detect corresponding points in successive frames. Further, a multiscale text detection algorithm is performed on each candidate traffic panel area. The text detection method integrates edge detection, adaptive searching, color analysis using GMM (Gaussian Mixture Models) and geometry alignment analysis. A minimum bounding rectangle is fitted to cover every detected text line. A feature-based tracking algorithm is then used to track all detected areas over the timeline as they are merged with other newly detected texts in the sequence. Finally, all detected text lines are extracted for recognition, but the authors do not comment how the recognition is carried out. In terms of text detection, this method provides good results under different lighting conditions and it is not affected by rotations and projective distortions. It achieves an overall text detection rate of 89% in their own dataset, which is not publicly available.

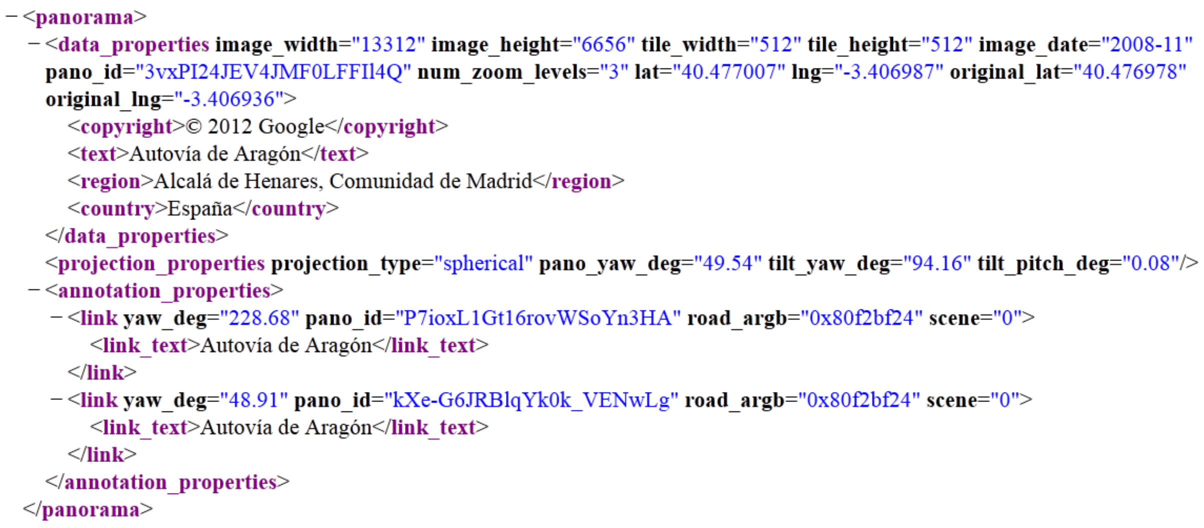
In this thesis we propose a different approach. Unlike the works above explained, which use features such as edges or geometrical characteristics to detect the traffic panels in the image, we propose to model the traffic panels using visual appearance, which is the main contribution of this chapter. Visual appearance techniques, especially Bag of Visual Words, have become very popular in computer vision in the last few years to classify images. In this chapter, we will show that this kind of algorithms are also adequate to model objects like traffic panels, despite their huge variability, without having to use geometric features. We use visual appearance to detect if an image contains a traffic panel. If so, the text detection and recognition method explained in the previous chapters is applied on the image in order to extract the information depicted in the panel. We have created for this purpose a database using images captured from Google Street View.

5.2 Image capture and dataset creation

The images used in this work have been obtained from the Street View service developed by Google. It provides high-resolution 360º panoramic views from various positions along many streets and roads in the world. These panoramic images are taken at discrete locations, around 5 or 10 meters one from each other. Each panorama is uniquely identified by a *panoid*, which is an identifying code composed of letters and digits. Given a geographical location (LAT, LON) in terms of latitude and longitude in decimal degrees respectively, it is possible to get some metadata for it with a call to a URL like this one: `http://cbk0.google.com/cbk?output=xml&ll=LAT,LNG`.

Similarly, given a *panoid* PID, the same metadata is returned with a call like: `http://cbk0.google.com/cbk?output=xml&panoid=PID`.

For instance, the following call would return the data shown in figure 5.4: `http://cbk0.google.com/cbk?output=xml&ll=40.477007,-3.406987`.



```
- <panorama>
  - <data_properties image_width="13312" image_height="6656" tile_width="512" tile_height="512" image_date="2008-11"
    pano_id="3vxPI24JEV4JMF0LFFIl4Q" num_zoom_levels="3" lat="40.477007" lng="-3.406987" original_lat="40.476978"
    original_lng="-3.406936">
      <copyright>© 2012 Google</copyright>
      <text>Autovía de Aragón</text>
      <region>Alcalá de Henares, Comunidad de Madrid</region>
      <country>España</country>
    </data_properties>
    <projection_properties projection_type="spherical" pano_yaw_deg="49.54" tilt_yaw_deg="94.16" tilt_pitch_deg="0.08"/>
  - <annotation_properties>
    - <link yaw_deg="228.68" pano_id="P7ioxL1Gt16rovWSoYn3HA" road_argb="0x80f2bf24" scene="0">
        <link_text>Autovía de Aragón</link_text>
      </link>
    - <link yaw_deg="48.91" pano_id="kXe-G6JRBlqYk0k_VENwLg" road_argb="0x80f2bf24" scene="0">
        <link_text>Autovía de Aragón</link_text>
      </link>
    </annotation_properties>
  </panorama>
```

Figure 5.4: `http://cbk0.google.com/cbk?output=xml&ll=40.477007,-3.406987`

The attribute *pano_id* in the structure *data_properties* is the *panoid* of the nearest panoramic view to that position (LAT,LON). Given a *panoid* PID, the whole panoramic image can be accessed with a URL like: `http://cbk0.google.com/cbk?output=tile&panoid=PID&zoom=0&x=0&y=0`.

It returns a 512×512-pixel image like the shown in figure 5.5. However, it is possible to zoom in on each panoramic image up to 5 zoom levels. At each zoom level, the image is given in 512-pixel square tiles. The following request gives a certain tile at a certain zoom level:
`http://cbk0.google.com/cbk?output=tile&panoid=PID&zoom=ZOOM&x=X&y=Y`.

PID is the *panoid* of the image, ZOOM is the zoom level, which ranges from 0 to 5, and X and Y relate to the horizontal and vertical tile positions. The number of X and Y positions increases with each zoom level. Figure 5.6 shows the first 4 zoom levels for a certain view and the values X and Y for each tile.

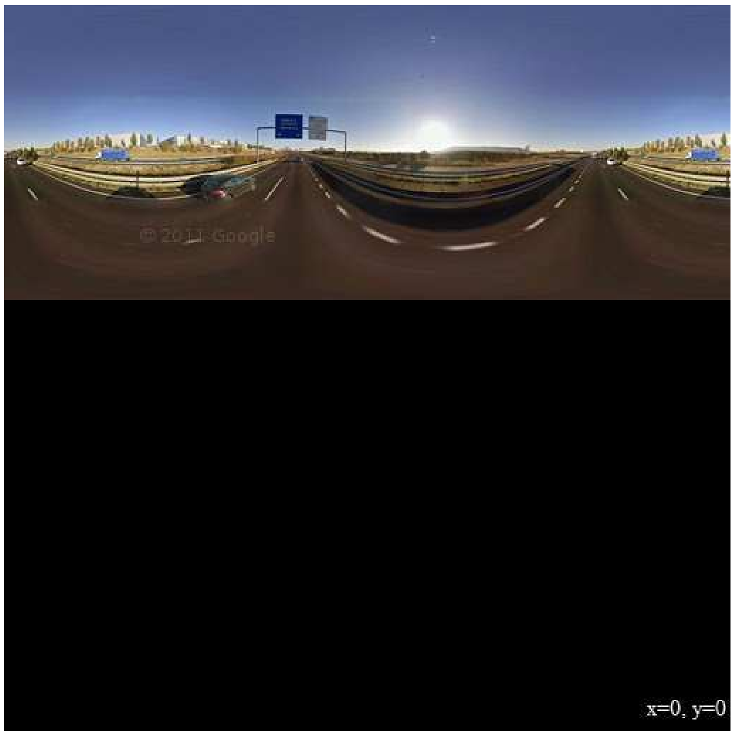


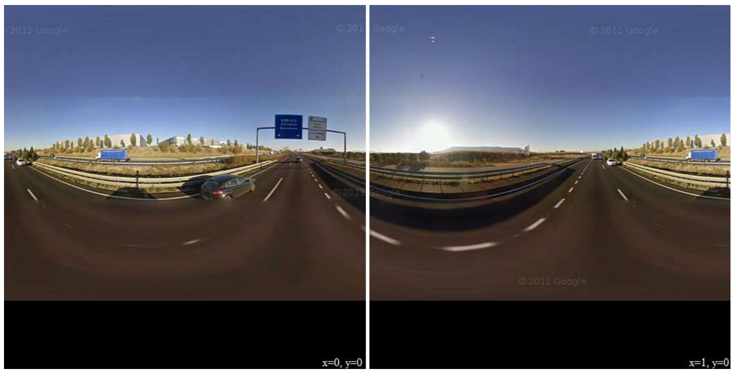
Figure 5.5: Panoramic view

For the purpose of this thesis of detecting traffic panels, it has been chosen a zoom level of 4 and the panoramic view has been cropped to the region shown in figure 5.7, where the traffic panels typically appear, both above the road and on its right side. This region corresponds to the tiles $(x = 6, y = 2)$, $(x = 6, y = 3)$, the right half side of the tiles $(x = 5, y = 2)$ and $(x = 5, y = 3)$ and the left half side of the tiles $(x = 7, y = 2)$ and $(x = 7, y = 3)$.

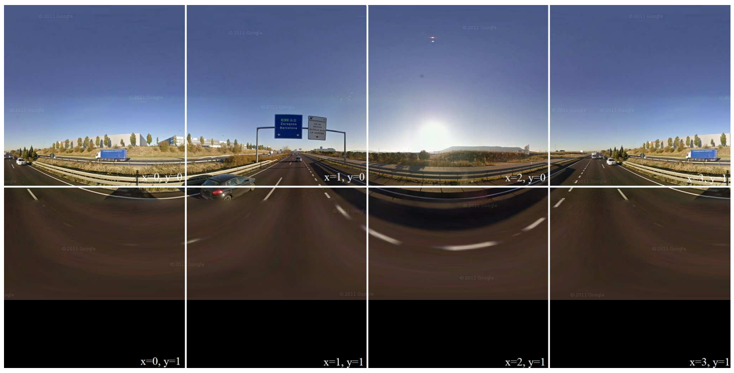
On the other hand, the metadata returned at a certain location has a structure *annotation_properties* that typically has at least two *link* elements, each of them contains the *panoid* of the previous and next panorama on the road. Three or more *link* elements means that an intersection between two or more roads or streets is given at that location.

We have developed an algorithm that, given a certain initial position, extracts consecutively, for all the positions in a road and inside a limited area, the panoramic views and crop them to the region of interest explained before. The way of doing it is summarized in algorithm 5.1.

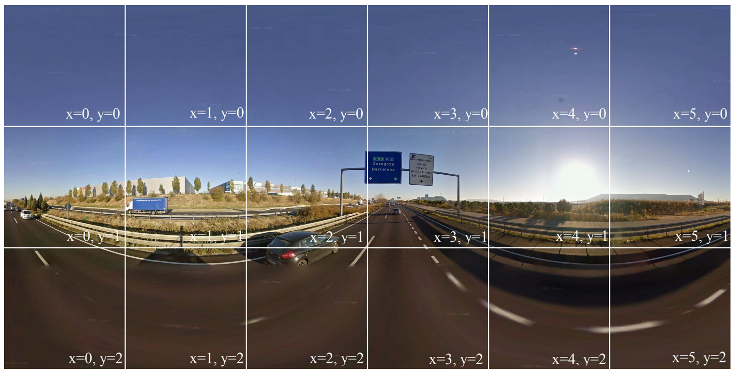
A total of 16277 images has been extracted and two independent subsets of images have been created, one for training the system, composed of 5514 images (1047 positive samples of 509 different panels and 4467 negative samples), and other subset for testing



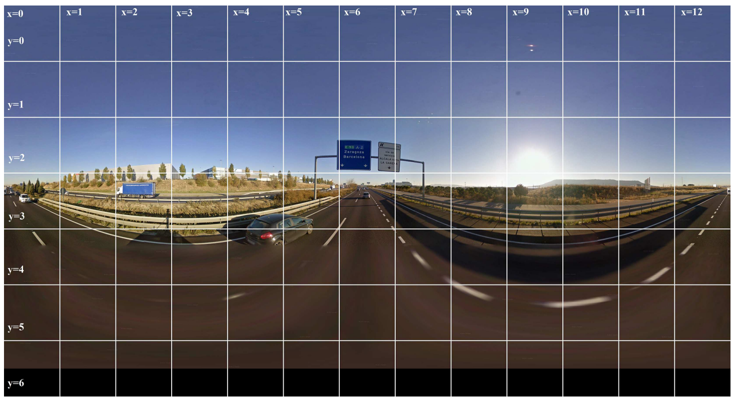
(a) Zoom=1



(b) Zoom=2



(c) Zoom=3



(d) Zoom=4

Figure 5.6: Different zoom levels for a panoramic view

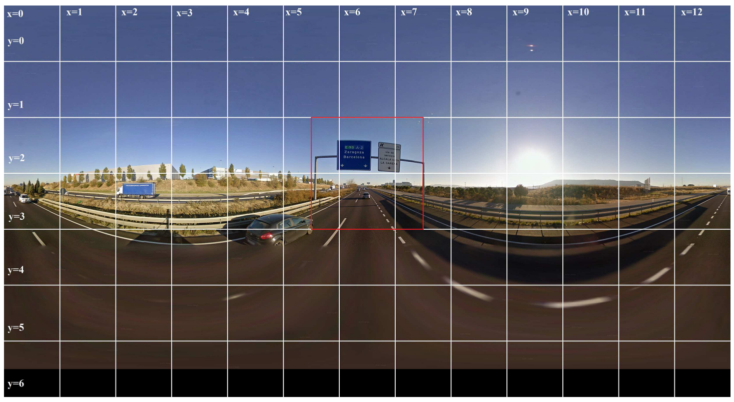


Figure 5.7: Region of interest in the panoramic view (red)

Algorithm 5.1 Image capture

```
1: Let $(LAT_0, LON_0)$ be the initial coordinates, $A = [(LAT_{min}, LON_{min}), (LAT_{max}, LON_{max})]$
   the area where the images are to be extracted and $M$ an empty array
2: Do $LAT = LAT_0$, $LON = LON_0$
3: Compute $PANOID$ from $(LAT, LON)$

4: procedure IMAGE CAPTURE($LAT$,$LON$,$PANOID$)
5:     if $LAT \geq LAT_{min}$ and $LAT \leq LAT_{max}$ and $LON \geq LON_{min}$ and $LON \leq LON_{max}$
   then
6:         if $(LAT, LON)$ is not in $M$ then
7:             Save $(LAT, LON)$ in $M$
8:             Extract panoramic view from $PANOID$ and crop to region of interest
9:             Compute $PANOID$ of the next frame
10:            Compute $(LAT, LON)$ of the next frame from $PANOID$
11:            Image capture($LAT$,$LON$,$PANOID$)
12:            Compute $PANOID$ of the previous frame
13:            Compute $(LAT, LON)$ of the previous frame from $PANOID$
14:            Image capture($LAT$,$LON$,$PANOID$)
15:        end if
16:    end if
17: end procedure
```

the system, composed of 10763 images. All the images have been obtained from street-level images of the Spanish road network, specifically from the roads shown in figure 5.8 (the training set from the roads shown in red and the test set from the roads shown in blue). The training set corresponds to images in the following roads: A-2, A-3, AP-36 and A-5 in the centre of Spain, A-6 in the northwest and A-8 in the north of the country, while the test set corresponds to some completely different roads, specifically the roads A-49 in the south of Spain and A-66 in the west. We have tried to choose a wide variety of situations, including different landscapes, weather conditions and times of the day.

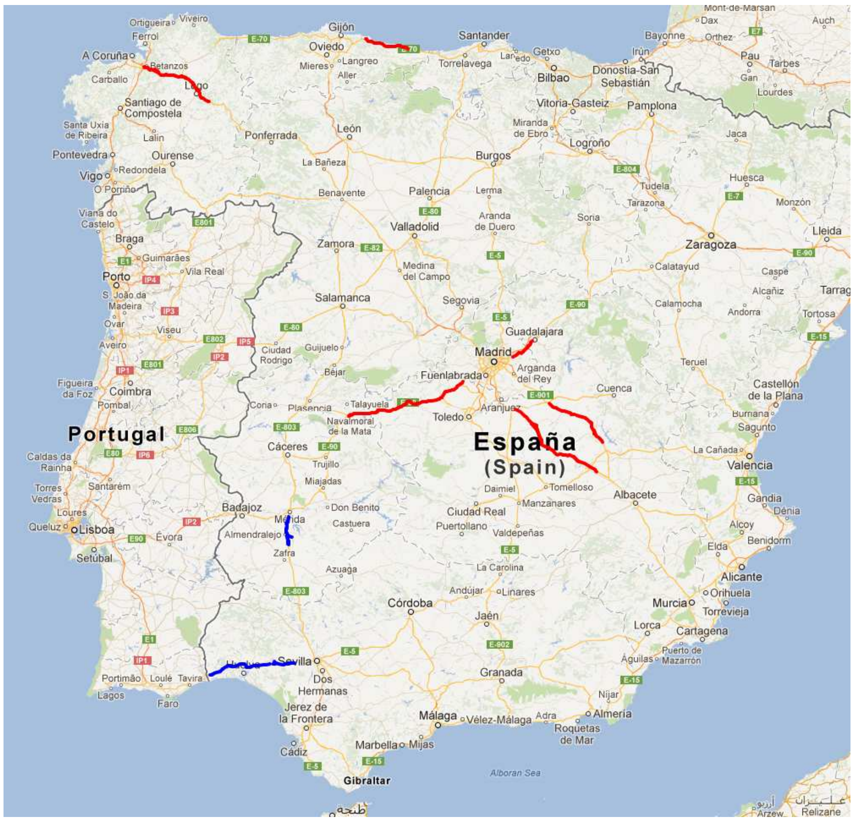


Figure 5.8: Roads from which the images have been obtained: training set (red) and test set (blue)

As it was explained in the introduction of the chapter, there are two kinds of traffic panels, those with blue background and those with white background. They can be located above the road and on the right margin of the road. Table 5.1 shows the number of panels of each type in both train and test sets. Since there can be several samples of each panel taken at different distances, we also show the number of images.

			Train		Test	
			Panels	Images	Panels	Images
Positives	Lateral	Blue	314	613	84	480
		White	35	68	24	87
	Upper	Blue	79	167	45	164
		White	81	199	32	123
Negatives			-	4467	-	9909
Total			509	5514	185	10763

Table 5.1: Number of panels and images in the dataset.

5.3 Traffic panels detection using visual appearance

The idea of the proposed system is to apply the text detection and recognition algorithm only on those images in which there is a traffic panel present in order to increase the efficiency of the system. For this purpose, it has been developed a method that detects the presence of traffic panels in the images. It is based on representing the images using a Bag of Visual Words (BOVW) approach [Csurka et al., 2004]. We have chosen this technique since it has become one of the the most popular in terms of classifying images. In this chapter, we want to prove that BOVW is suitable to model traffic panels despite the challenge that supposes their immense variability, and we want to show that geometrical characteristics are not needed to be added for the searched purpose. The diagram of blocks of the proposed method to detect the presence of a panel in an image is shown in figure 5.9. For both training and test images, the BOVW technique is applied only over certain areas of the image given by blue and white color masks. We will explain later in this section the reason why we are applying these color masks. But firstly, we are going to explain the BOVW technique.

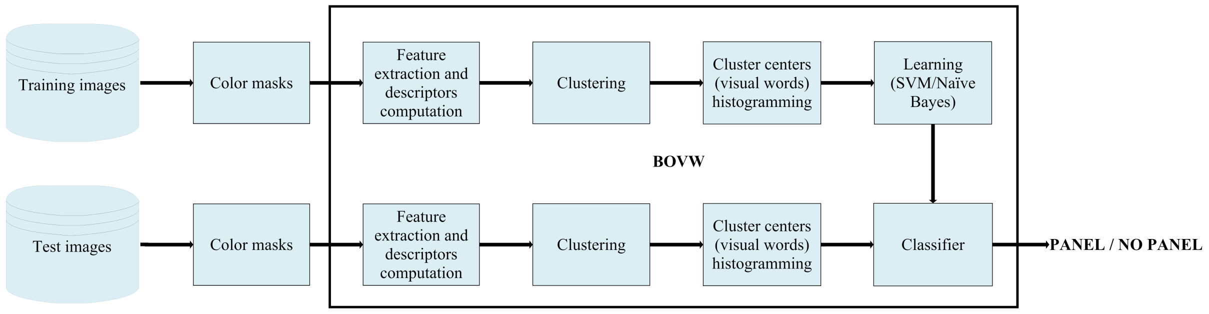
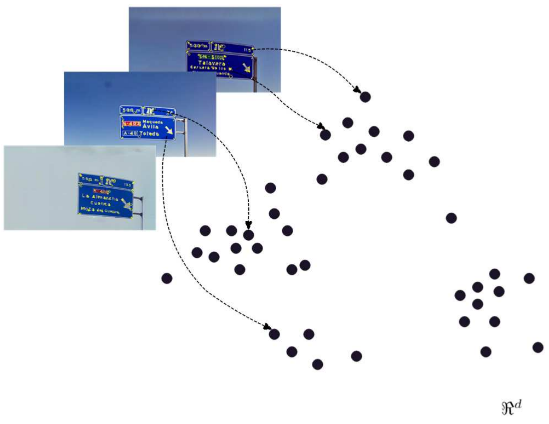

Training images
Color masks
Feature extraction and descriptors computation
Clustering
Cluster centers (visual words) histogramming
Learning (SVM/Naïve Bayes)
BOVW
Test images
Color masks
Feature extraction and descriptors computation
Clustering
Cluster centers (visual words) histogramming
Classifier
PANEL / NO PANEL

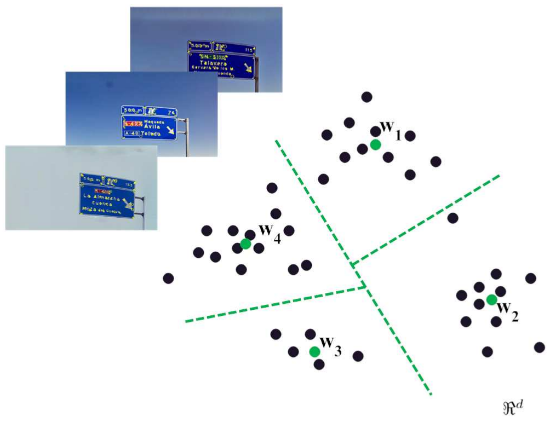
Figure 5.9: Traffic panels detection

The BOVW method stems from text analysis wherein a document is represented by word frequencies without regard to their order. These frequencies are then used to perform document classification. The BOVW approach to image representation follows the same idea. The visual equivalent of words are local image features. Therefore, the BOVW technique models an image as a sparse vector of occurrence counts of vocabulary of local image features. In other words, it translates a very large set of high-dimensional local descriptors into a single sparse vector of fixed dimensionality (a histogram) across all images. A schema that illustrates how this technique works is shown in figure 5.10.

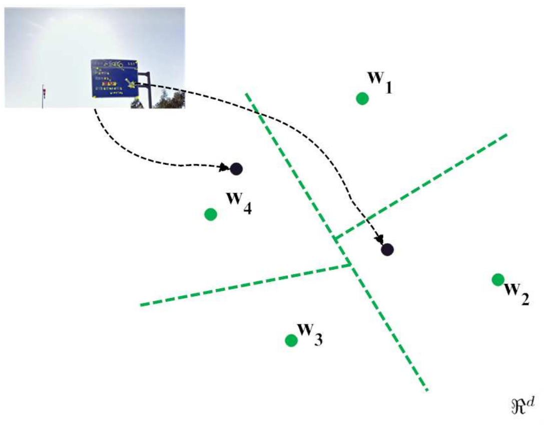
Firstly, features are extracted in the train images and converted into feature descrip-



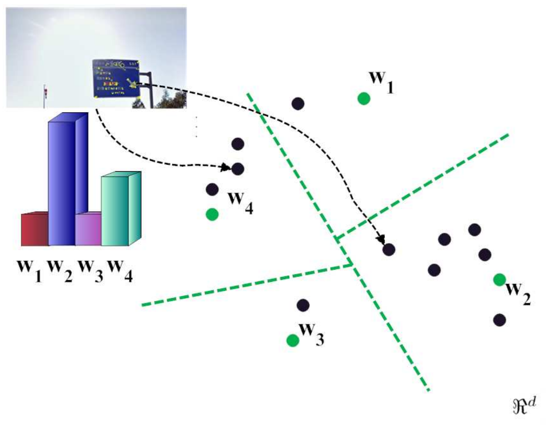
(a) Local visual features of train images



(b) Clustering into visual words



(c) Local visual features of test images



(d) Bag-of-visual-words histogram construction and word assignment

Figure 5.10: Visual vocabulary construction and word assignment

tors, which are high-dimensional vectors. Good descriptors should be able to handle intensity, rotation, scale and affine transformations. In this thesis, we have compared different descriptors of the state of the art, as it will be explained later in this section. In figure 5.10(a), the yellow circles denote local feature regions, while the black dots denote points in some feature space. Then, the sampled features are clustered in order to quantize the space into a discrete number of visual words using k-means clustering. The visual words are the cluster centers and can be considered as a representative of several similar local regions. The visual words are denoted with green circles in figure 5.10(b). The image can be represented by the histogram of the visual words, which counts how many times each of the visual words occurs in the image. To account for the difference in the number of interest points between images, the BOVW histogram is normalized to have unit L1 norm. The classes or categories of the input train images are learned by a classifier. In this thesis, we compare two classifiers: Support Vector Machines (SVM) [Cortes and Vapnik, 1995] and Naïve Bayes [Lewis, 1998].

SVM performs classification by constructing a N-dimensional hyperplane that optimally separates the data into two categories. The goal of SVM modeling is to find the optimal hyperplane that separates clusters of data in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other side of the plane. The descriptors near the hyperplane are the support vectors and the distance between the support vectors is called the margin. The optimum separation is achieved by the hyperplane that maximizes the margin, since in general the larger the margin is, the lower the generalization error of the classifier is.

On the other hand, the basic assumption of the Naïve Bayes model is that each category has its own distribution over the visual vocabulary, and that the distributions of each category are observably different. Suppose V is the number of visual words. Let be each image represented by $m = [m_1, m_2, \dots m_N]$, where m_i is a V-dimensional vector whose i^{th} component equals to one and all the other components equal to zero, meaning that m_i belongs to cluster i^{th}. Let c represent the category of the image. Given a collection of training examples, the Naïve Bayes classifier learns the different distributions for different categories. The classification decision is made by (5.1), which finds the class c that maximizes the posterior probability $p(c|m)$.

$$c_{MAP} = \arg\max_c p(c|m) \tag{5.1}$$

Applying the Bayes rule, (5.1) can be expressed as in (5.2).

$$c_{MAP} = \arg\max_c \frac{p(m|c)p(c)}{p(m)} \tag{5.2}$$

$p(m)$ can be dropped out, and assuming that the distributions on each category are independent, (5.2) reduces to (5.3).

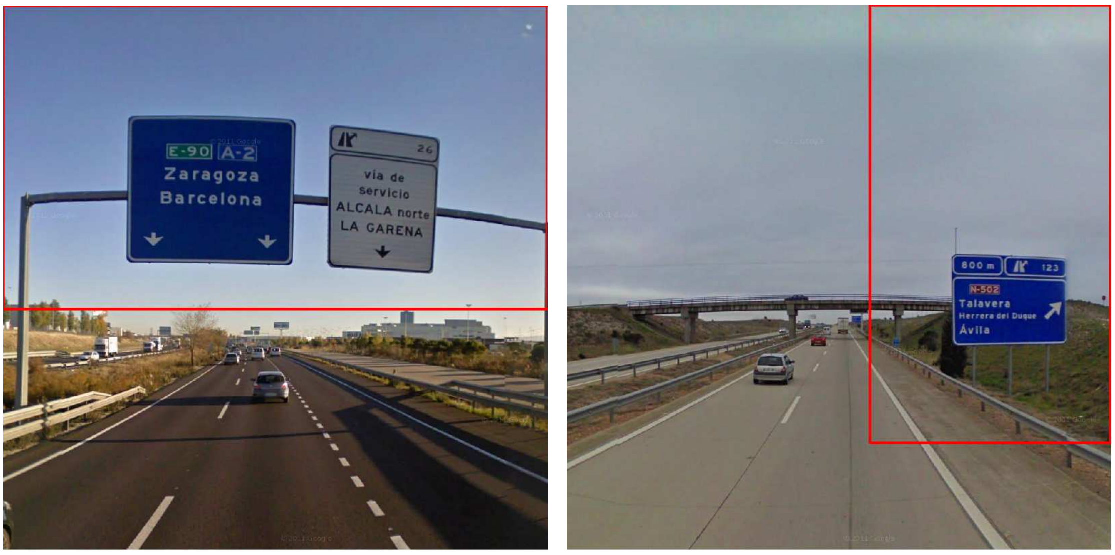
$$c_{MAP} = \arg\max_c p(c) \prod_{n=1}^{N} p(m_n|c) \tag{5.3}$$

$p(c)$ is the prior probability of class c.

Given a test image, the nearest visual word is identified for each of its features using the Euclidean distance between the cluster centers (visual words) and the input descriptors (figure 5.10(c)). A bag-of-visual-words histogram is computed to represent the whole image (figure 5.10(d)) and the classification decision is made by the classifier previously trained, either SVM or Naïve Bayes.

This has been the explanation of how BOVW is applied. Now, we are going to show how the first step of the BOVW technique (the feature extraction) is carried out.

Since the traffic panels are located above the road or on the right side, two independent regions of interest are applied on the images. These regions are shown in figure 5.11. Feature extraction, training and testing is done separatedly on each region of interest.



(a) Upper region of interest (b) Lateral region of interest

Figure 5.11: Regions of interest on the images

The features are extracted at some interest points, which are obtained using the Harris-Laplace salient point detector [Mikolajczyk and Schmid, 2001]. It uses a Harris corner detector and subsequently the Laplace operator for scale selection. Initially, it was tried to extract the keypoints in the whole images, but the results in terms of panel detection were very poor, as the number of false positives was too large. Therefore, in order to increase the robustness and efficiency of the proposed algorithm by minimizing the number of false positives, the local features are extracted only on those regions of the images which are mainly blue or white, as the traffic panels can have blue or white background. A method that detects blue pixels and white pixels in the images has been developed. We propose to detect the blue regions in the image as a combination of three independent methods using a logical AND operation as in (5.4), where the two first methods have been proposed by other authors but the third one is a proposal that we are making in this thesis, as well as the combination of the three methods.

$$BlueMask = g_1(x,y)\ AND\ g_2(x,y)\ AND\ g_3(x,y) \tag{5.4}$$

$g_1(x,y)$ is computed using (5.5) as it is proposed in [Kulkarni, 2012]. $R(x,y)$ is the red channel of the image and $T_r = 90$ is the optimum value according to the source article. This method has been proved to be really useful to discard the blue regions corresponding to the sky, while keeping the blue regions corresponding to the panels, which are typically darker than the sky. On the other hand, this method has the disadvantage that it is not able to reject dark regions in the image (black, gray, dark colors). This is solved using the next two methods.

$$g_1(x,y) = \begin{cases} 255 & \text{if } R(x,y) \leq T_r \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

On the other hand, $g_2(x,y)$ is computed using (5.6) as it is proposed in [Gómez-Moreno et al., 2010]. $H(x,y)$ is the Hue component of the image and $T_1 = 200°$ and $T_2 = 280°$ are the optimum values of the thresholds according to the authors. Unlike the previous method, this one is not able to distinguish between the blue regions in the sky and the blue regions in the panels, and it is not able to discard white regions in the image, but it is very useful to reject colors whose tonality is completely different to blue, like green, red or orange.

$$g_2(x,y) = \begin{cases} 255 & \text{if } H(x,y) \geq T_1 \ and \ H(x,y) \leq T_2 \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

Finally, our proposal, apart from (5.4), consists of computing $g_3(x,y)$ using (5.7), which applies the Otsu's segmentation method [Otsu, 1979] on the image obtained by subtracting the blue color component $B(x,y)$ from the red color one $R(x,y)$. The Otsu's method reduces the input image to a binary image, assuming that the input image contains two classes of pixels or a bi-modal histogram. It computes the optimum threshold that separates both classes so that their intra-class variance is minimal. Unlike the first method, this one is not able to discard the blue regions that correspond to the sky, but it improves the performance of the first method by rejecting dark regions in the image and it improves the performance of the second method by rejecting white regions in the image.

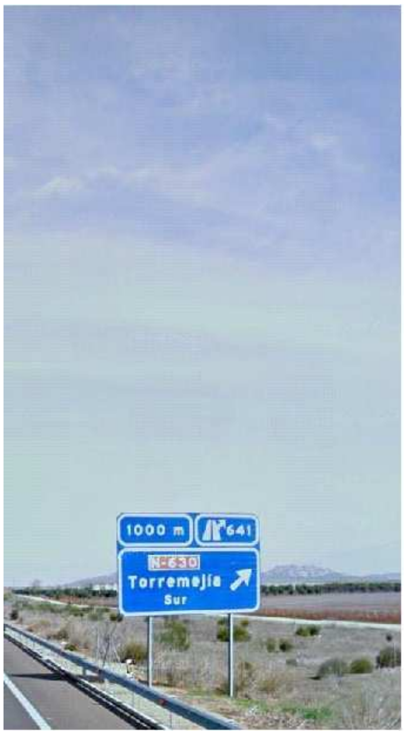
$$g_3(x,y) = Otsu(|R(x,y) - B(x,y)|) \tag{5.7}$$

Figure 5.12 shows the result of applying this blue color detection method in (5.4) on two images (a positive and a negative sample).

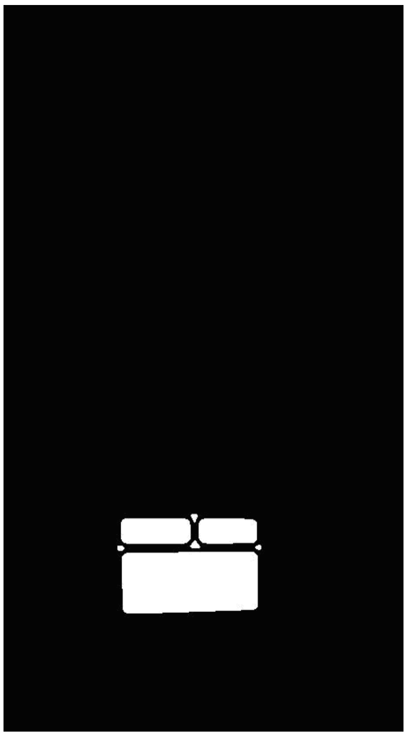
On the other hand, the method to detect white regions in the image is based on the Maximally Stable Extremal Regions method (MSER) [Matas et al., 2002], which is a region detector that allows to detect bright-on-dark regions in the image. Figure 5.13 shows an example of applying this white color detection method on an image with a traffic panel and on an image without any panel.

A comparison of different grey-based and color-based descriptors has been carried out. Specifically, the following descriptors have been used: SIFT [Lowe, 1999], C-SIFT [Abdel-Hakim and Farag, 2006], Hue-SIFT [van de Weijer et al., 2006], RGB-SIFT [van de Sande et al., 2010], Hue Histogram [van de Weijer et al., 2006] and Transformed Color Histogram (TCH) [van de Sande et al., 2010]. They have been computed using the ColorDescriptor library [2]. Results will be shown in section 5.6.

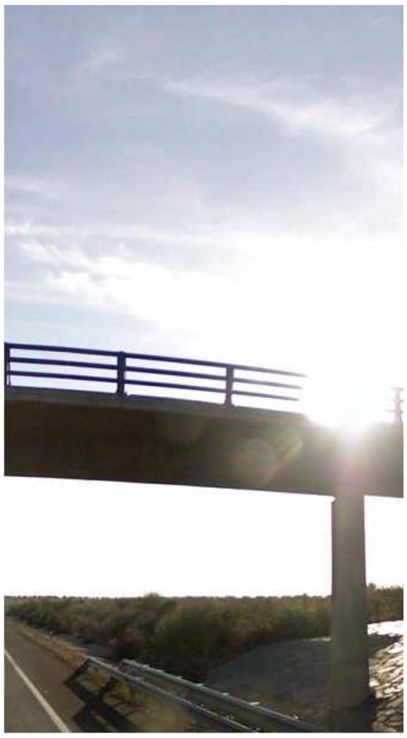
[2]`http://www.colordescriptors.com/`



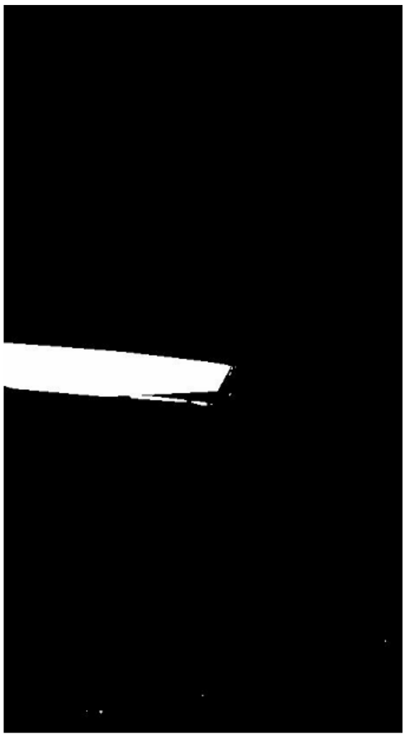
(a) Positive sample



(b) Blue color detection

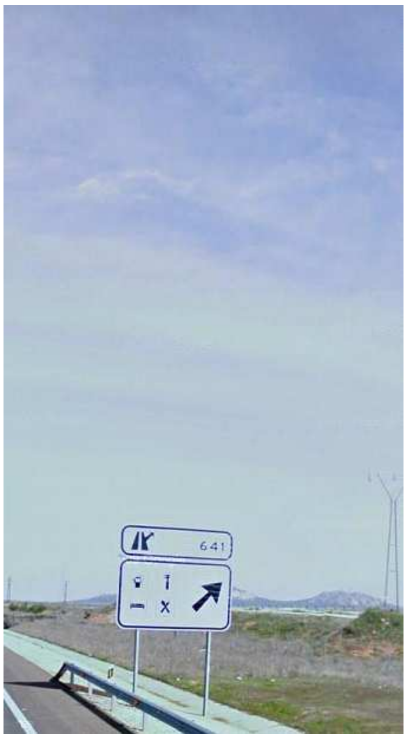


(c) Negative sample

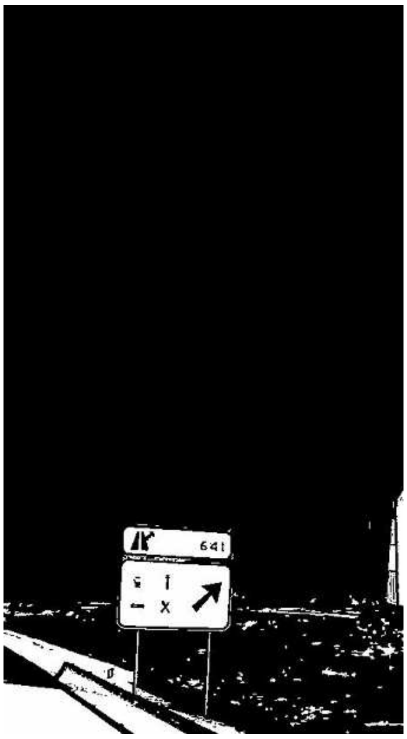


(d) Blue color detection

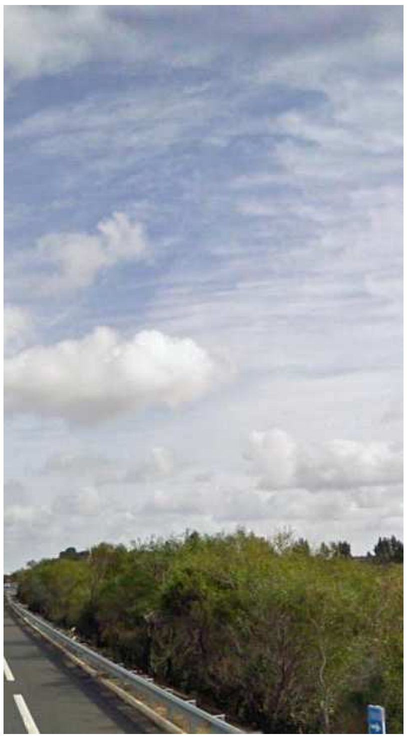
Figure 5.12: Blue color detection



(a) Positive sample



(b) White color detection



(c) Negative sample

(d) White color detection

Figure 5.13: White color detection

5.4 Text detection and recognition in traffic panels

If the previous method finds that there is a traffic panel in an image, the text location and recognition method explained in the previous chapters is applied on the image. However, some modifications have been proposed in order to increase the efficiency and reduce the number of false positives. Instead of applying the text location method in the whole image, it is done only on those areas of the image given by the blue and white color masks, like those shown in figures 5.12 and 5.13. Previously, the holes of the mask regions are filled in.

Then, character and word recognition is applied. The character recognizer described in the previous chapter was developed to recognize letters and digits. However, traffic panels contain not only words and numbers, but also symbols such as direction arrows and petrol station indications. Therefore, the system has been modified to recognize also this kind of symbols in the following way. Some of the most common symbols that appear in traffic panels have been chosen and several samples for each one have been added to the training set of the character recognizer. The chosen symbols are shown in figure 5.14.

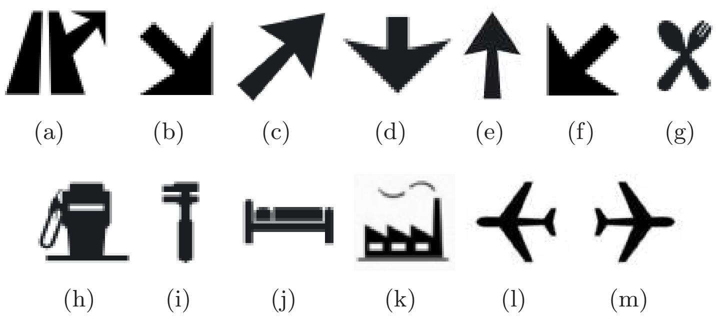


Figure 5.14: Considered symbols

The character recognizer may fail when panels are far, as text is small and difficult to segmentate and recognize. However, it is not necessary to recognize all the characters perfectly. They are just an estimation, because a word recognizer is applied later. The word recognizer is based on a unigram probabilistic language model that constrains the output of the character recognizer to a set of meaningful words. The model used in the previous chapter to recognize single words in natural images was based on the BNC, which is a representation of the English language. However, in this case, we are not recognizing English words, but text that appears in Spanish traffic panels. Therefore, instead of using the BNC, we use a dictionary of words that includes all the words that the system is able to recognize, that is, name of cities, places and other common words that typically appear in traffic panels, such as “cambio de sentido”, “via de servicio” or “centro comercial”. However, we do not have any information available on the frequency of each word, so it is not possible to compute the prior probabilities of the words. Therefore, we are assuming equal prior probability for all the words.

In order to increase the effectiveness of the recognition algorithm, we make use of a Web Map Service (WMS) to reduce the size of the dictionary to a limited geographical area, *i.e.* to the nearest places. A WMS is a standard protocol for serving georeferenced map images and data that are generated by a map server using data from a database. The specification was developed and first published by the Open Geospatial Consortium (OGC) in 1999.

The service used is in this thesis is provided by the project Cartociudad [3], which is an information system based on an official database of the Spanish road network, including any kind of routes, highways, urban thoroughfares and streets. It is supported by different public state Spanish institutions and it is updated every short time. At the moment, Cartociudad covers almost the whole Spanish territory, as it is shown in figure 5.15.

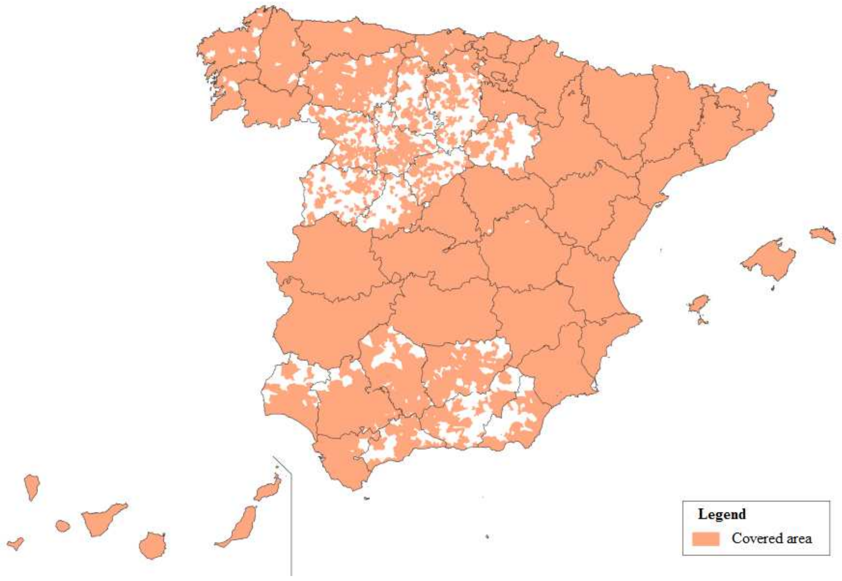


Figure 5.15: Area covered by Cartociudad

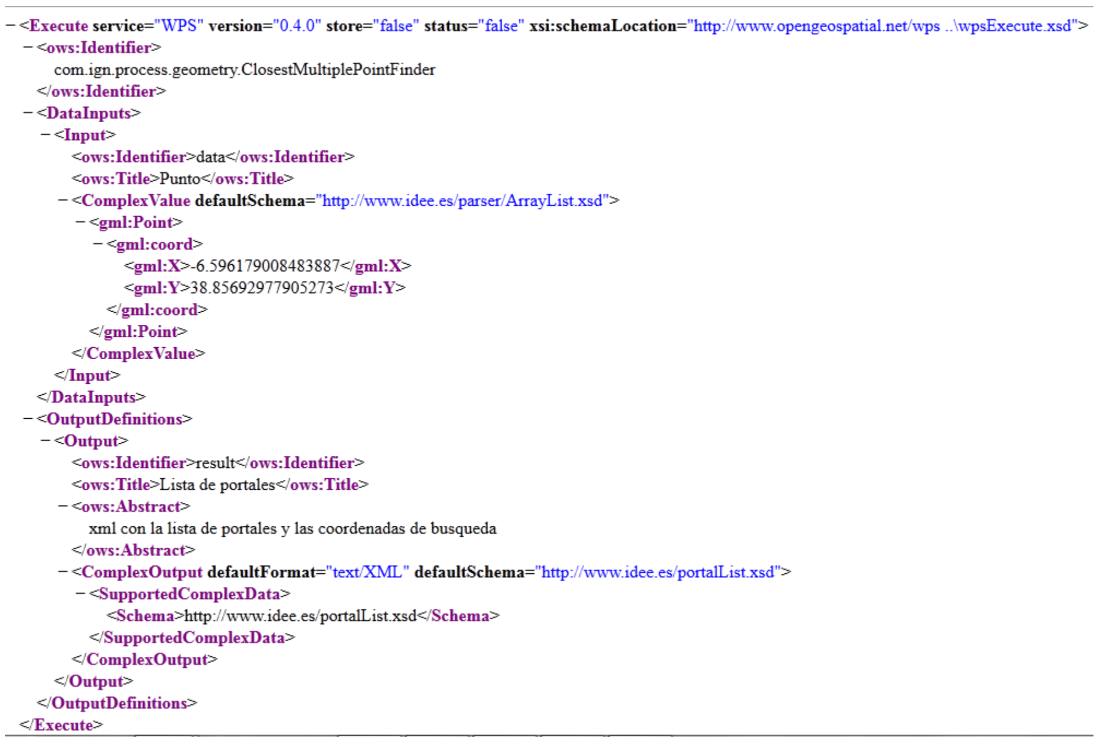
Cartociudad provides four types of services, which fulfill the OGC standard protocols:

- Map service. It allows the visualization of the cartography.
- Cached map service. Similarly to the previous service, it provides map images but making use of a cache of static images so the service is quicker.
- Gazetter service. It allows to search for street addresses, ZIP codes, postal districts, municipalities, provinces and autonomous communities.
- Geoprocessing service. It includes operations such as route planning, computation of areas of influence and reverse geocoding.

In this thesis, we make use of the last service, specifically the reverse geocoding service, which allows to get certain geographic data such as street addresses, names of roads, postal codes, milestones, municipalities, provinces and autonomous communities, from geographic coordinates (latitude and longitude). These coordinates are known for every image, as it was explained in section 5.2, where it was shown that every image has an associated pair of coordinates (LAT, LON). The reverse geocoding service works as follows. A request to the Cartociudad server is done in the way of a POST method, which is supported by the HTTP protocol. It is designed to request that a web server accepts the data enclosed in the request message's body for storage or analysis. In this case, the message is parsed in XML format. The request is done to the following URL: `http://www.cartociudad.es/wps/WebProcessingService`. The petition returns a document also parsed in XML format in which different features are stated for the position

[3]`http://www.cartociudad.es/`

under request. The input file includes the coordinates in decimal format in the tags *<gml:X></gml:X>* for the longitude and *<gml:Y></gml:Y>* for the latitude. An example is shown in figure 5.16. The output file includes a series of data such as the name of the province (*<province></province>*), the town (*<municipio></municipio>*), the type of road (*<tipoVia></tipoVia>*), the name of the road (*<nombreVia></nombreVia>*) and the milestone (*<rotulo></rotulo>*). An example is shown in figure 5.17.

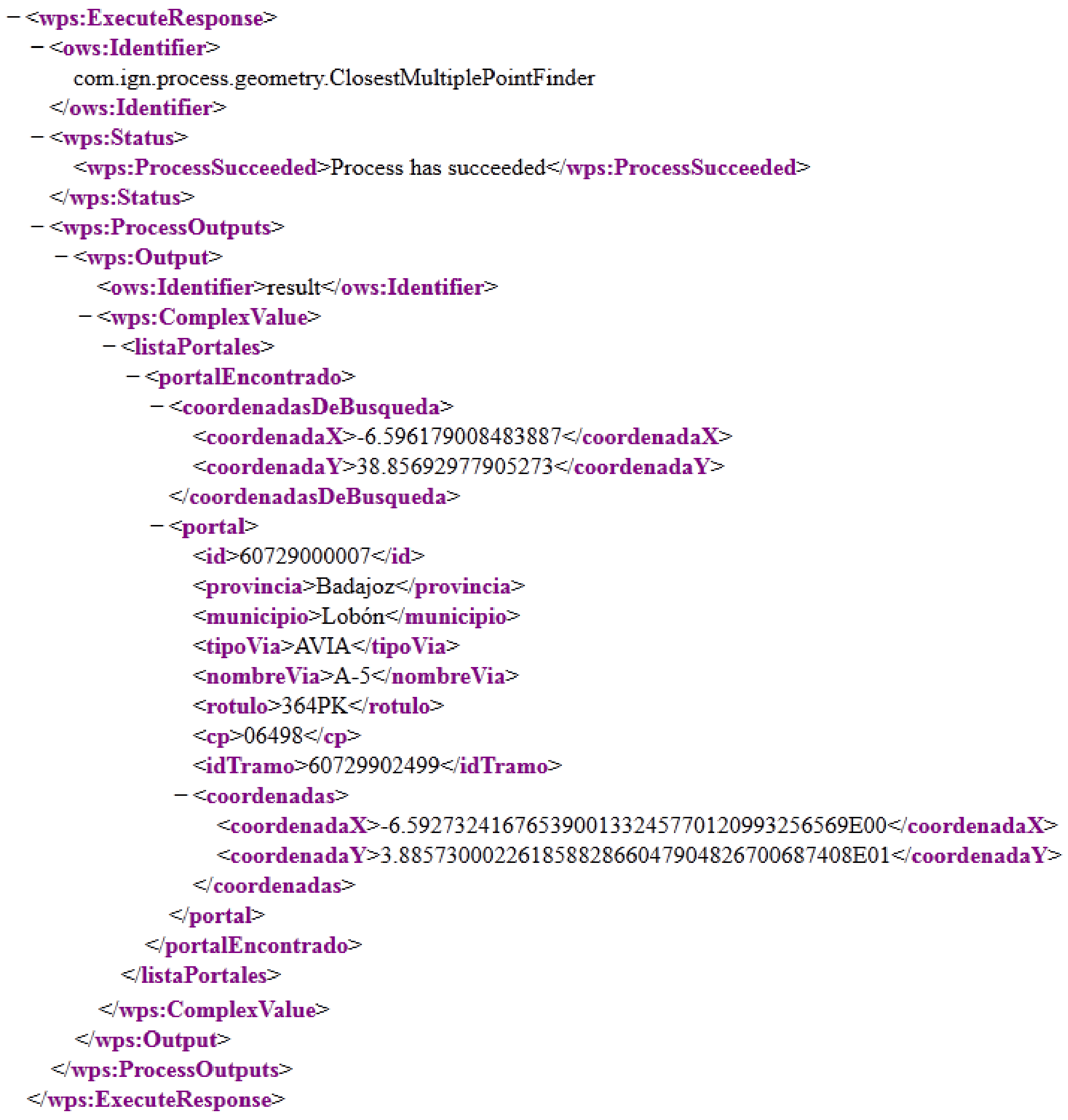


```
– <Execute service="WPS" version="0.4.0" store="false" status="false" xsi:schemaLocation="http://www.opengeospatial.net/wps ..\wpsExecute.xsd">
  – <ows:Identifier>
      com.ign.process.geometry.ClosestMultiplePointFinder
    </ows:Identifier>
  – <DataInputs>
    – <Input>
        <ows:Identifier>data</ows:Identifier>
        <ows:Title>Punto</ows:Title>
      – <ComplexValue defaultSchema="http://www.idee.es/parser/ArrayList.xsd">
        – <gml:Point>
          – <gml:coord>
              <gml:X>-6.596179008483887</gml:X>
              <gml:Y>38.85692977905273</gml:Y>
            </gml:coord>
          </gml:Point>
        </ComplexValue>
      </Input>
    </DataInputs>
  – <OutputDefinitions>
    – <Output>
        <ows:Identifier>result</ows:Identifier>
        <ows:Title>Lista de portales</ows:Title>
      – <ows:Abstract>
          xml con la lista de portales y las coordenadas de busqueda
        </ows:Abstract>
      – <ComplexOutput defaultFormat="text/XML" defaultSchema="http://www.idee.es/portalList.xsd">
        – <SupportedComplexData>
            <Schema>http://www.idee.es/portalList.xsd</Schema>
          </SupportedComplexData>
        </ComplexOutput>
      </Output>
    </OutputDefinitions>
  </Execute>
```

Figure 5.16: XML input file format to Cartociudad server

Therefore, instead of using an unique dictionary of words for the whole country, we have created a dictionary for every province in Spain, each one contains the names of all the municipalities in the province, and another dictionary that has a set of typical words that appear in traffic panels and do not depend on the geographical position, like “centro comercial”, “via de servicio” or “cambio de sentido”. The names of the capital cities of each province have been also added to the second dictionary, in order to deal with those situations in which a capital city of a province is referenced in a panel that is not located in the same province. The sizes of the dictionaries depend on the province itself, but each one is composed of several hundreds of words.

Given an input image with its associated pair of latitude and longitude coordinates, we make a request to the Cartociudad server using these coordinates as it has been shown above. Then, we use the name of the province included in the body of the returned file to choose the corresponding dictionary. As well as this, the dictionary that contains the common words is also used. Therefore, the language model used to recognize single words is partly based on a fixed dictionary and partly based on a dynamic dictionary that depends on the province where the image was taken.



```
- <wps:ExecuteResponse>
  - <ows:Identifier>
      com.ign.process.geometry.ClosestMultiplePointFinder
    </ows:Identifier>
  - <wps:Status>
      <wps:ProcessSucceeded>Process has succeeded</wps:ProcessSucceeded>
    </wps:Status>
  - <wps:ProcessOutputs>
    - <wps:Output>
        <ows:Identifier>result</ows:Identifier>
      - <wps:ComplexValue>
        - <listaPortales>
          - <portalEncontrado>
            - <coordenadasDeBusqueda>
                <coordenadaX>-6.596179008483887</coordenadaX>
                <coordenadaY>38.85692977905273</coordenadaY>
              </coordenadasDeBusqueda>
            - <portal>
                <id>60729000007</id>
                <provincia>Badajoz</provincia>
                <municipio>Lobón</municipio>
                <tipoVia>AVIA</tipoVia>
                <nombreVia>A-5</nombreVia>
                <rotulo>364PK</rotulo>
                <cp>06498</cp>
                <idTramo>60729902499</idTramo>
              - <coordenadas>
                  <coordenadaX>-6.5927324167653900133245770120993256569E00</coordenadaX>
                  <coordenadaY>3.8857300022618588286604790482670068740 8E01</coordenadaY>
                </coordenadas>
              </portal>
            </portalEncontrado>
          </listaPortales>
        </wps:ComplexValue>
      </wps:Output>
    </wps:ProcessOutputs>
  </wps:ExecuteResponse>
```

Figure 5.17: XML output file format from Cartociudad server

5.5 Text geolocalization

We want to estimate the approximated position of the detected text for two main reasons. Firstly, we want to see how our text detection and recognition algorithm performs as a function of the distance to the panel. Secondly, the aim of this chapter is to develop a system to make an inventory of road panels. For this purpose, it is necessary to compute the coordinates of the panels, because the geographical coordinates of the images correspond to the location where the images were taken by the vehicle and not to the location of the panels. The camera parameters are unknown and the system is monocular, so in principle it is not possible to compute the relative position of the objects in the image respect to the camera. However, we know that the size of the text contained in traffic panels is not constant, but it does not differ very much from one panel to another. We are presenting in this section a coarse method to localize the position of the detected text, since we do not need an accurate calculation of the coordinates of the panels for our purpose.

A function that converts from text height in pixels to depth distance in meters has been computed. The data to compute this function has been obtained from different panels of the training set estimating the distance from the vehicle to the panel by using the satellite image in Google Maps. Figure 5.18 shows the data points and the approximated function, which is quadratic as it was expected. This function has been obtained after fitting the data points to an exponential function using least squares. It is defined by (5.8), where d is the distance from the panel to the vehicle in meters, h is the mean text height in pixels (only letters and numbers, and not symbols, have been considered), $a = 81.66$ and $b = -0.06225$.

$$d = a \cdot e^{b \cdot h} \tag{5.8}$$

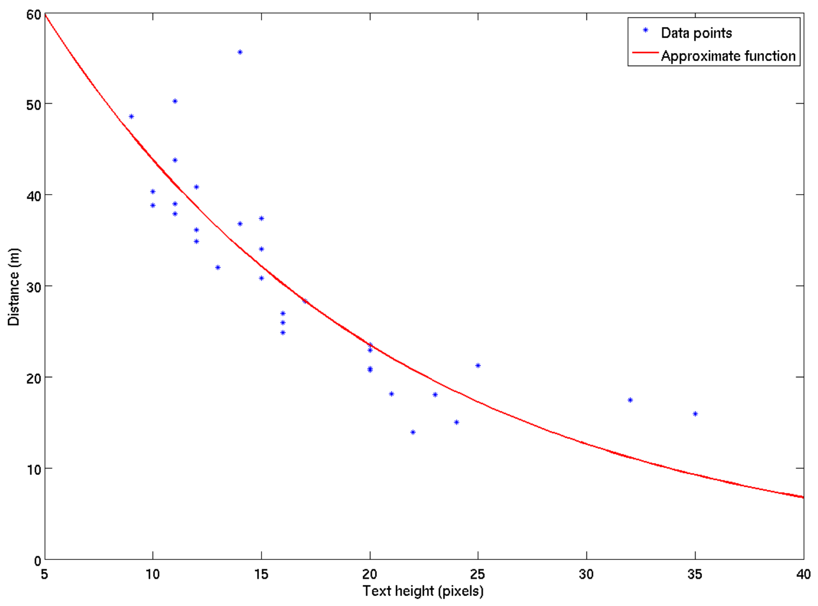


Figure 5.18: Function to convert from height (pixels) to distance (meters)

Once we have detected text in a certain image, we compute the mean height in pixels of the text and apply (5.8) to obtain the distance d from the panel to the vehicle. Then, in order to estimate the position of the panel, we need to compute the direction vector of the vehicle. We transform the latitude and longitude values of the vehicle in the current and in the following frame, (lat_1, lon_1) and (lat_2, lon_2) respectively, to UTM coordinates in meters, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ (refer to appendix C.1). The direction vector $\vec{v}$ and the angle α of the vector respect to the horizontal axis are computed using (5.9) and (5.10). α is always referenced to the positive horizontal semiaxis.

$$\vec{v} = (v_x, v_y) = (x_2 - x_1, y_2 - y_1) \tag{5.9}$$

$$\alpha = \arccos \frac{v_x}{\sqrt{v_x^2 + v_y^2}} \tag{5.10}$$

Finally, (5.11) is applied to compute the position of the traffic panel $P_3 = (x_3, y_3)$, using the geometry shown in figure 5.19. These are UTM coordinates, which can be easily transformed into latitude and longitude values applying the correspondent conversion formulas (refer to appendix C.2).

$$P_3 = (x_3, y_3) = (x_1 + d \cdot \cos\alpha, y_1 + d \cdot \sin\alpha) \tag{5.11}$$

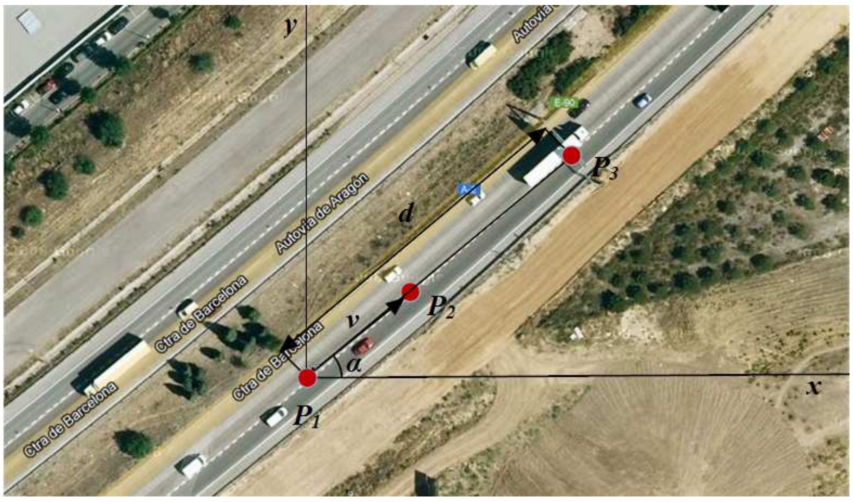


Figure 5.19: Geometry to estimate the position of the panel

5.6 Experimental results

5.6.1 Traffic panels detection

As it was stated in section 5.3, a comparison of different descriptors has been carried out in order to check which are the most suitable for the proposed application of traffic panels detection using visual appearance. Specifically, the following descriptors have been compared:

- Scale Invariant Feature Transform (SIFT) [Lowe, 1999]. Keypoints are located in the gray-level image by the Harris-Laplace detector. The descriptors at each keypoint are computed as a set of orientation histograms on 4×4 pixel neighborhoods. The orientation histograms are relative to the keypoint orientation. The histograms contains each 8 bins, and each descriptor contains a 4×4 array of 16 histograms around the keypoint. This leads to a feature vector with 128 elements. SIFT is invariant to light intensity changes and light intensity shift.

- Colored Scale Invariant Feature Transform (C-SIFT) [Abdel-Hakim and Farag, 2006]. It is similar to SIFT, but the keypoints and the descriptors are computed from the normalized channels of the opponent color space $\frac{O_1}{O_3}$ and $\frac{O_2}{O_3}$, defined as in (5.12). The dimensionality of the descriptor vector is 384 elements. C-SIFT is scale-invariant with respect to light intensity, but it is not shift-invariant.

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix} \tag{5.12}$$

- Hue Histogram [van de Weijer et al., 2006]. The histogram of the hue component of the HSV color space is computed at the keypoints. However, in order to deal with the unstability of the hue near the grey axis, the hue histogram is made more robust by weighing each sample of the hue by its saturation. The feature vector has 37 elements. Hue Histogram is scale-invariant and shift-invariant with respect to light intensity.

- Hue Scale Invariant Feature Transform (Hue-SIFT) [van de Weijer et al., 2006]. It is a concatenation of the Hue Histogram with the SIFT descriptor. The dimensionality of the feature vector is 165 elements. Similar to the Hue Histogram, the Hue-SIFT descriptor is scale-invariant and shift-invariant.

- RGB Scale Invariant Feature Transform (RGB-SIFT) [van de Sande et al., 2010]. SIFT descriptors are computed for every RGB channel independently and concatenated into a single vector of 384 dimensions. This descriptor is scale-invariant, shift-invariant and invariant to light color changes and shift.

- Transformed Color Histogram (TCH) [van de Sande et al., 2010]. RGB channels are normalized independently using the mean μ_i and standard deviation σ_i of each channel computed over the area under consideration (the keypoints, which are located in the gray-level image by the Harris-Laplace detector). The normalization follows the expression (5.13). The histograms of each normalized channel are computed and concatenated into a vector of 45 elements. TCH is scale-invariant and shift-invariant with respect to light intensity and light color.

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} \frac{R-\mu_R}{\sigma_R} \\ \frac{G-\mu_G}{\sigma_G} \\ \frac{B-\mu_B}{\sigma_B} \end{pmatrix} \tag{5.13}$$

Only with SIFT, Hue Histogram and TCH, it has been possible to successfully cluster the descriptors and train the classifier, because convergence was not reached using the other descriptors.

Tables 5.2-5.5 show the results for each defined class: blue-background lateral panels, blue-background panels above the road, white-background lateral panels and white-background panels above the road. Table 5.6 shows the results for all the panels on the right of the road regardless of their color, while table 5.7 shows the results for the panels above the road regardless of the color. The results are shown in terms of panel detection rate, sensitivity and specificity. The panel detection rate is the percentage of correctly detected panels. Usually, a panel appears in several images at different distances. In case the algorithm detects a panel in at least one of the images where it appears, we count it as a correct detection. Therefore, the panel detection rate is computed in multi-frame. On the other hand, sensitivity and specificity are computed in single-frame. They are defined as in (5.14) and (5.15). They measure the system's ability to identify positive and negative samples, that is, if a panel is present or not in an image and if it has been detected or not, regardless of if the same panel appears in previous or subsequent frames.

$$Sensitivity = \frac{TP}{TP\ +\ FN} \tag{5.14}$$

$$Specificity = \frac{TN}{TN\ +\ FP} \tag{5.15}$$

TP stands for the number of true positives, FN stands for the number of false negatives, TN is the number of true negatives and FP is the number of false positives. The sensitivity measure relates to the system's ability to identify positive samples, while the specificity relates to the system's ability to identify negative samples. In order to join both measures into one, the f-measure is defined in (5.16).

$$f = \frac{Sensitivity + Specificity}{2} \tag{5.16}$$

It can be seen that the best results are obtained for the color descriptors, being TCH the best one. The panel detection rate is above 95% for the four situations under study and the value of the f-measure is the highest in all cases except for blue panels located on the side of the road, although it is very close to the highest value which is obtained with the Hue Histogram descriptor. However, the highest value of the specificity measure is achieved in most cases for the SIFT descriptor. It means that the number of false positives for this descriptor is very low. Nevertheless, the sensitivity is much lower for SIFT respect to the other descriptors. It means that the number of false negatives is very high respect to the number of true positives. In other words, the classifier trained with the SIFT descriptor categorizes most of the images as if there is not any panel present in the image. That is the reason why the detection rate for SIFT is so low respect to the other descriptors.

A study on how the number of clusters when constructing the visual vocabulary affects the performance of the system, has been carried out. Figure 5.20 shows how the f-measure,

Descriptor	Panel detection rate	Sensitivity	Specificity	f
SIFT	67.86%	0.2500	0.9192	0.5846
Hue Histogram	94.05%	0.6625	0.8782	0.7704
Transformed Color Histogram	98.81%	0.6042	0.9253	0.7674

Table 5.2: Detection for blue lateral panels

Descriptor	Panel detection rate	Sensitivity	Specificity	f
SIFT	86.66%	0.5366	0.9789	0.7577
Hue Histogram	100%	0.9512	0.8438	0.897489
Transformed Color Histogram	100%	0.8963	0.9536	0.9300

Table 5.3: Detection for blue upper panels

Descriptor	Panel detection rate	Sensitivity	Specificity	f
SIFT	45.83%	0.1724	0.9264	0.5494
Hue Histogram	58.33%	0.3563	0.6107	0.4835
Transformed Color Histogram	95.83%	0.6552	0.5079	0.5815

Table 5.4: Detection for white lateral panels

Descriptor	Panel detection rate	Sensitivity	Specificity	f
SIFT	75%	0.3740	0.9542	0.6641
Hue Histogram	93.75%	0.8293	0.6827	0.7560
Transformed Color Histogram	96.88%	0.7480	0.8998	0.8238

Table 5.5: Detection for white upper panels

Descriptor	Panel detection rate	Sensitivity	Specificity	f
SIFT	62.96%	0.3304	0.8511	0.5907
Hue Histogram	86.11%	0.7625	0.5385	0.6505
Transformed Color Histogram	98.15%	0.7464	0.4772	0.6118

Table 5.6: Detection including all the lateral panels

Descriptor	Panel detection rate	Sensitivity	Specificity	f
SIFT	81.82%	0.5708	0.9394	0.7551
Hue Histogram	97.40%	0.9292	0.5975	0.7634
Transformed Color Histogram	98.70%	0.8821	0.8817	0.8819

Table 5.7: Detection including all the upper panels

for each type of panel and for each descriptor, varies as a function of the size of the vocabulary. For blue lateral panels, it can be seen in figure 5.20(a) that the value of f remains constant for Hue Histogram, while it increases rapidly from 25 to 300 visual words and then it tends to be asymptotic from 300 onwards, both for TCH and SIFT. Therefore, $k = 300$ seems to be the optimum size of the vocabulary for TCH and SIFT in blue lateral panels, because with a higher number of visual words the training is slower but the results obtained do not change drastically.

Similarly, the performance of the system for blue upper panels using Hue Histogram remains constant as the size of the vocabulary increases, while it increases quickly from 25 to 200 visual words and tends to an asymptote onwards when using TCH, as shown in figure 5.20(b). Surprisingly, the best results using SIFT are achieved using only 25 clusters.

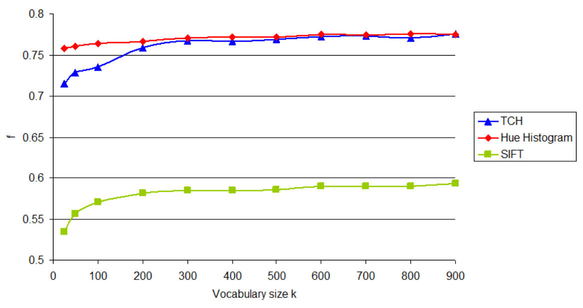
The performance of the system for white lateral panels tends to be constant regardless of the descriptor used, with small variations of 5% at highest, as shown in figure 5.20(c).

The number of visual words using TCH and SIFT in white upper panels does not affect drastically to the performance of the system, as shown in figure 5.20(d), but, on the contrary, it does for Hue Histogram, as the f-measure is low for less than 200 clusters and tends to be asymptotic from 200 onwards.

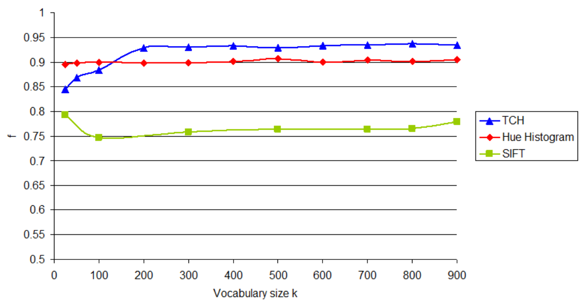
On the other hand, figure 5.21 shows how the size of the vocabulary affects the sensitivity. It can be seen that, when using TCH descriptor, the sensitivity tends to be higher while the number of clusters increases, except for white upper panels for which the sensitivity remains constant, and for white lateral panels for which the sensitivity decreases. However, the sensitivity is constant for blue panels when using Hue Histogram, but it decreases for white upper panels. When using SIFT descriptor, the sensitivity tends to be constant, although it slightly decreases for white panels.

Figure 5.22 shows how the size of the vocabulary affects the specificity. It tends to be constant when using TCH and Hue Histogram, except for white upper panels for which the specificity increases with the number of clusters, and for white lateral panels for which the specificity increases as the number of visual words gets higher. However, when using SIFT, the specificity increases while the number of clusters is higher.

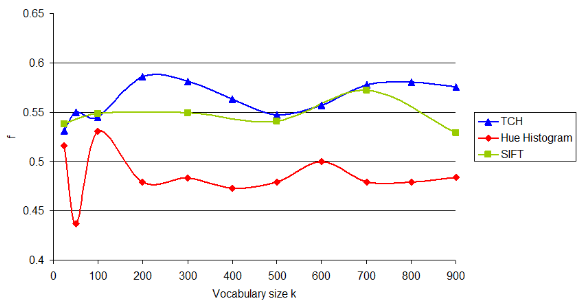
All the previous experiments have been carried out using a Naïve Bayes classifier. However, another classifier based on SVM with linear kernel has been tested. We have found that, in general, the number of false positives using SVM is much lower than using Naïve Bayes and, therefore, the specificity is higher. However, the number of false negatives (when the algorithm does not detect a panel but there is one in reality) is higher and consequently the sensitivity is lower than if a Naïve Bayes classifier is used. The panel detection rate is also lower and, in addition, we have seen that the computational time using SVM is much higher than using a Naïve Bayes classifier. Therefore, in this application



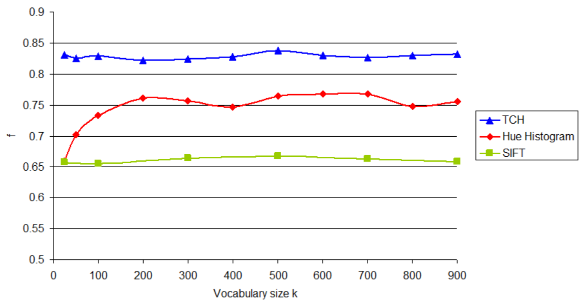
(a) Blue lateral panels



(b) Blue upper panels

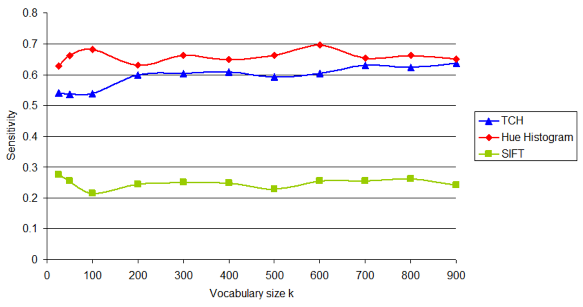


(c) White lateral panels

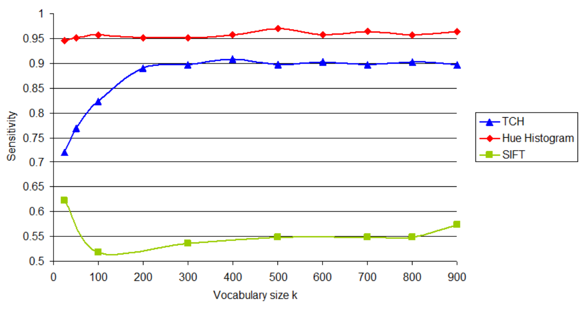


(d) White upper panels

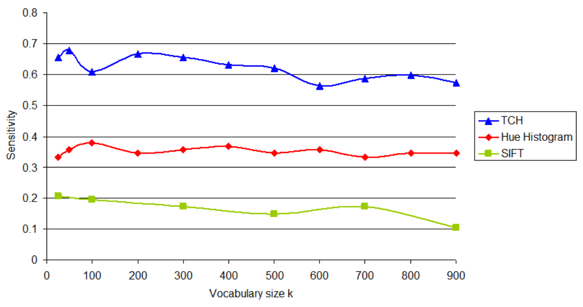
Figure 5.20: f-measure as a function of the size of the vocabulary



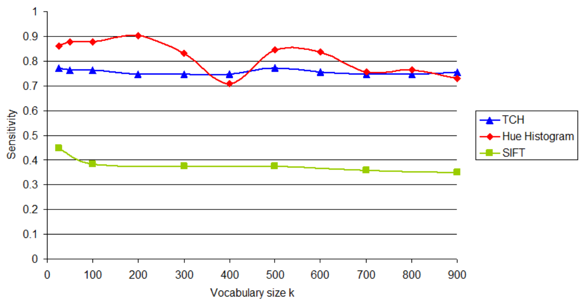
(a) Blue lateral panels



(b) Blue upper panels



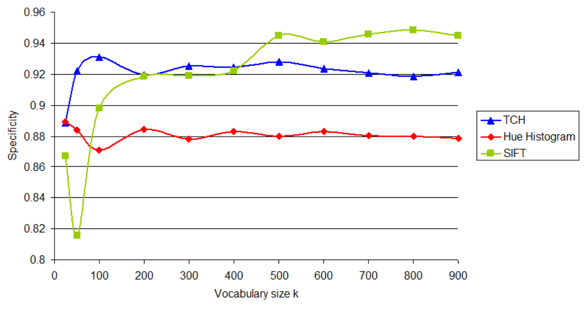
(c) White lateral panels

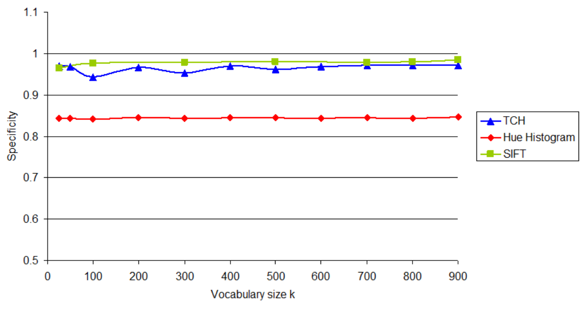


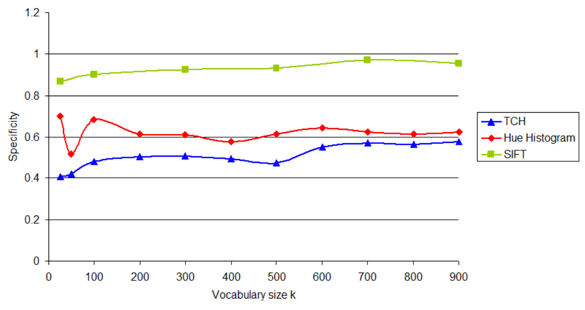
(d) White upper panels

Figure 5.21: Sensitivity as a function of the size of the vocabulary

it is preferred to use Naïve Bayes against SVM. As an example, the comparisons between Naïve Bayes and SVM using TCH for each kind of panel are shown in table 5.8-5.11.


(a) Blue lateral panels


(b) Blue upper panels


(c) White lateral panels

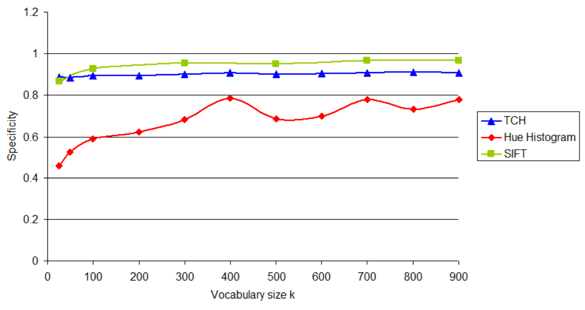

(d) White upper panels

Figure 5.22: Specificity as a function of the size of the vocabulary

Classifier	Panel detection rate	Sensitivity	Specificity	f
Naïve Bayes	98.81%	0.6042	0.9253	0.7674
SVM	90.48%	0.4167	0.9794	0.6980

Table 5.8: Effect of classifier for blue lateral panels

Classifier	Panel detection rate	Sensitivity	Specificity	f
Naïve Bayes	100%	0.8963	0.9536	0.9300
SVM	97.78%	0.6524	0.9968	0.8246

Table 5.9: Effect of classifier for blue upper panels

Classifier	Panel detection rate	Sensitivity	Specificity	f
Naïve Bayes	95.83%	0.6552	0.5079	0.5815
SVM	4.17%	0.0115	0.9831	0.4973

Table 5.10: Effect of classifier for white lateral panels

Classifier	Panel detection rate	Sensitivity	Specificity	f
Naïve Bayes	96.88%	0.7480	0.8998	0.8238
SVM	96.88%	0.4798	0.9804	0.7300

Table 5.11: Effect of classifier for white upper panels

5.6.2 Text detection and recognition

It has been analyzed a total of 145 kilometers of two different highways of the Spanish road network, as shown in figure 5.8. In this stretch, there are 185 panels of which 77 are panels above the road and 108 are traffic panels located at the right side of the road. Typically, there are several samples for every panel, because each panel usually appears in different frames at different distances. The detection and recognition have been carried out for every frame independently. Therefore, the text detection and recognition rates shown in tables 5.12-5.18 have been computed in single-frame. We show the results as a function of the distance from the vehicle to the panel in order to show how the distance to the panels affects to the algorithm performance. We have defined three ranges: short distance, when the panel is less than 40 meters far; medium distance, when it is in the range 40-70 meters; and long distance, when the panel is further than 70 meters, approximately. In addition, the detection and recognition rates have been computed separatedly for words, numbers and symbols. In general, the nearer the panel is, the better the performance is.

The best performance is achieved for words, being the detection and recognition rates above 90% and close to 80%, respectively, when the panel is at short distance, as shown in table 5.18. The performance hardly decreases when the panel is at medium distance (up to 70 meters far). However, the detection rate for numbers and symbols is lower compared to words because the text detection method, as explained in previous chapters, focuses on detecting text lines that has at least 3 elements, thus numbers and symbols that may appear isolated in the panel could not be detected. A lower threshold than 3 could be used for this specific application, although the number of false positives may increase, but what we wanted to show in this chapter is that the proposed text detection and recognition algorithm trained with a concrete dataset (ICDAR 2003) can be generalized to any other situation like the one shown in this chapter, and it achieves a reliable performance.

From the tables it can also be seen that the recognition rate for symbols remains above 70% even when the panel is at long distance. This is due to the fact that, in general, symbols in the traffic panels are typically bigger than letters and numbers, thus it is easier to be recognized even when the panel is further than 70 meters.

Another conclusion that can be extracted from the tables below is that, in general, the performance of the algorithm at medium distance is worse for panels located beside the road than for panels above the road, while it happens the opposite at very short distance. This is due to stitching and to deformations that appear at the margins of the image, both left and right and top and bottom, which affect the detection and recognition.

Some examples of the detection and recognition are shown in figures 5.23-5.40.

Data	Distance	Detection rate	Recognition rate
Words	Short	92.00%	75.36%
	Medium	76.47%	42.31%
	Long	34.62%	0%
Numbers	Short	68.85%	49.21%
	Medium	26.67%	12.50%
	Long	7.56%	0%
Symbols	Short	70.59%	75.00%
	Medium	62.79%	62.96%
	Long	50.00%	60.00%

Table 5.12: Text detection and recognition for blue lateral panels.

Data	Distance	Detection rate	Recognition rate
Words	Short	97.00%	70.10%
	Medium	97.83%	61.11%
	Long	80.00%	10.23%
Numbers	Short	83.94%	63.58%
	Medium	41.36%	51.90%
	Long	23.03%	11.43%
Symbols	Short	66.23%	90.20%
	Medium	83.33%	96.67%
	Long	52.94%	88.89%

Table 5.13: Text detection and recognition for blue upper panels.

Data	Distance	Detection rate	Recognition rate
Words	Short	75.51%	86.49%
	Medium	21.05%	100%
	Long	0%	0%
Numbers	Short	36.73%	44.44%
	Medium	25.00%	0%
	Long	0%	0%
Symbols	Short	7.69%	33.33%
	Medium	2.94%	0%
	Long	0%	0%

Table 5.14: Text detection and recognition for white lateral panels.

Data	Distance	Detection rate	Recognition rate
Words	Short	89.29%	88.00%
	Medium	72.83%	82.09%
	Long	15.49%	63.64%
Numbers	Short	82.68%	58.10%
	Medium	60.00%	42.22%
	Long	5.77%	0%
Symbols	Short	19.30%	90.91%
	Medium	37.21%	100%
	Long	6.67%	50.00%

Table 5.15: Text detection and recognition for white upper panels.

Data	Distance	Detection rate	Recognition rate
Words	Short	85.48%	79.25%
	Medium	64.37%	46.43%
	Long	20.22%	0%
Numbers	Short	62.07%	48.61%
	Medium	26.44%	10.87%
	Long	5.81%	0%
Symbols	Short	43.33%	71.79%
	Medium	36.36%	60.71%
	Long	18.29%	60.00%

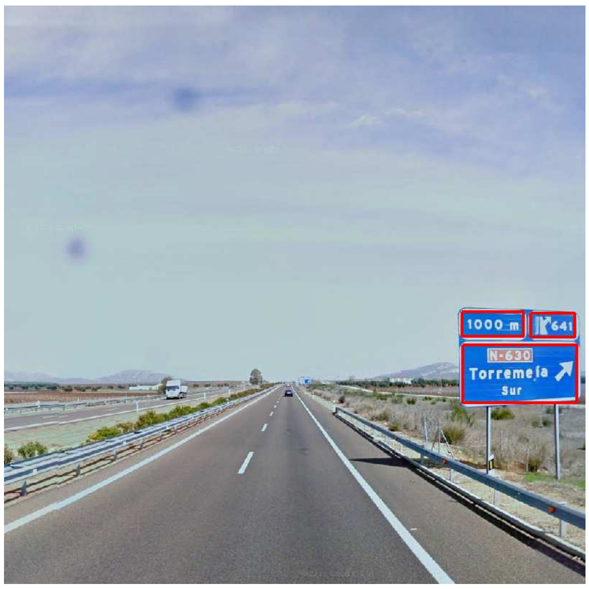
Table 5.16: Text detection and recognition including all lateral panels.

Data	Distance	Detection rate	Recognition rate
Words	Short	92.92%	79.19%
	Medium	85.33%	70.06%
	Long	46.32%	26.98%
Numbers	Short	83.44%	61.42%
	Medium	46.62%	48.39%
	Long	18.63%	10.53%
Symbols	Short	46.27%	90.32%
	Medium	66.09%	97.37%
	Long	31.25%	85.00%

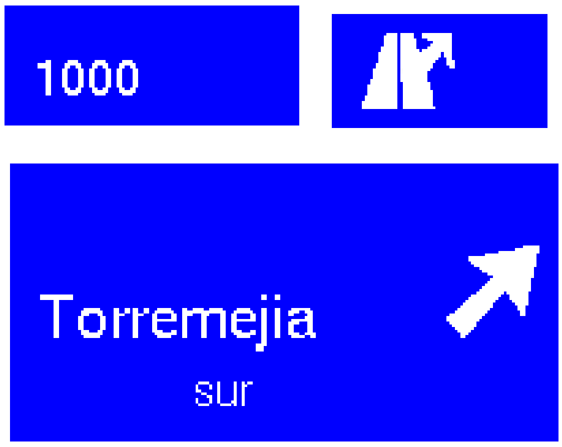
Table 5.17: Text detection and recognition including all upper panels.

Data	Distance	Detection rate	Recognition rate
Words	Short	90.18%	79.21%
	Medium	78.60%	63.85%
	Long	36.00%	20.99%
Numbers	Short	74.46%	56.93%
	Medium	38.64%	38.24%
	Long	13.09%	8.51%
Symbols	Short	45.09%	83.17%
	Medium	54.17%	87.50%
	Long	23.97%	74.29%

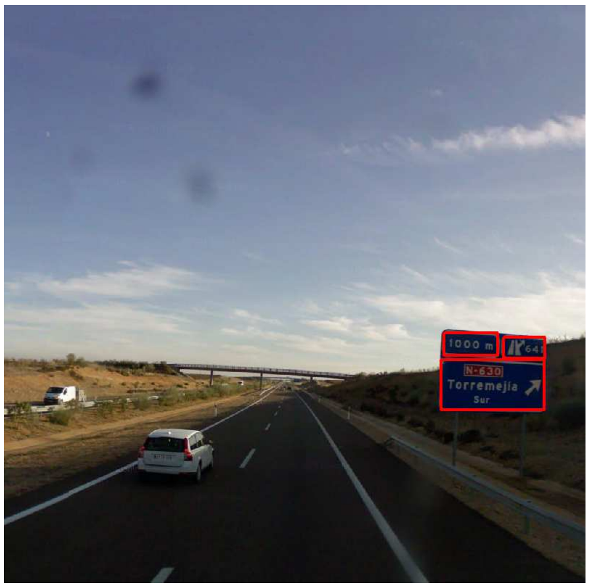
Table 5.18: Text detection and recognition including all the panels.



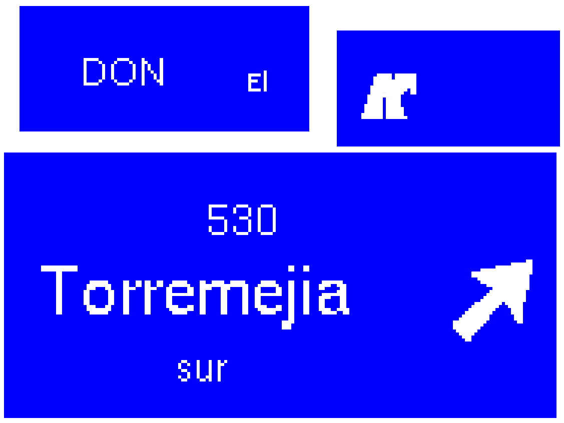
(a) Panel detection



(b) Text recognition

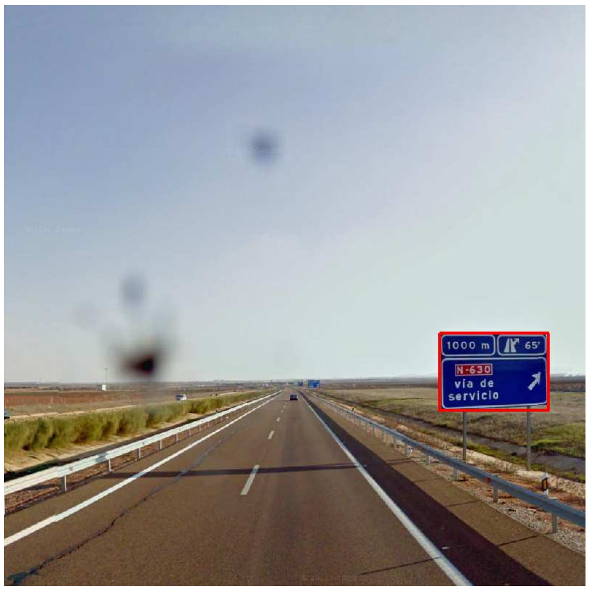


(c) Panel detection

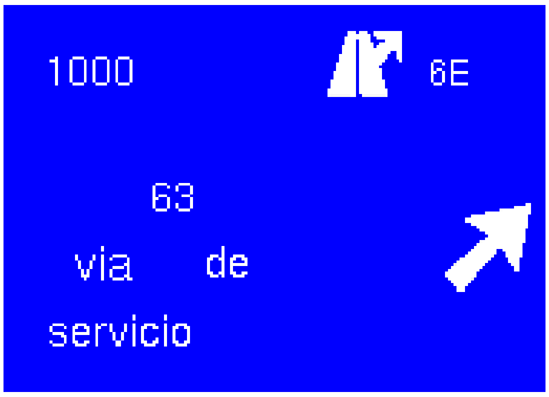


(d) Text recognition

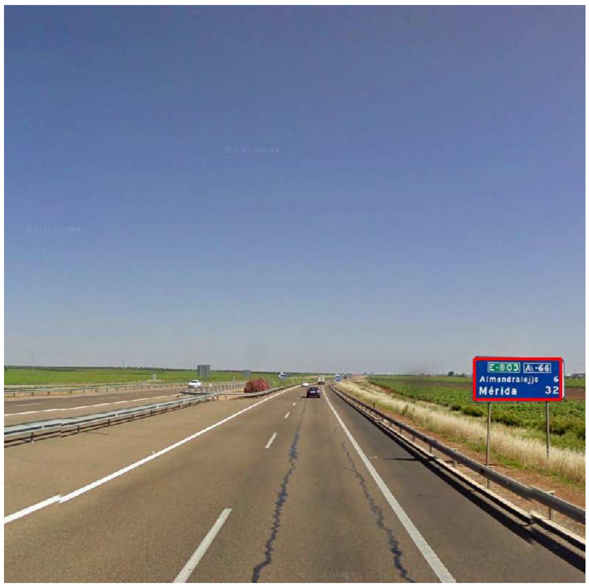
Figure 5.23: Image results



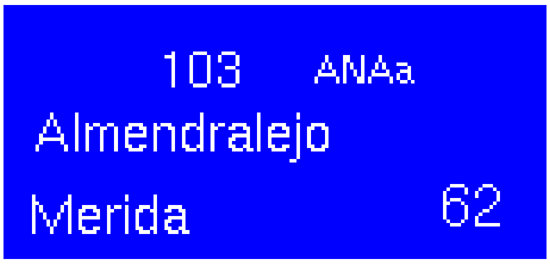
(a) Panel detection



(b) Text recognition

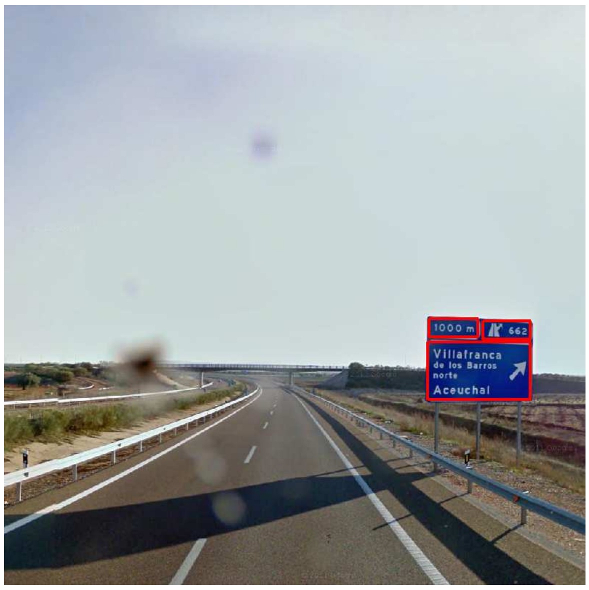


(c) Panel detection

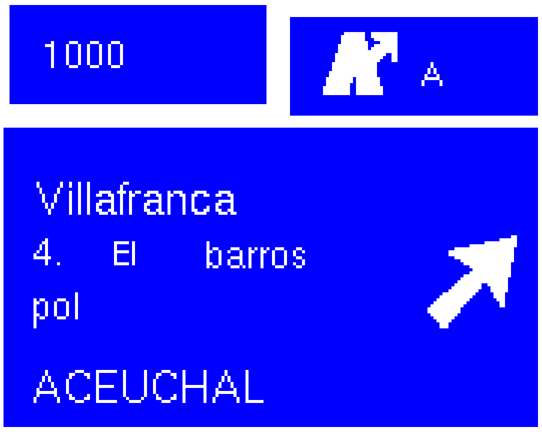


(d) Text recognition

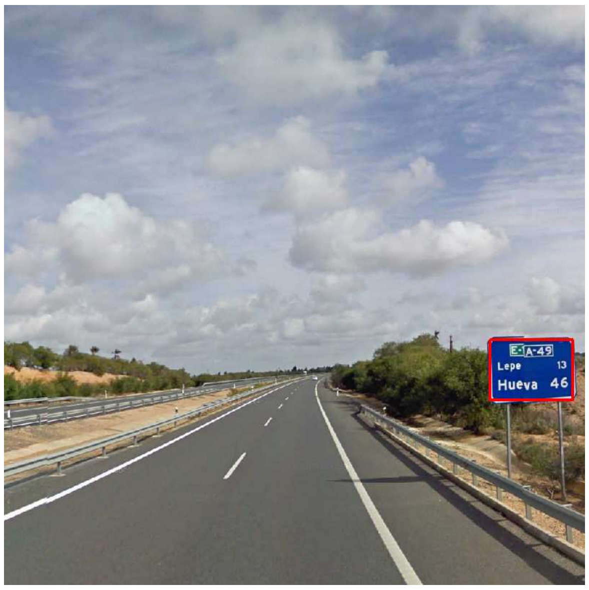
Figure 5.24: Image results



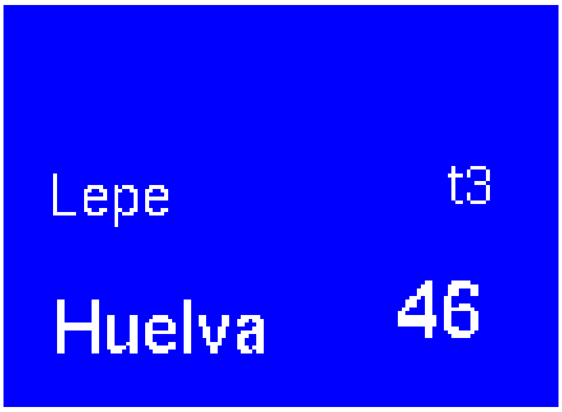
(a) Panel detection



(b) Text recognition

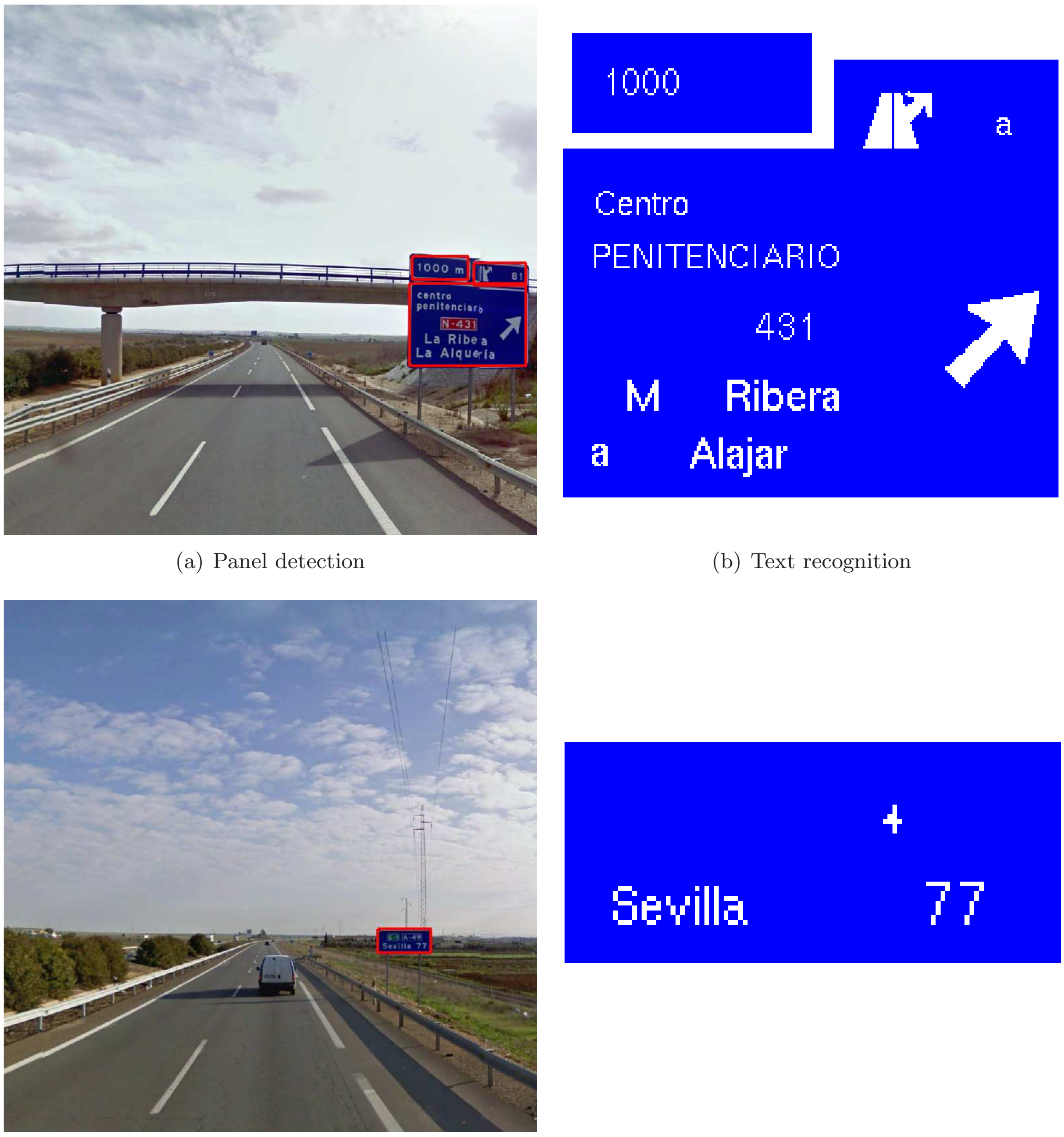


(c) Panel detection



(d) Text recognition

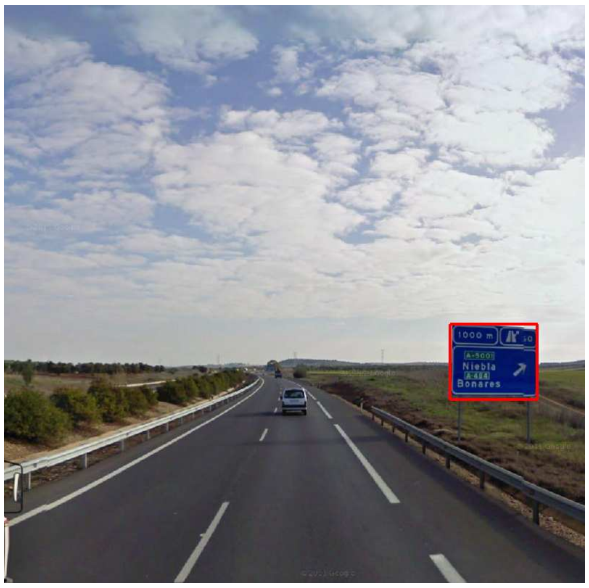
Figure 5.25: Image results



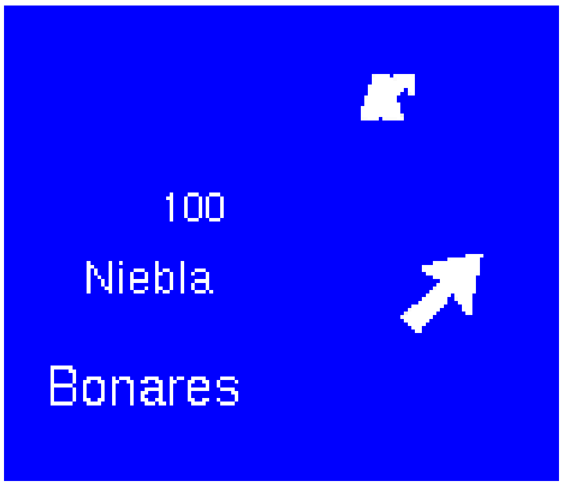
(a) Panel detection (b) Text recognition

(c) Panel detection (d) Text recognition

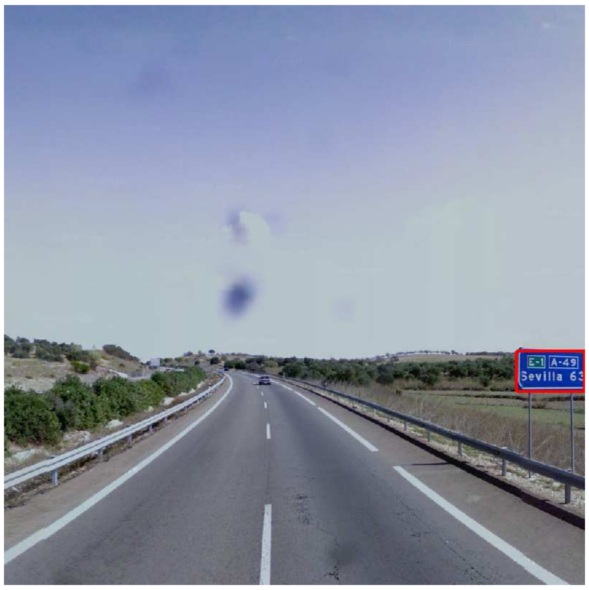
Figure 5.26: Image results



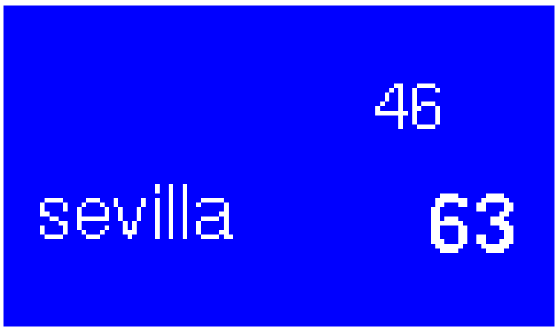
(a) Panel detection



(b) Text recognition

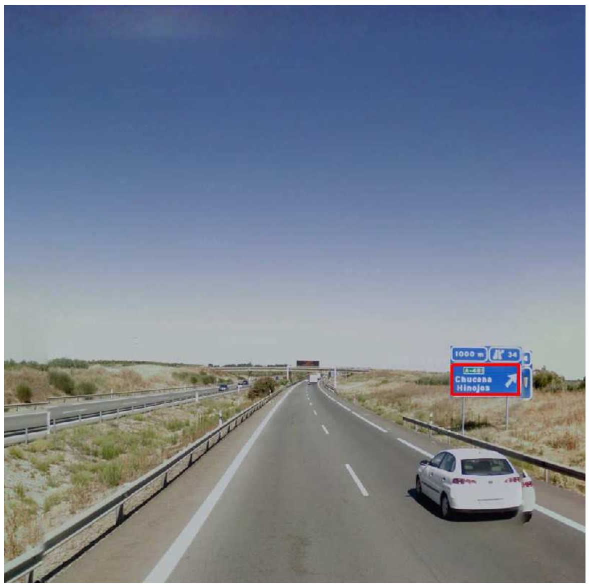


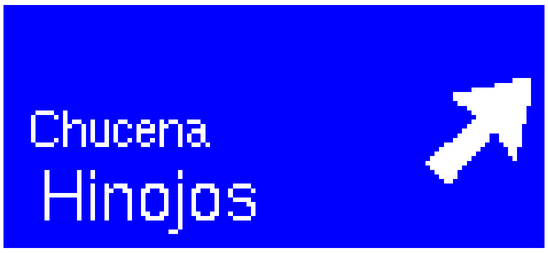
(c) Panel detection



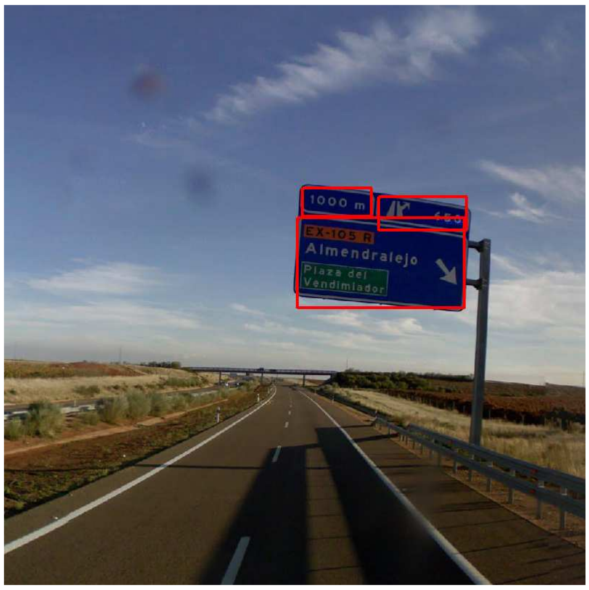
(d) Text recognition

Figure 5.27: Image results

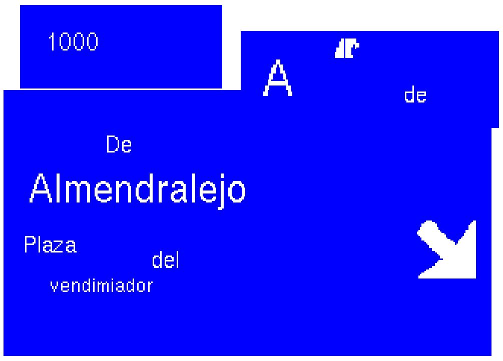

(a) Panel detection



(b) Text recognition

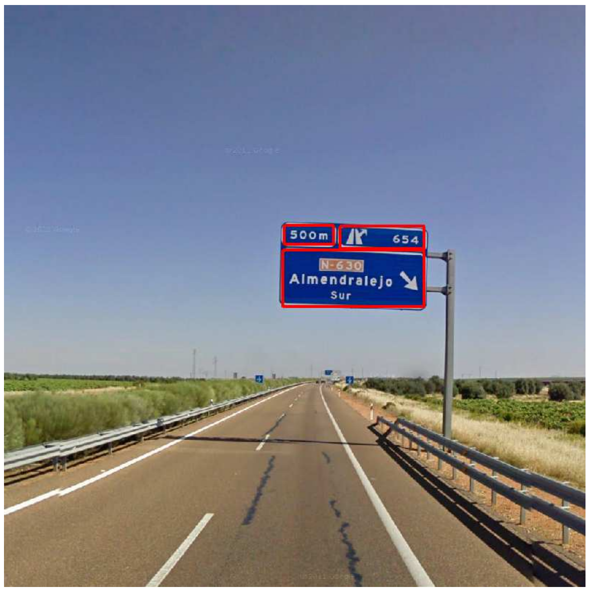


(c) Panel detection

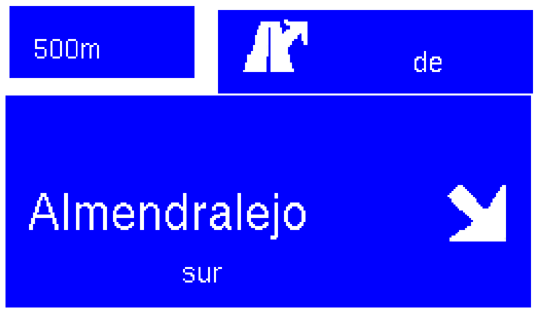


(d) Text recognition

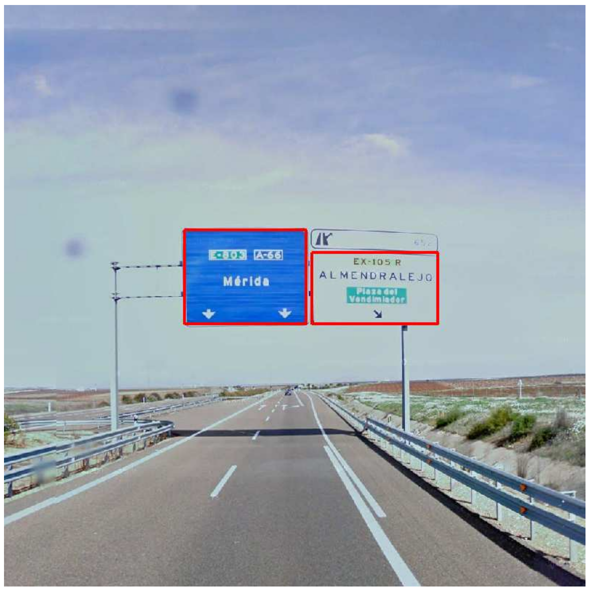
Figure 5.28: Image results



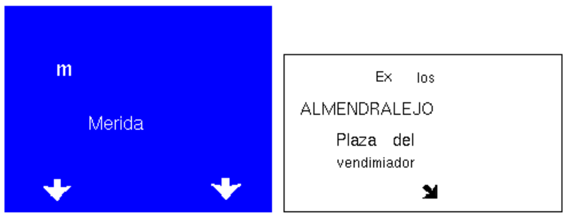
(a) Panel detection



(b) Text recognition

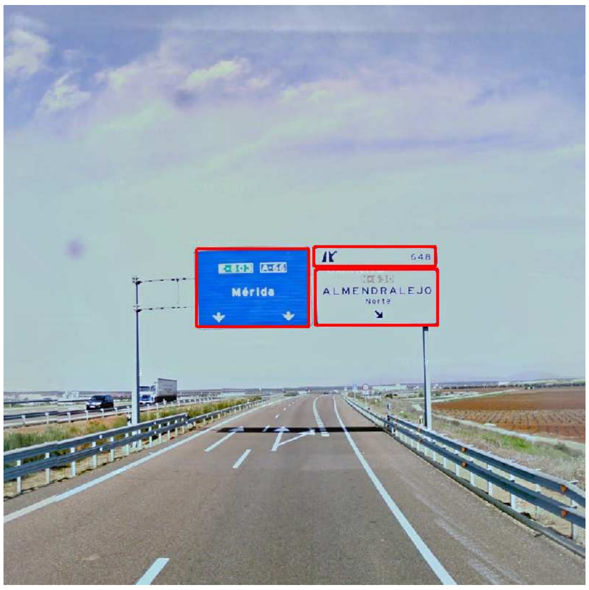


(c) Panel detection

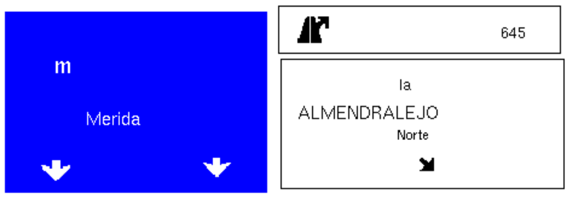


(d) Text recognition

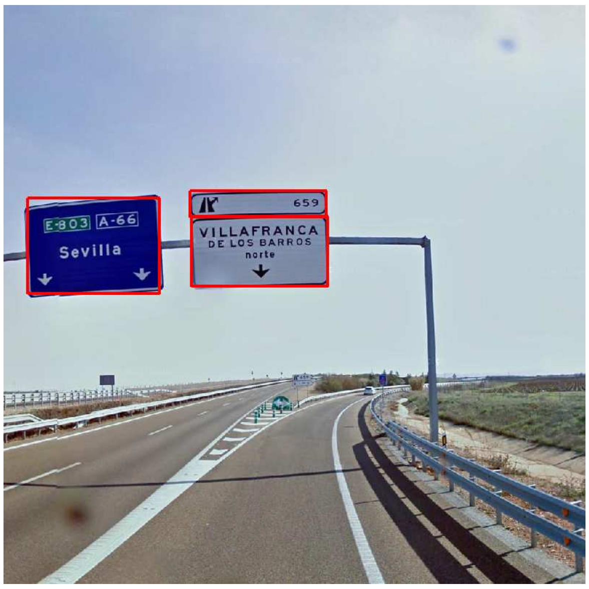
Figure 5.29: Image results



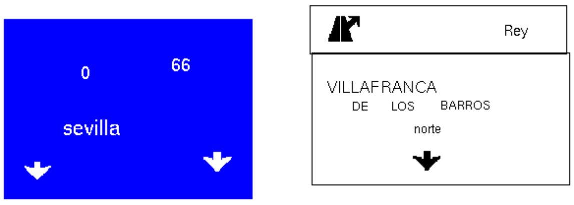
(a) Panel detection



(b) Text recognition

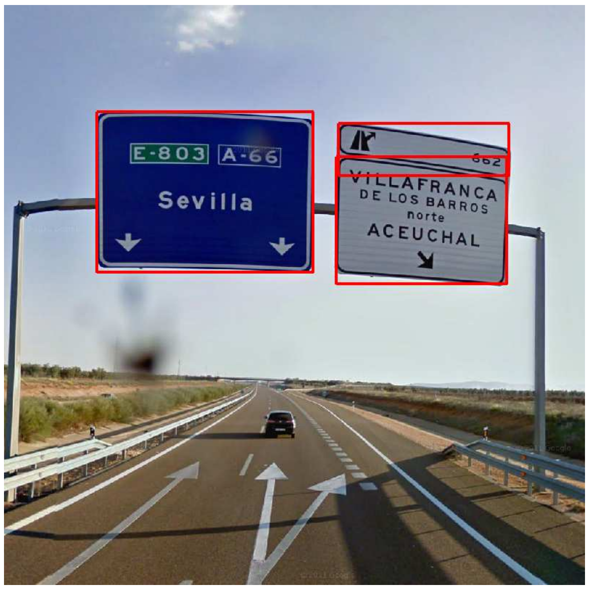


(c) Panel detection

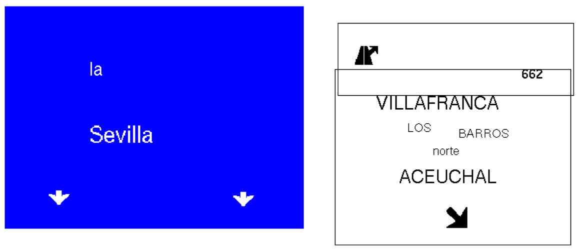


(d) Text recognition

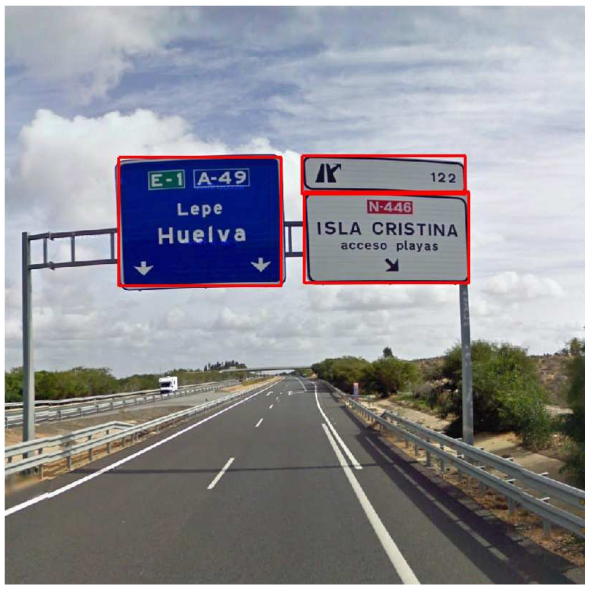
Figure 5.30: Image results



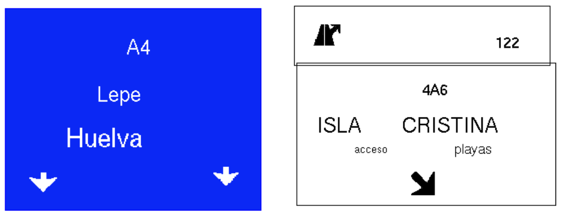
(a) Panel detection



(b) Text recognition

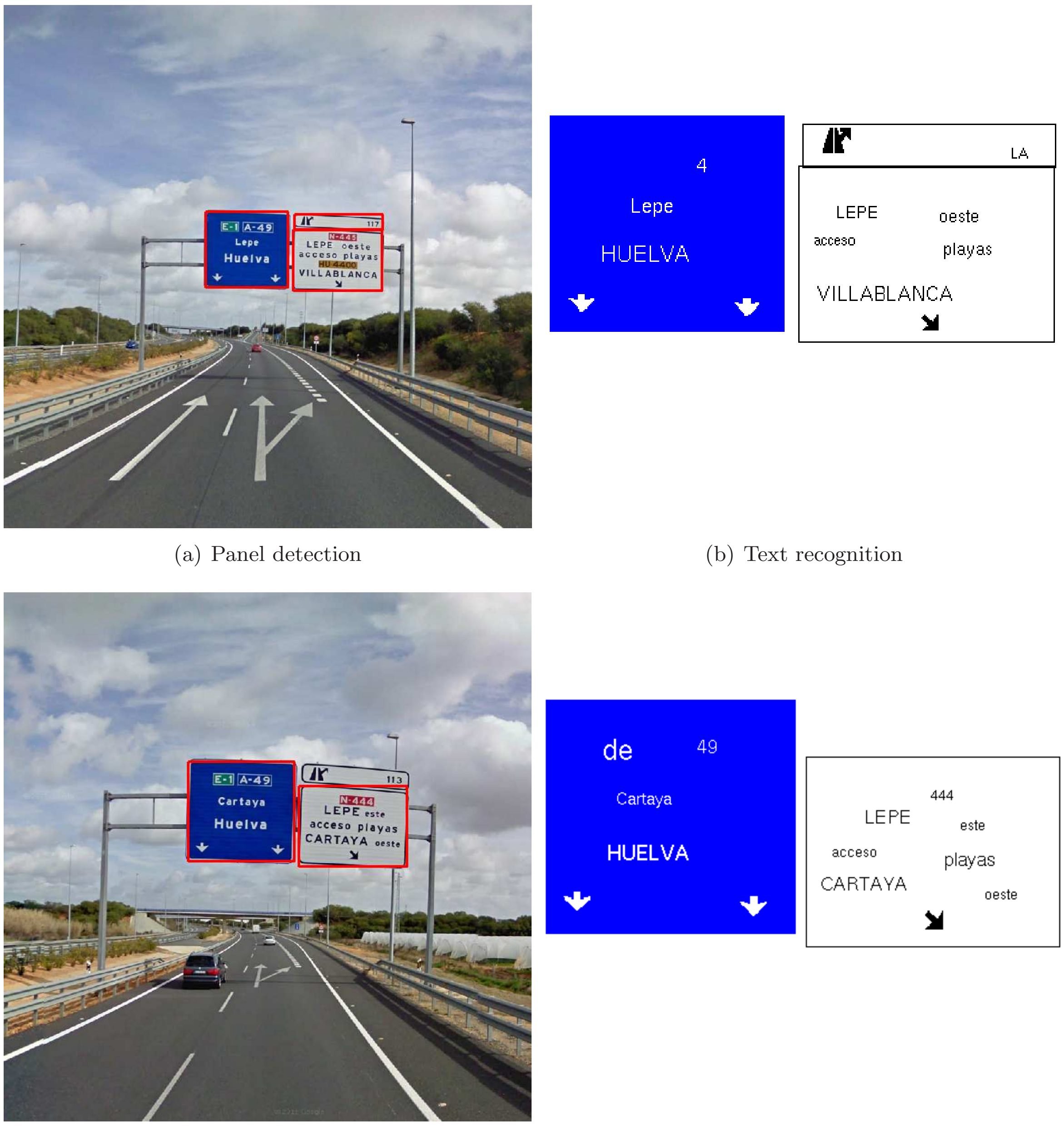


(c) Panel detection



(d) Text recognition

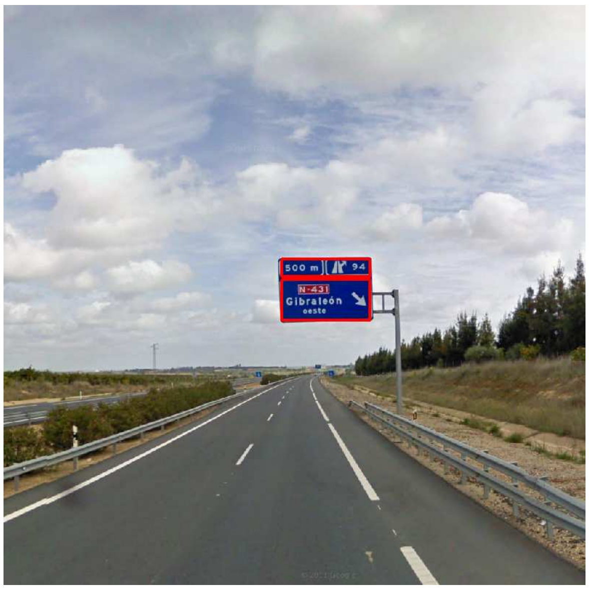
Figure 5.31: Image results



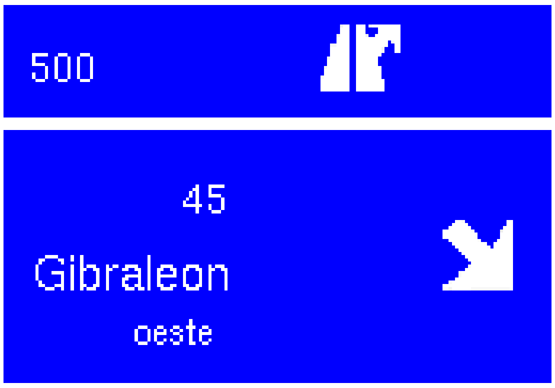
(a) Panel detection (b) Text recognition

(c) Panel detection (d) Text recognition

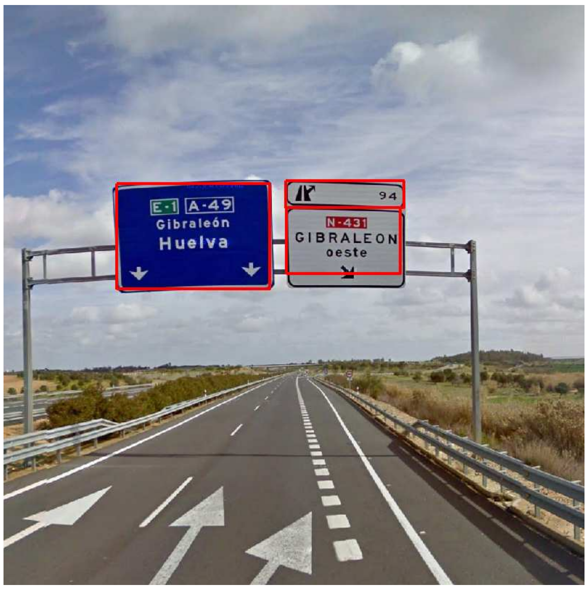
Figure 5.32: Image results



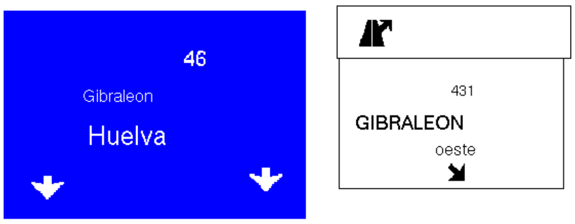
(a) Panel detection



(b) Text recognition

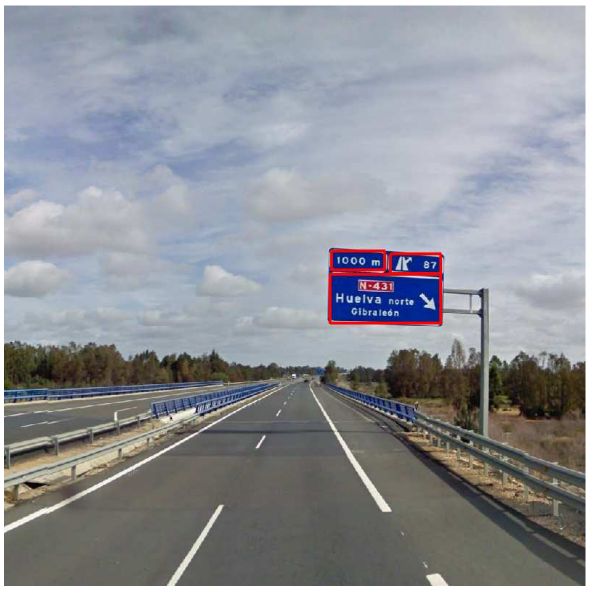


(c) Panel detection

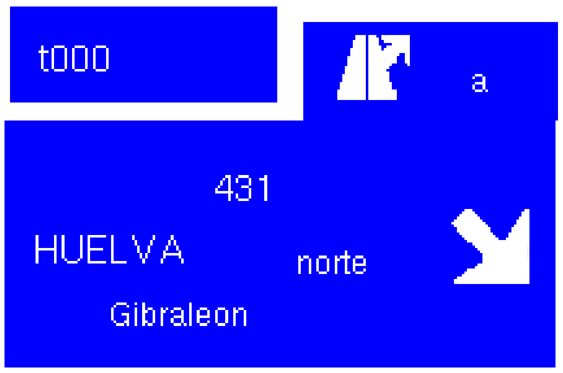


(d) Text recognition

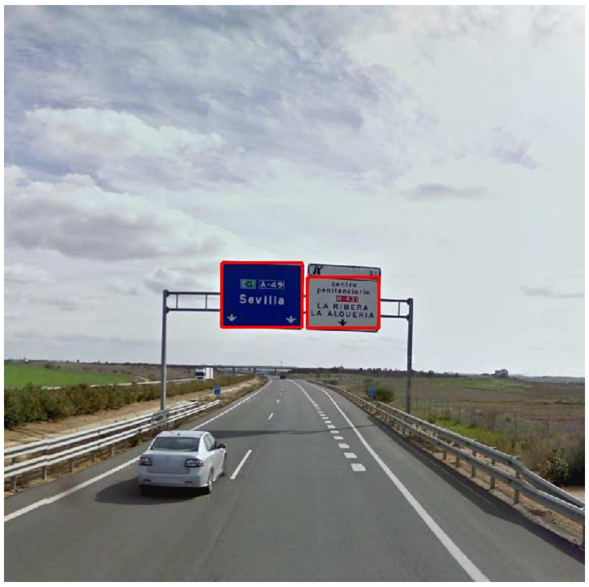
Figure 5.33: Image results



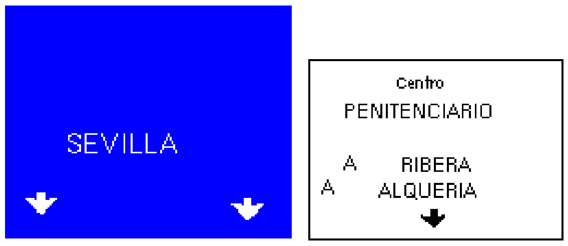
(a) Panel detection



(b) Text recognition

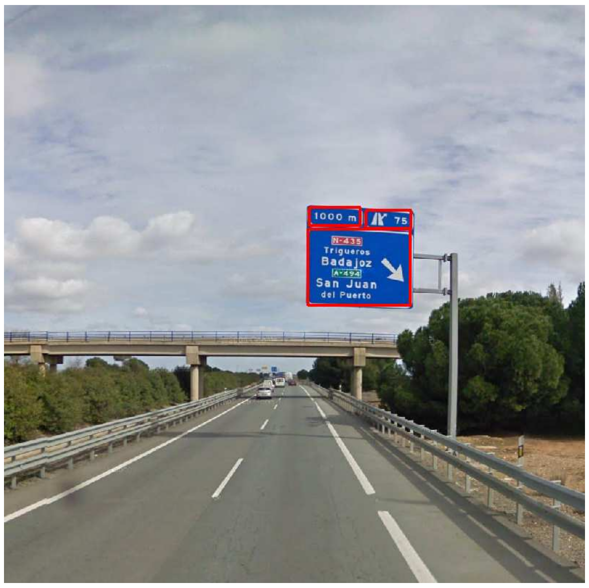


(c) Panel detection

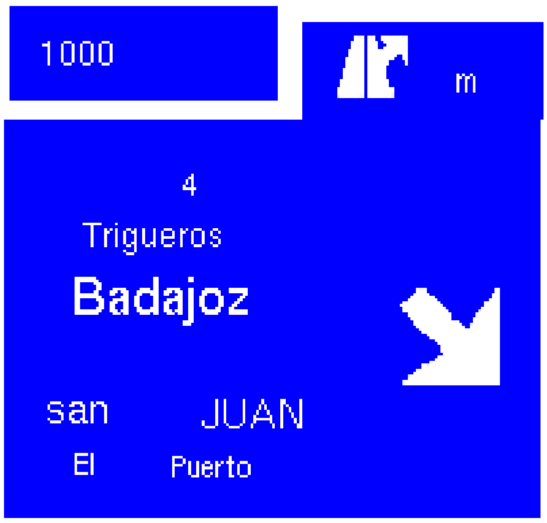


(d) Text recognition

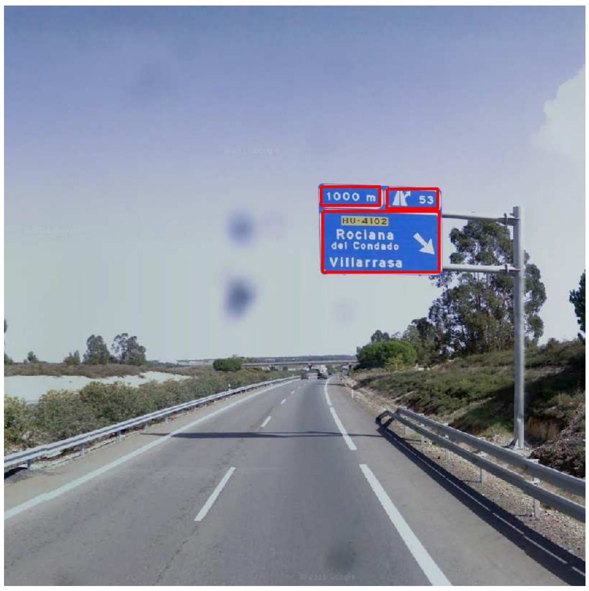
Figure 5.34: Image results



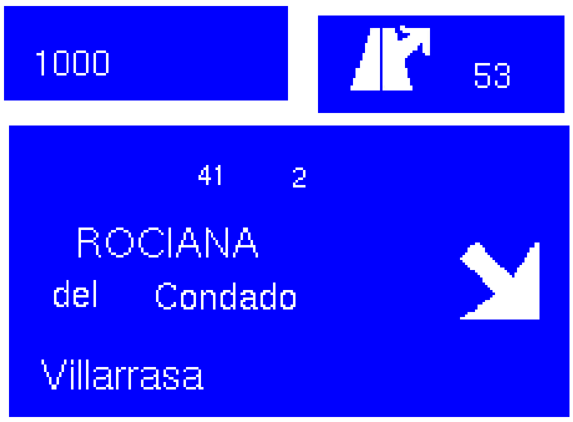
(a) Panel detection



(b) Text recognition

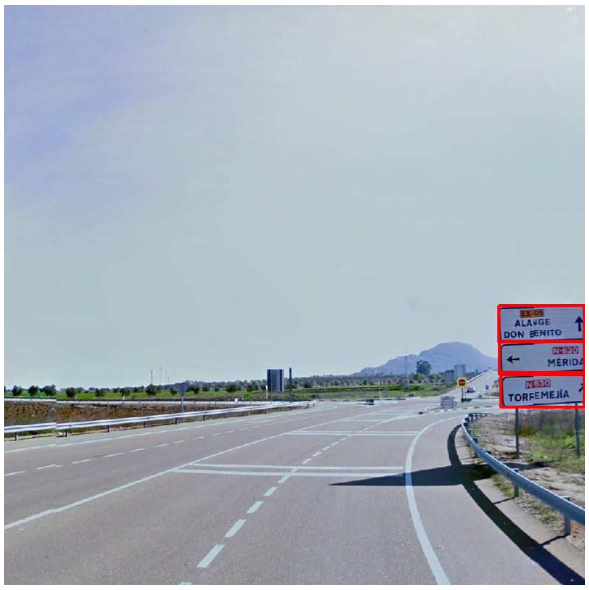


(c) Panel detection

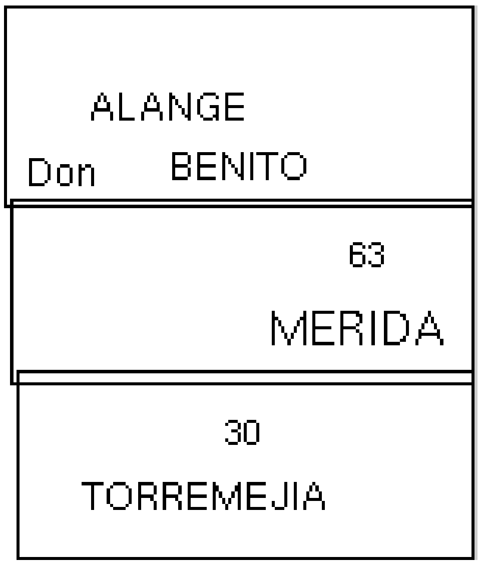


(d) Text recognition

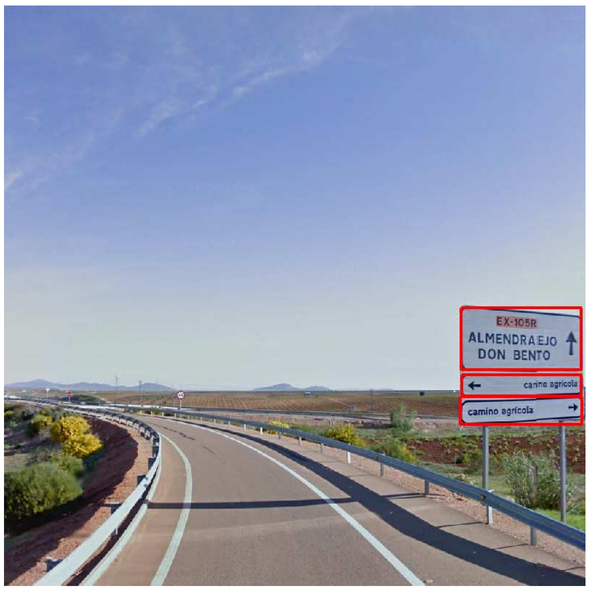
Figure 5.35: Image results



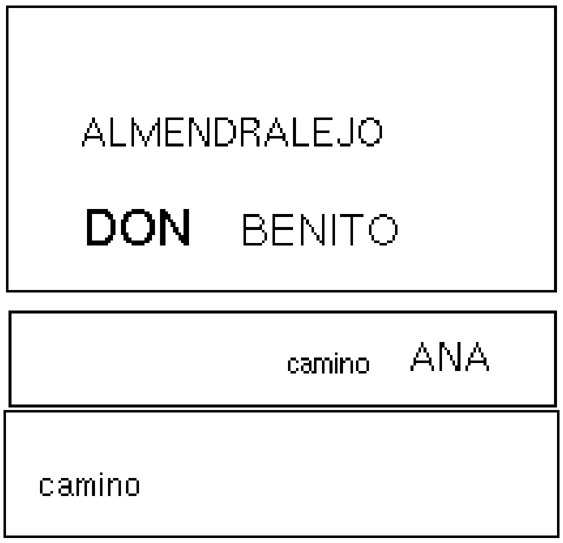
(a) Panel detection



(b) Text recognition

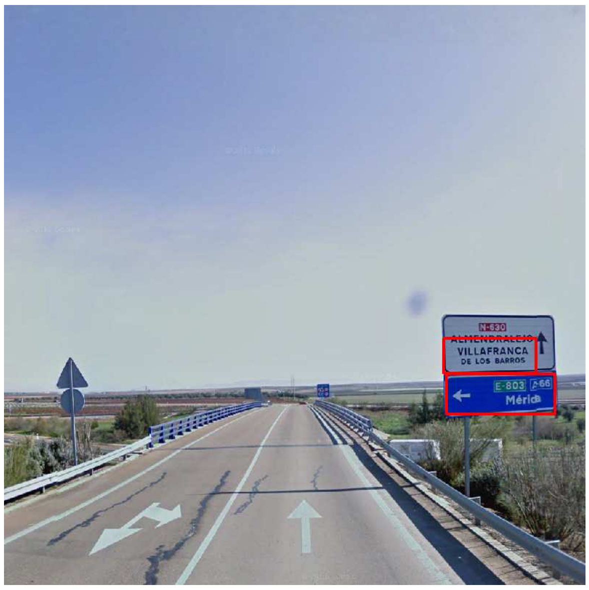


(c) Panel detection

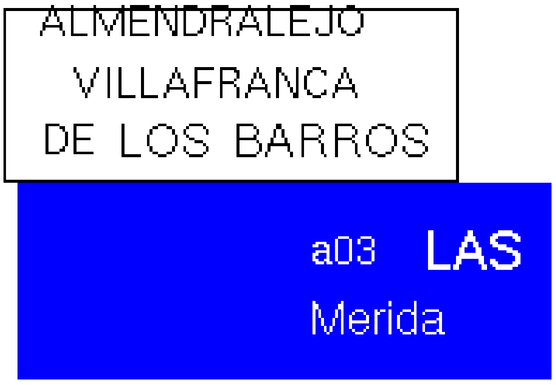


(d) Text recognition

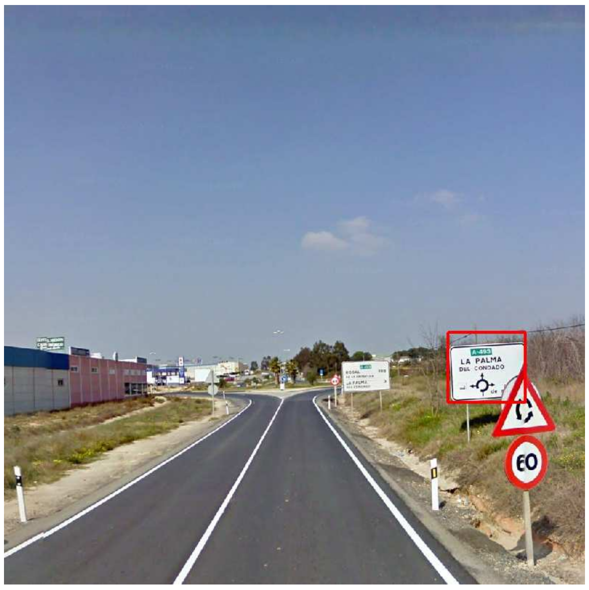
Figure 5.36: Image results



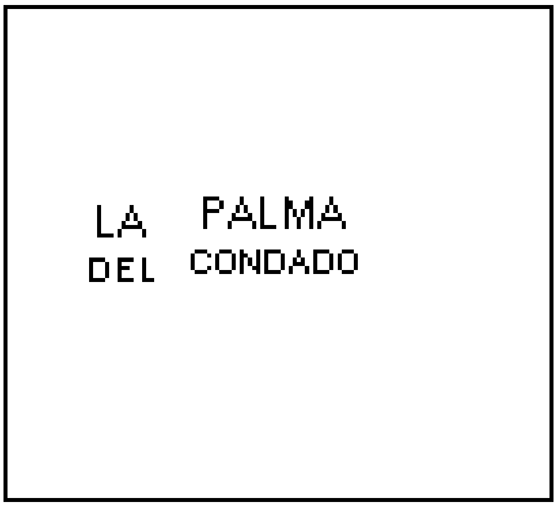
(a) Panel detection



(b) Text recognition

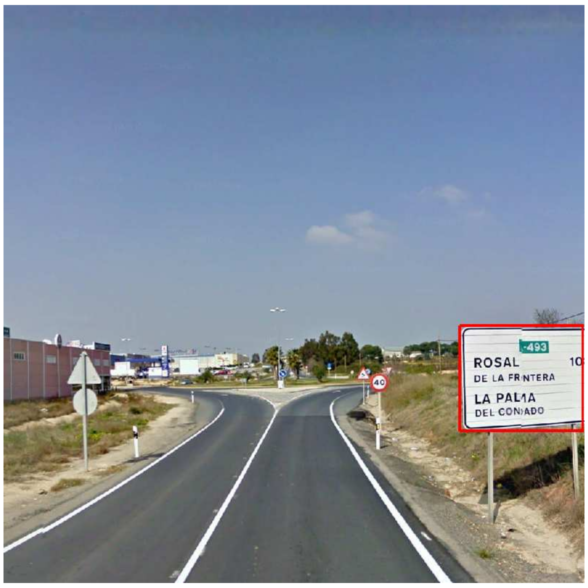


(c) Panel detection

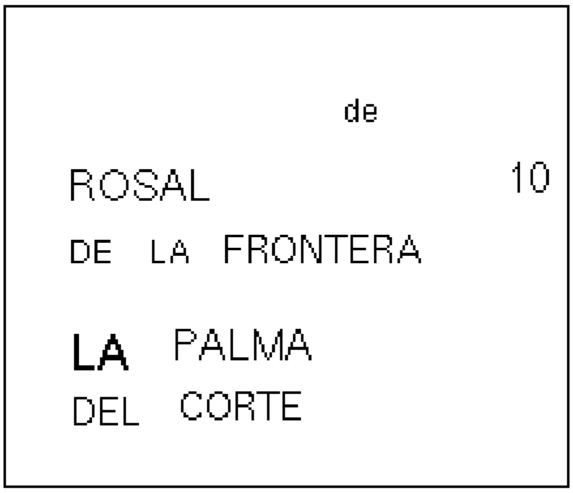


(d) Text recognition

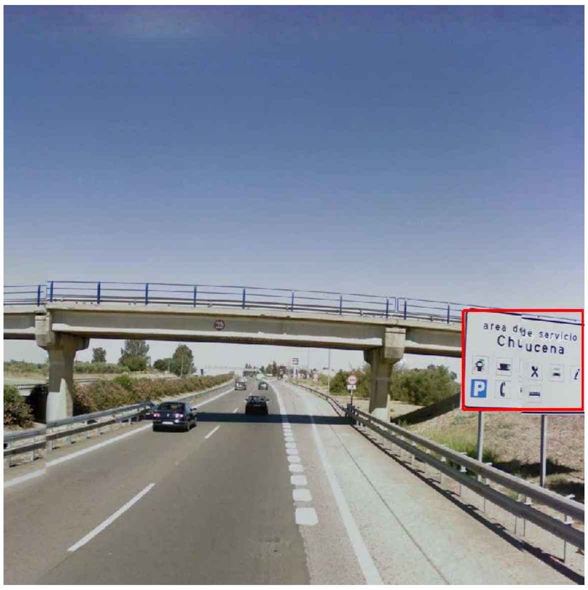
Figure 5.37: Image results



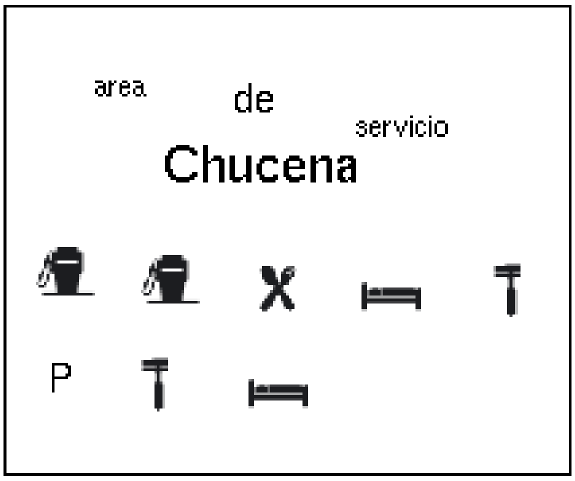
(a) Panel detection



(b) Text recognition

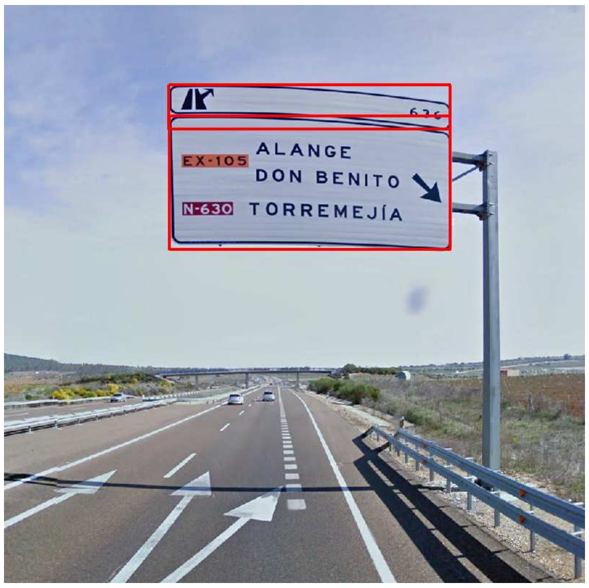


(c) Panel detection

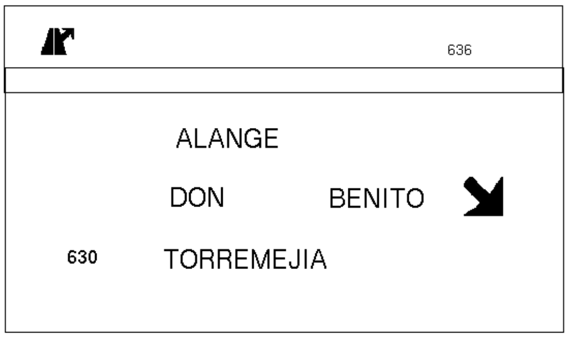


(d) Text recognition

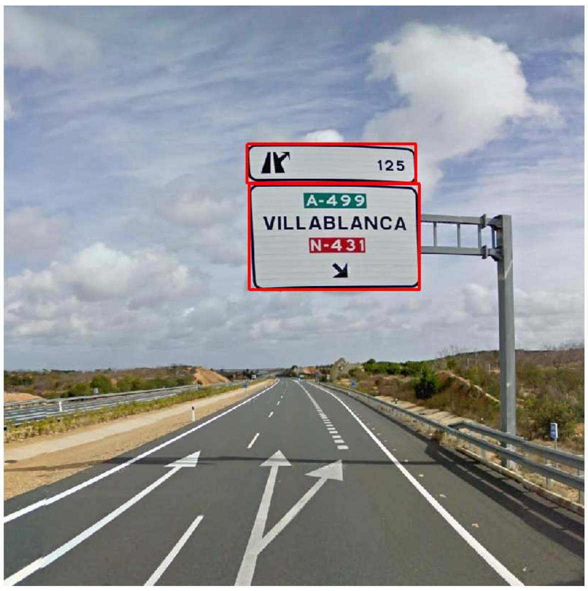
Figure 5.38: Image results



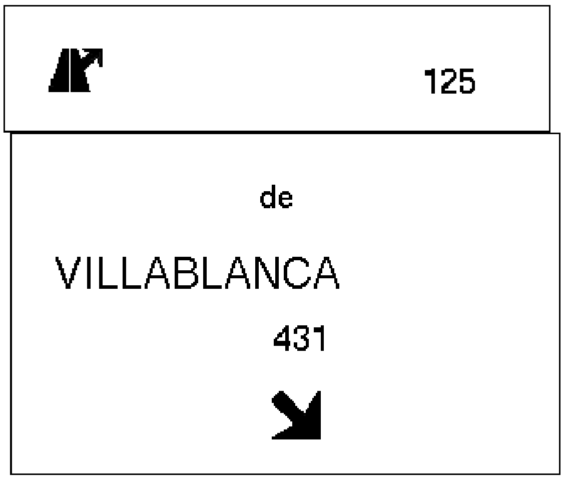
(a) Panel detection



(b) Text recognition

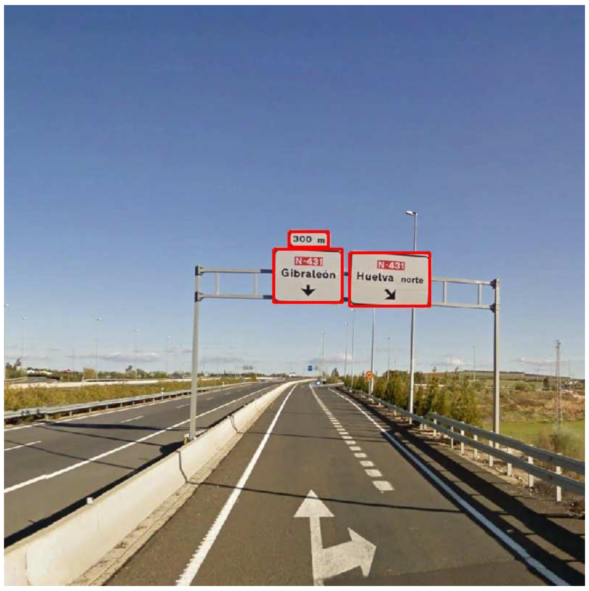


(c) Panel detection

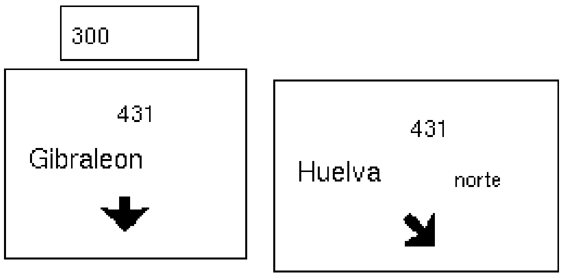


(d) Text recognition

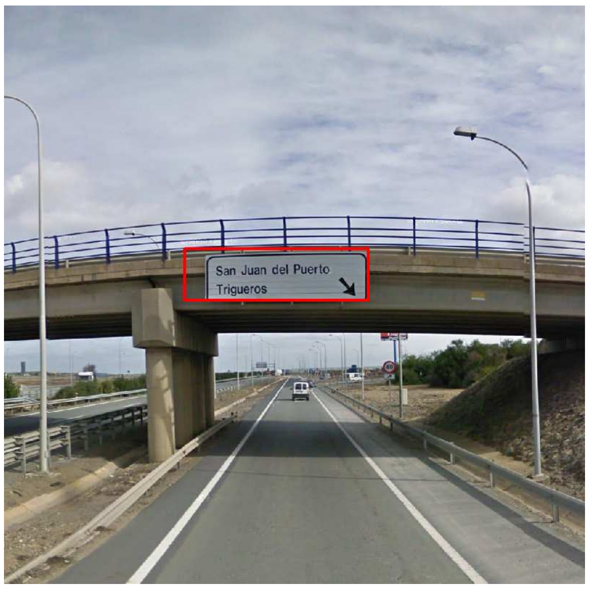
Figure 5.39: Image results



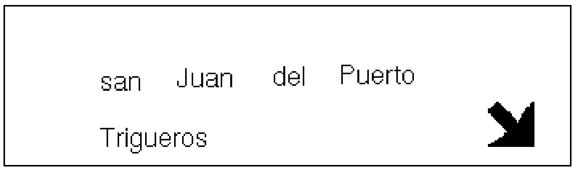
(a) Panel detection



(b) Text recognition



(c) Panel detection



(d) Text recognition

Figure 5.40: Image results

5.7 Conclusions and future work

In this chapter we have presented a real application of the text detection and recognition algorithm proposed in the previous chapters. It consists of reading the information depicted in traffic panels using panoramic images downloaded from the Google Street View service. The main use of this application is to automatically create up-to-date inventories of traffic panels of whole regions or countries. This information is very useful for supporting road maintenance and for developing future driver assistance systems.

The main contribution of this chapter is the modelling of traffic panels using a BOVW technique from local descriptors extracted at interest keypoints, instead of using other features such as edges or geometrical characteristics as it has been done up to now in the literature. This is not an easy task due to the immense variability of the traffic panels. However, the experimental results show the effectiveness of the proposed method. Different gray-based and color-based descriptors have been compared and color descriptors, specifically Hue Histogram and Transformed Color Histogram, have proved to be more adequate to be used in this application. In addition, the dimensionality of these two descriptors is small (only 37 and 45 elements, respectively), thus the training time is lower than using other descriptors of higher dimensions.

On the other hand, the BOVW technique used is able to detect the presence or the absence of a traffic panel in the image, but it is not able to detect the exact position of the panel in the image. We have used color masks to restrict the possible areas in the image where the traffic panels can be located. However, as future work we intend to use spatial extensions to BOVW so it can also detect the precise position of the panels in the image.

The text location and recognition method described in the previous chapters is applied only on those images where a panel is found, in order to reduce the number of false positives and increase the efficiency of the proposed algorithm. The word recognition is based on a unigram language model, which is partly based on a fixed dictionary that contains common words that can be found everywhere, and partly based on a dynamic dictionary that depends on the province where the traffic panel is located. The model assumes that all the words have equal prior probability. As future work, we intend to compute the prior probabilities of all the words in the dictionary, so the word recognition is more precise and reliable. In the same way, the use of an unigram language model does not take into account the likelihood of appearing two or more words together. Using language models of a higher order would allow to recognize more precisely the names of places composed of several words.

Finally, the recognition of the information depicted in the traffic panels is done frame by frame. Typically, a panel appears in several consecutive frames. As future work, we intend to do a multi-frame integration of the recognized information at each single frame. In addition, the use of the a priori knowledge that we know about the design of traffic panels would improve the recognition rates, because certain objects, especially symbols and numbers, are located only at certain parts of the panels.

Chapter 6

Conclusions and future work

6.1 Conclusions

The starting point of this thesis is the development of a robust algorithm to locate and recognize text in camera-captured real-world scenes that can be successfully applied to different kinds of applications such as: help to visually impaired people, support to robotic navigation, image spam filtering, translation services for tourists or driver assistance, among many others.

We have made a review of the main and most remarkable works of the last years in the field of text reading in natural scene images and we have developed a method to robustly detect the presence of text in real-world images, locate it and recognize it. We have carried out a study of the most suitable features that describe text versus non-text, using some features proposed by other authors and suggesting a new set of features. It has been shown in this thesis that the probability density functions of the proposed features can be fitted to a unimodal Gaussian distribution for text objects, while they do not follow any known unimodal distribution for non-text components.

The text location method proposed here is based on CC analysis. We have proved that MSER is one of the most suitable region detectors for text reading systems, but it is a global method. We have improved its performance by complementing it with a local adaptive thresholding algorithm that enables our text location approach to segment text even when blur motion is present in the images or if the image resolution is small. The suggested location method is able to detect both light and dark text, even when both situations are present in the same image, improving state-of-the-art proposals. For CC filtering and text line aggregation, two sets of geometric constraints have been proposed and the respective thresholds have been optimized using genetic algorithms. Unlike other methods, we have proposed a recursive method to restore character CCs that may have been initially erroneously discarded, improving the accuracy of the detection. In order to deal with certain structures that can be typically confused with text, especially repetitive structures such as windows or fences, a text line classifier based on gradient and grey-level features has been proved to be robust enough.

On the other hand, the text recognition method is based on identifying single characters and then applying a language model based on probabilistic inference to correct misspelled words, as the outputs are constrained to a dictionary of all the possible terms. The dictionary is the British National Corpus, which is a compendium of the vast majority of all the modern British English words. Character recognition is carried out using a new feature proposed in this thesis, Direction Histogram (DH), which is a simplification of HOG, as the gradient directions are computed only at the contour pixels of the characters.

DH has proved to achieve better results compared to HOG and other state-of-the-art features. The classification is based on KNN, although instead of giving only a binary output, we propose to give a fuzzy output with membership probabilities for each character to be recognized. The probabilities are computed from the distance ratios of each of the K nearest neighbors. These probabilities, together with other evidences, have proved to be useful to separate characters that may have been wrongly connected in the segmentation process.

Experimental results obtained with different datasets show the robustness of the proposed method. In terms of text location, four datasets have been tested and we have proved that we outperform state-of-the-art performance in two of them, achieving a performance of almost 80% for web images and 45% for CD cover images, while we are slightly lower than other works in the other two datasets, but our method obtains the highest value of precision (80% for natural scene images and 90% for web images), meaning that our algorithm is the one that gets the lowest amount of false text detected. On the other hand, in terms of word recognition, we improve state-of-the-art performance in all cases, achieving a recognition rate of 47% for natural images and almost 70% for web images. In addition, we have shown that the use of a language model multiplies by three the word recognition rate obtained if a language model were not used.

Finally, a real application to ITS has been proposed in this thesis with the aim to be used as an automatic system to create up-to-date inventories of traffic panels to support road maintenance or to assist drivers. From our knowledge, there is not any available dataset of images of traffic panels. Therefore, a new dataset composed of panoramic views downloaded from the Google Street View service has been compiled. The dataset comprises more than 10,000 images with positive and negative samples for training and testing, which contain around 700 different traffic panels at different lighting conditions, with different sizes and colors. We have proposed to detect the presence of traffic panels in the images using a BOVW technique and local descriptors extracted at interest keypoints. Actually, several descriptors in the literature have been compared and the best results have been obtained with low-dimensional color descriptors such as Hue Histogram and Transformed Color Histogram. Panel detection rate is higher than 96%. Blue and white color masks have been proposed as a way of restrict the possible areas in the image where the traffic panels may appear. The text detection and recognition is carried out using the algorithm presented in this thesis. However, slight modifications have been introduced for this specific application. The most important one refers to the use of a dynamic reduced dictionary that depends on the geographic position of the traffic panels, instead of using a fixed whole dictionary like the British National Corpus, which is used in the general case of recognizing English text in any situation. We achieve a detection rate bigger than 90% for words, 75% for numbers and almost 50% for symbols, being the recognition rates around 80% for words and symbols and almost 60% for numbers.

6.2 Main contributions

From the results obtained in previous chapters, we consider that the main contributions of this thesis are the following:

- **Text features analysis.** In this thesis we have studied different set of features to find which are the most adequate to describe text versus non-text regions. We have used features proposed by other authors and we have suggested new features. We have shown that the probability density functions of the proposed features follow an unimodal Gaussian distribution for text characters, while they do not fit any known unimodal distribution for non-text objects. This conclusion has been obtained using only one dataset that contains text and non-text regions in many different situations and conditions, including different fonts, sizes, layouts, colors and lighting conditions, but from the good experimental results obtained with other datasets, we can conclude that the results from this study can be generalized to any dataset and to any situation, thus neither re-training nor re-tuning of the parameters of the algorithm are necessary.

- **Image segmentation.** Many authors have shown that MSER is one of the most suitable region detectors for text reading systems, but it is a global method that may fail when the illumination in the images is not homogeneous. In this thesis, we have improved its performance complementing it with a local adaptive thresholding method that enables our text location approach to segment text even when blur motion is present in the images or if the image resolution is small. The suggested location method is able to detect both light and dark text, even when both situations are present in the same image.

- **Text line classification.** The text location approach proposed in this thesis is based on connected component analysis. However, it has been combined with the idea used on texture-based methods consisting of extracting certain features on large areas of the image and applying machine learning techniques to classify these areas in order to validate the detected text line candidates. We have shown the improvement that this technique produces in the performance of the algorithm. Different features have been compared, showing that the combination of gradient features (HOG) with simple features such as mean (MDF) and standard deviation (SD) is the best option.

- **Text detection results.** As mentioned before, the proposed framework has been trained using only one dataset, but it has been tested on four very different datasets, achieving or even improving in some cases state-of-the-art performance. Actually, in terms of precision, our method is the one that gets the highest precision (80% for natural scene images and 90% for web images), that is, the lowest number of false positives.

- **Character recognition.** We have proposed a new feature, Direction Histogram (DH), to characterize single letters. This new feature can be seen as a simplification of HOG, as the gradient directions are computed only at the contour pixels of the characters. Actually, we have shown that the proposed feature performs better than HOG for the proposed framework. Moreover, the character recognition is carried out with a classification method based on KNN that gives a fuzzy output instead of the classical binary output. The probability values given by the fuzzy classifier have been proved to be a reliable indicator of situations like having characters wrongly connected during the segmentation process.

- **Word recognition.** We have proposed to use a language model based on probabilistic inference to constrain the output of the character recognizer to a set of meaningful words, in order to correct misspelled words. The model is based on the British National Corpus, which is a dictionary of modern British English words, but any other dictionary can be easily adapted to the proposed method depending on the application. For instance, for the application proposed in this thesis to recognize traffic panels, a dictionary that contains names of cities, places and other words has been easily created and used. The language model finds the most likely word that may have generated an observed sequence. This is done using the forward algorithm utilized in HMMs, but modeling the substitutions using probabilites instead of costs. Add-1 smoothing is applied in order to avoid the assignment of zero probabilites. On the other hand, we have proposed to deal with insertions and deletions in the observed sequences using a "split and merging" process over the segmented image. It has been shown the importance of using a language model, as it multiplies by three the word recognition rate compared to the case in which a language model is not used.

- **Text recognition results.** The proposed system has been tested using three different challenging datasets, the most commonly used in the field of text recognition in natural images, and the experimental results show that we outperform other state-of-the-art works, achieving a recognition rate of 47% for natural images and almost 70% for web images.

- **Application to recognize traffic panels.** A real application consisting of detecting and recognizing the information contained in traffic panels has been developed. A dataset composed of thousands of images downloaded from the Google Street View service has been created. We propose to model traffic panels using a BOVW technique from local descriptors extracted at interest keypoints, instead of using other features such as edges or geometrical characteristics as it has been done up to now in the literature. The keypoints are only extracted on some regions given by two blue and white color masks. For the white color mask we use MSER, but for the blue color mask, we propose to combine the advantages of three different methods, two of them previously proposed by other authors but the third one is proposed by us, using the Otsu's segmentation on the difference of two color channels. We also propose to recognize the words that appear in the panels using a dynamic language model, which is partly based on a fixed dictionary and partly based on a dynamic dictionary that depends on the geographical position.

6.3 Future work

From the results and conclusions of the present work, several lines of work can be proposed. For the text location and recognition method, the next aspects should be improved:

- **Text detection accuracy.** In terms of precision, the proposed method improves state-of-the-art performance in all the cases that have been study, which means that this method produces the lowest number of false positives among all the works reviewed. However, the recall, that is, the accuracy of the detection, is still quite low. As future work, we are interested in improving the recall value of the detection by keeping the precision high. The addition of new features should be studied for this purpose.

- **Character separation using fuzzy theory.** We have developed a character classification method that provides a fuzzy output for each object. We have shown that the probabilities given by the classifier can be used as a reliable evidence to detect if some characters may have been wrongly connected during the segmentation process. However, the proposed method has some heuristic characteristics. As future work, we intend to carry out the separation of characters using fuzzy inference.

- **Computational time.** We need to reduce the computational time of the proposed algorithm, so it can be embedded into a mobile device such as a smartphone or a Tablet PC. Our text detection and location proposal has been developed using Matlab, thus the code is not optimized. In addition, the processing time for the detection and location depends on the complexity of the image. It can be around 10 seconds for those images that have been taken in a very complex scenario and produce many connected components in the segmentation step. On the other hand, simple images may have a computational time of 1 or 2 seconds at maximum. In this sense, the bottlenecks of the method are, firstly, the segmentation step, which should be revised more in depth in order to reduce the number of character candidates that it generates, and secondly, the calculation of the Stroke Width Transform, which can take up to 50% of the total processing time of the detection and location algorithm in case that the image is too complex. Another aspect that we propose as future work in order to reduce the computational time of the method is the use of multithreading to locate dark-on-bright text and bright-on-dark text in parallel, because at the moment they are searched one after the other because only one thread is used. On the other hand, the character and word recognition has been developed in C++. The computational time for recognizing a single character is around 60 ms, while the time that takes to recognize a single word is around 500 ms.

- **Prior probabilities in the language model.** The language model used to recognize words in general cases using the British National Corpus takes into account the prior probability of each word in the dictionary. However, the language model used to recognize words in traffic panels assumes that all the words have equal prior probability, since we had not enough reliable data to compute the prior probabilities of all the words in the dictionary. As future work, we intend to collect enough reliable data to compute these values.

- **Use of language models of higher order.** The use of a unigram language model does not take into account the likelihood of appearing two or more words together. We intend to use language models of a higher order that allow to recognize more precisely sets of words that go along together and belong to the same context.

For the proposed application to traffic panels recognition, we suggest the following lines of work:

- **Spatial extensions to BOVW.** The BOVW technique used to detect the presence or absence of a traffic panel in an image is not able to extract the exact position of the panel in the image. We have proposed to use color masks to restrict the possible areas in the image where the traffic panels may appear, achieving very good results. However, as future work we intend to use spatial extensions to BOVW so it can also detect the precise position of the panels in the image without needing the use of color masks. In addition, we would like to explore the use of other distances like the Mahalanobis distance, instead of the Euclidean one, when identifying the nearest visual words of a test image respect to the trained system with the BOVW method.

- **Multi-frame integration.** The recognition of the information depicted in the traffic panels is done frame by frame independently. However, a panel typically appears in several consecutive frames. We intend to do a multi-frame integration of the recognized information at each single frame.

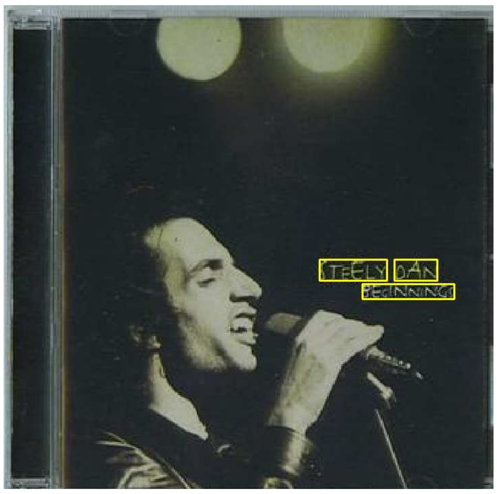
- **More accurate text geolocalization.** The proposed text geolocalization is based on a coarse computation of the coordinates of the panels respect to the coordinates of the vehicle that records the images. The precision of this method is not very good, as errors of more than 10 meters can occur. As future work, we intend to develop a more accurate geolocalization algorithm. One way of doing it is by using the camera parameters, but they are unknown at the moment.

Appendix A

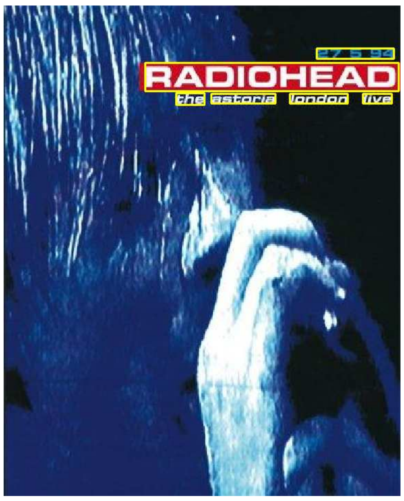
Results for the CoverDB dataset

This appendix shows some examples of the results obtained with the text location method proposed in this thesis using the CoverDB dataset, whose test set is composed of 300 low-resolution images of CD and DVD covers. Some images are shown in figures A.1-A.12. The result of each detection is painted using yellow, blue, cyan or red. Different colors have been used in order to show the results more clear, but there is not any special meaning for each color.

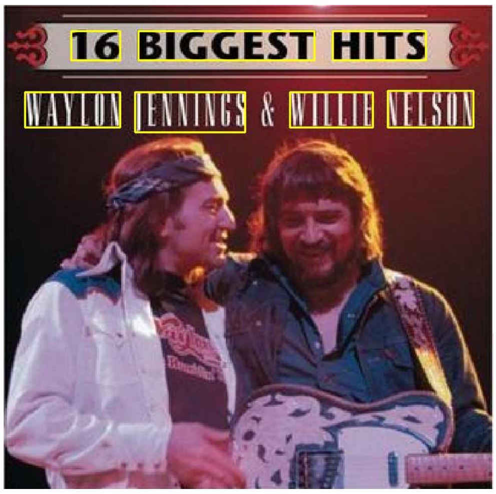
Like with the ICDAR 2003 and 2011 datasets, our text location method achieves high performance, as it is able to detect and precisely locate most of the text present in the images while producing a very low number of false positives. The biggest difficult of the CoverDB dataset is that the images have a low resolution, being the text very small and sometimes blurred. In addition, the text is embedded in complex backgrounds in many cases and, unlike the ICDAR 2003 and 2011, the CoverDB dataset has images with text with very artistic characteristics, such as text that emulates handwritting, like figure A.1(a), A.2(d), A.6(e) or A.7(d), or uncommon fonts, like figure A.5(a); transparent text, like figures A.7(b), A.10(b) and A.12(c); words whose characters have different font sizes, like “LED ZEPPELIN” in figure A.5(b); words with elements that play the role of a letter, like a star in the letter ‘O’ in “RINGO” in figure A.1(e) or in “BOOTSY COLLINS” in figure A.2(c); text with a certain deviation respect to the horizontal axis, like figure A.9(b); or words whose characters are quite far from each other, like “YUSUF” in figure A.3(e) and “Jimi Hendrix” in figure A.4(a). In general, as it happened with the ICDAR 2003 and 2011 datasets, the errors are due to a wrong segmentation, a wrong rejection of CCs by the Gaussian classifier, an incomplete CCs restoration and a wrong rejection of text lines by the text line classifier.



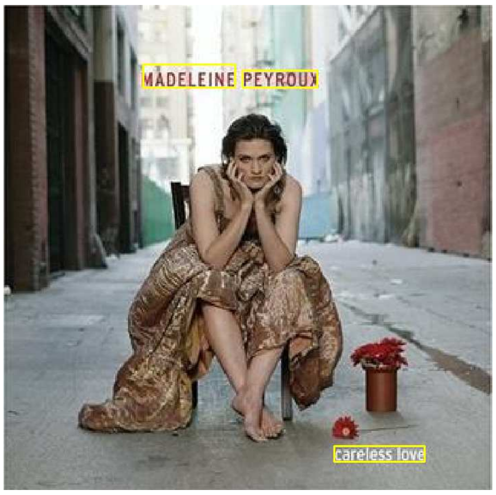
(a)



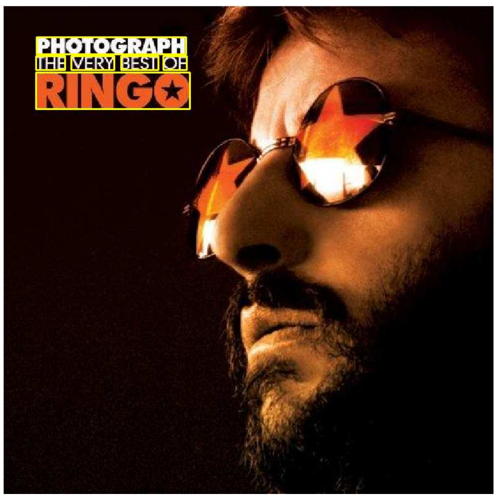
(b)



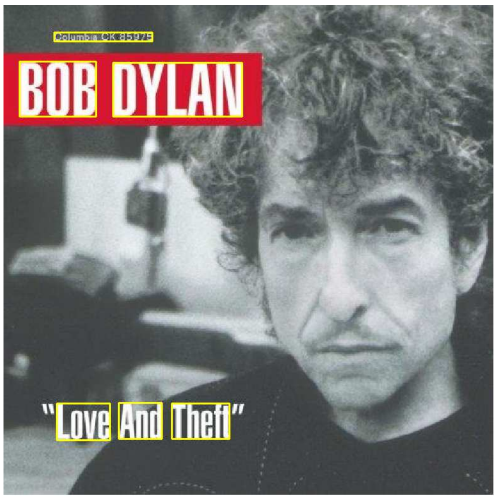
(c)



(d)

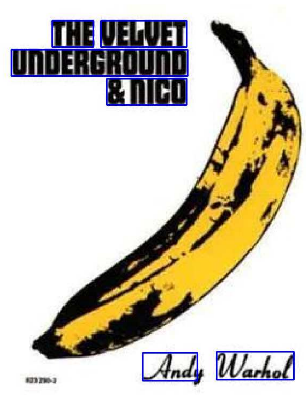


(e)

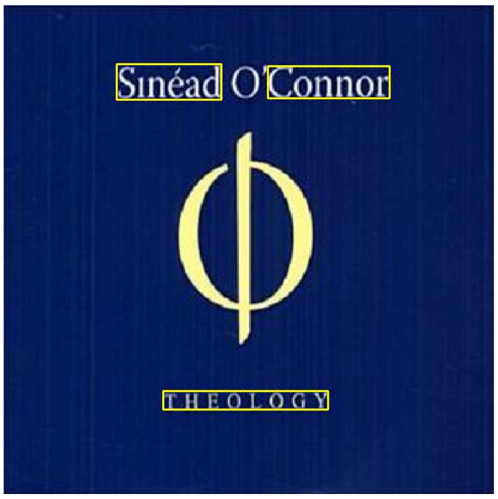


(f)

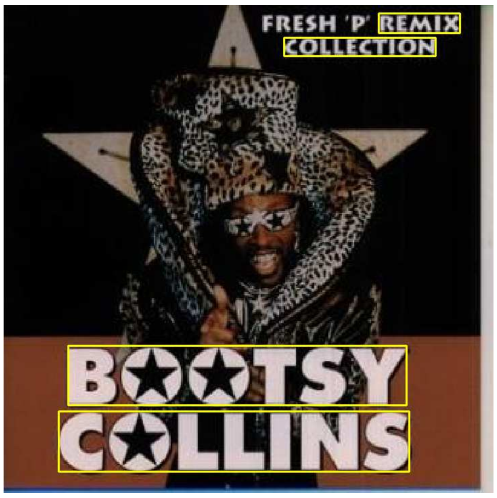
Figure A.1: Text detection results on some images from the CoverDB dataset.



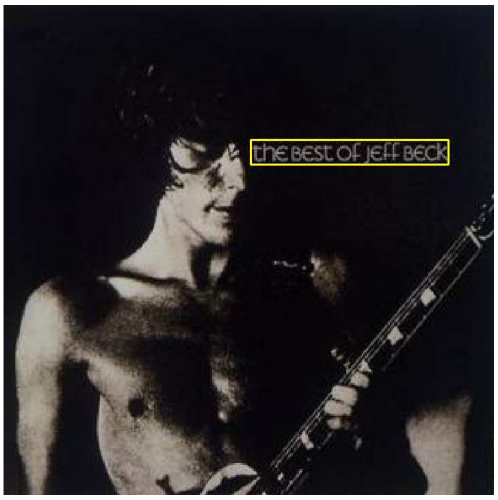
(a)



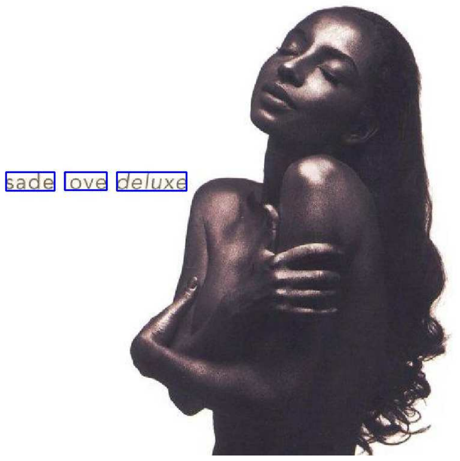
(b)



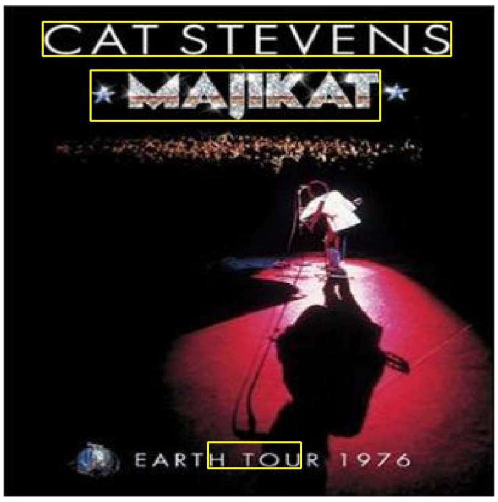
(c)



(d)

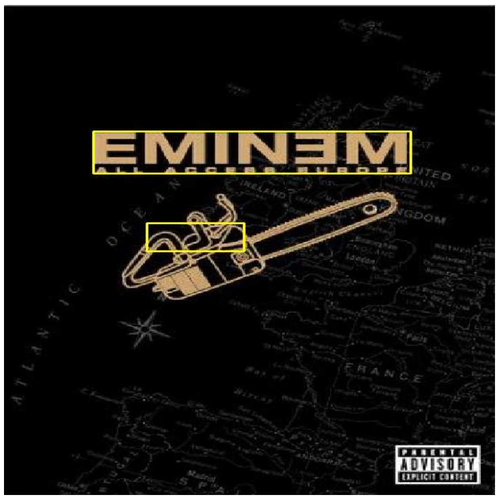


(e)

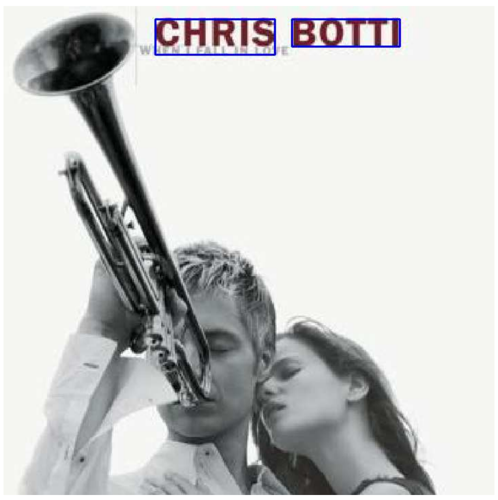


(f)

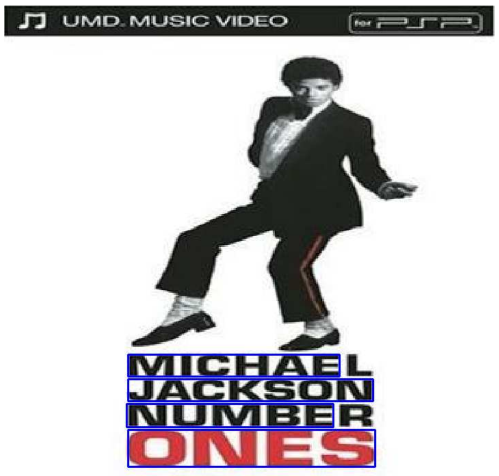
Figure A.2: Text detection results on some images from the CoverDB dataset.



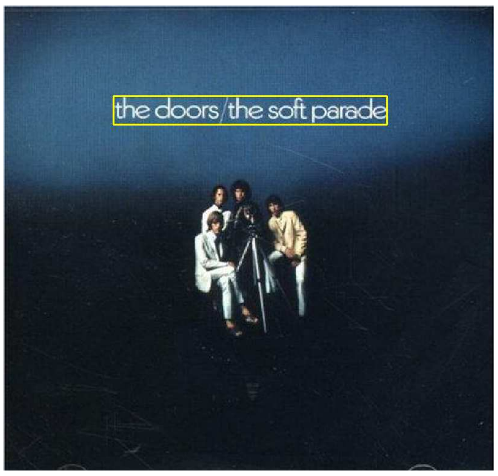
(a)



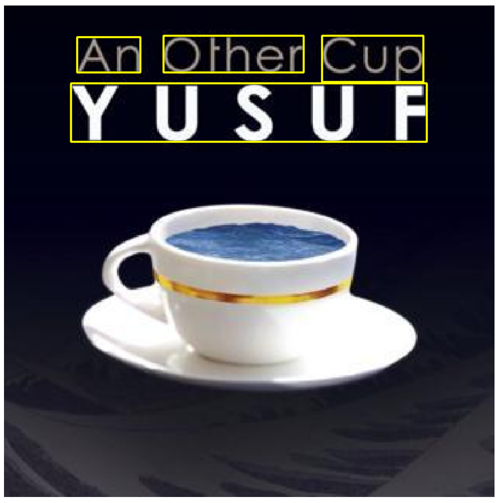
(b)



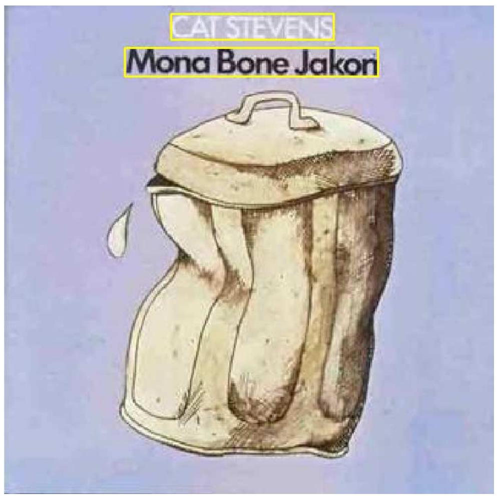
(c)



(d)

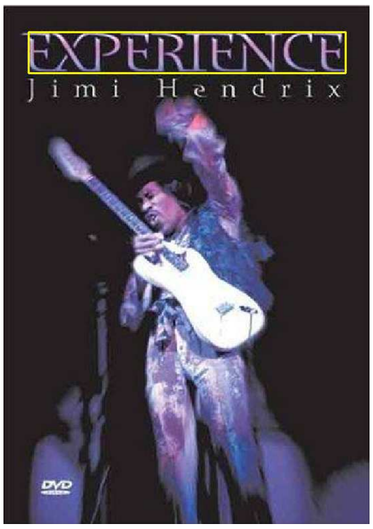


(e)

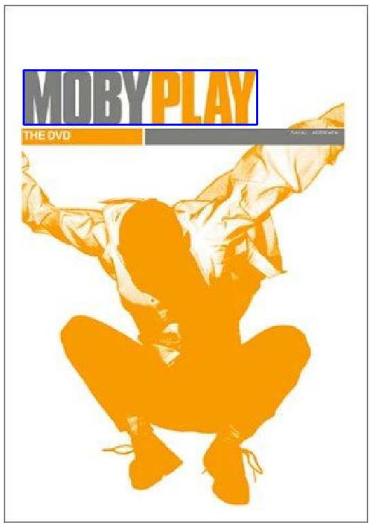


(f)

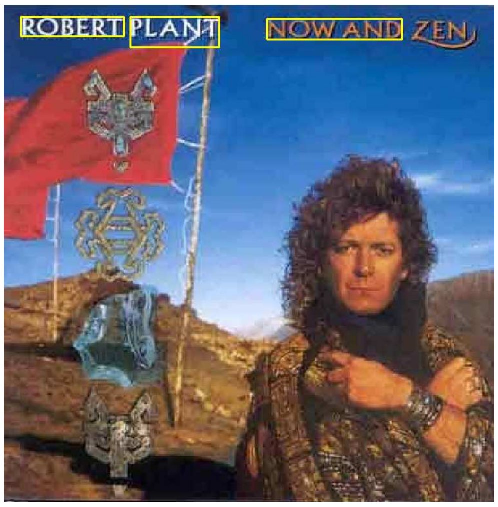
Figure A.3: Text detection results on some images from the CoverDB dataset.



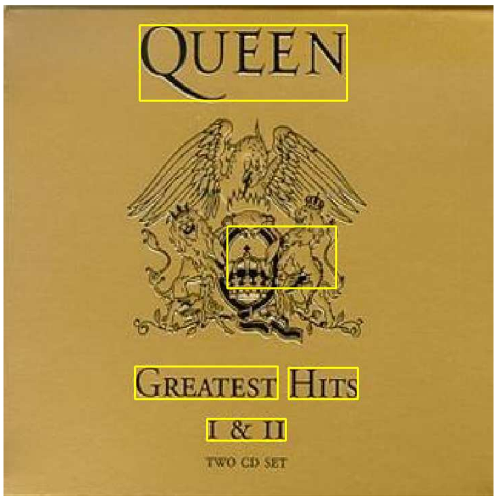
(a)



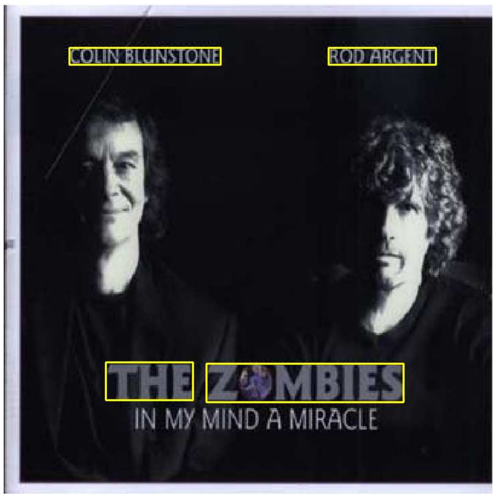
(b)



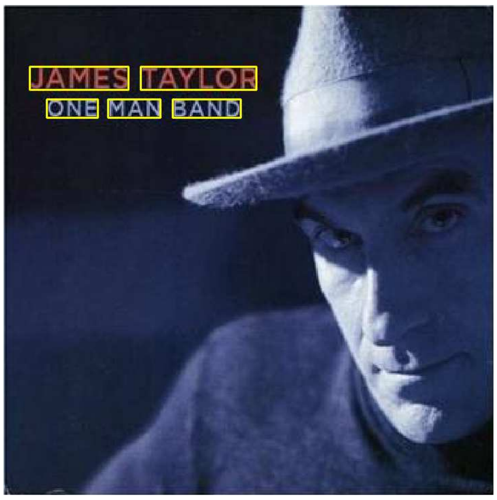
(c)



(d)

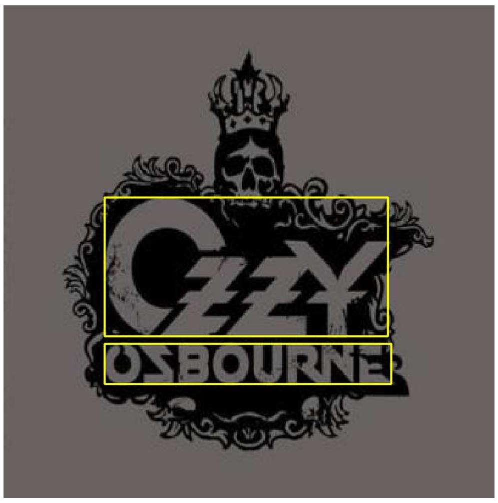


(e)

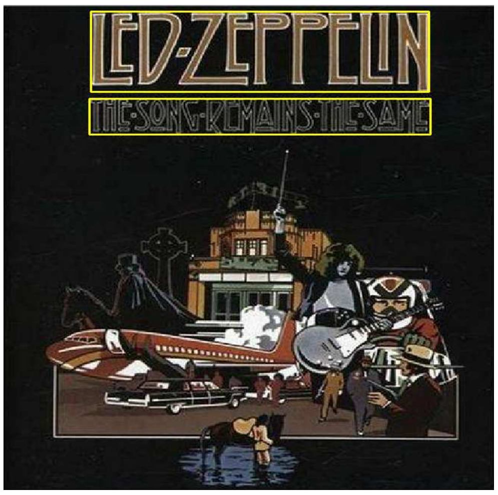


(f)

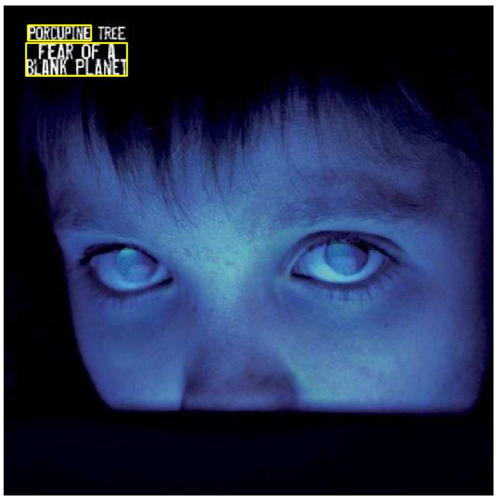
Figure A.4: Text detection results on some images from the CoverDB dataset.



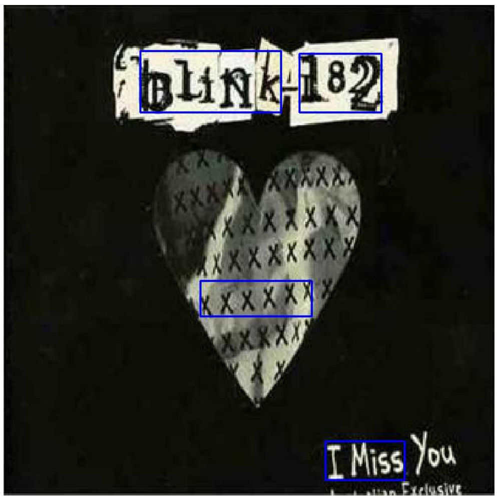
(a)



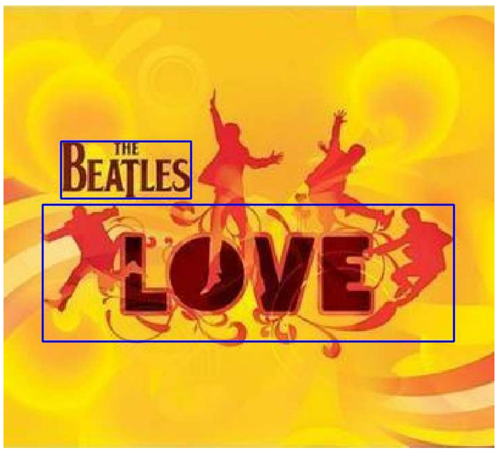
(b)



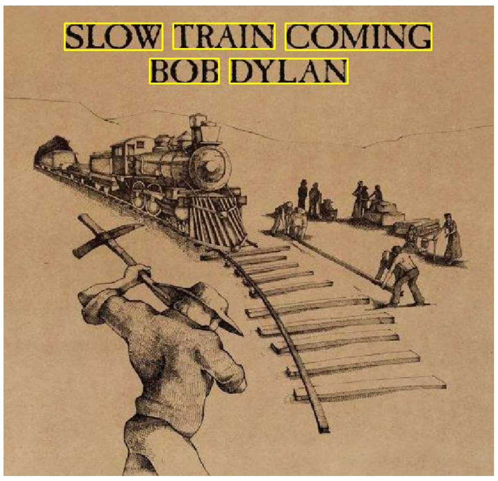
(c)



(d)

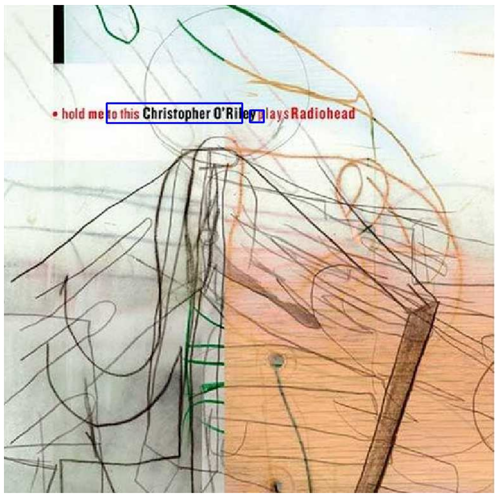


(e)

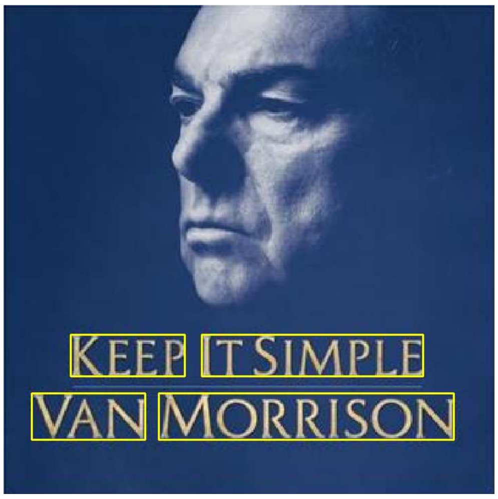


(f)

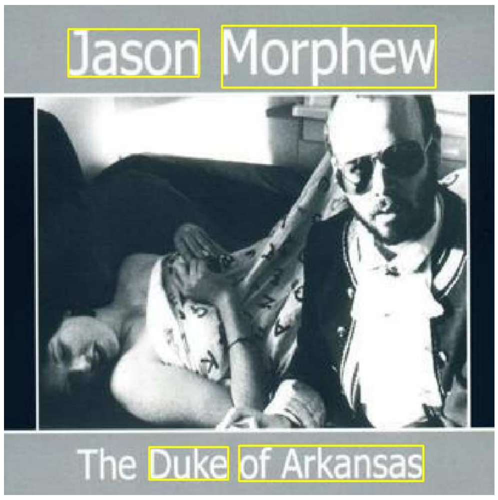
Figure A.5: Text detection results on some images from the CoverDB dataset.



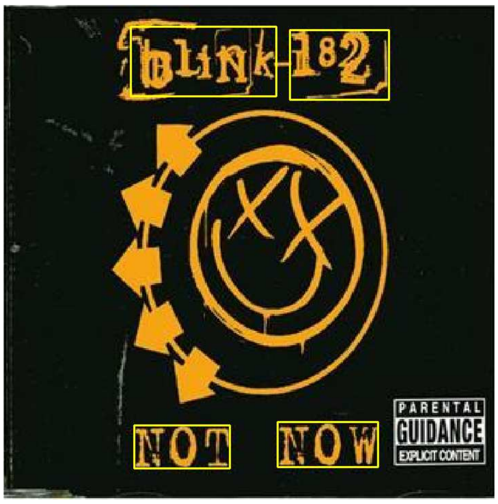
(a)



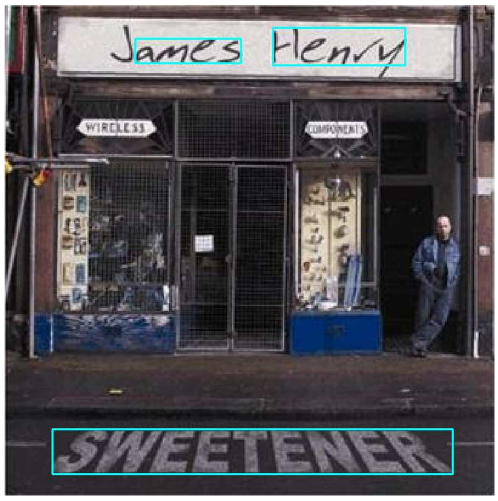
(b)



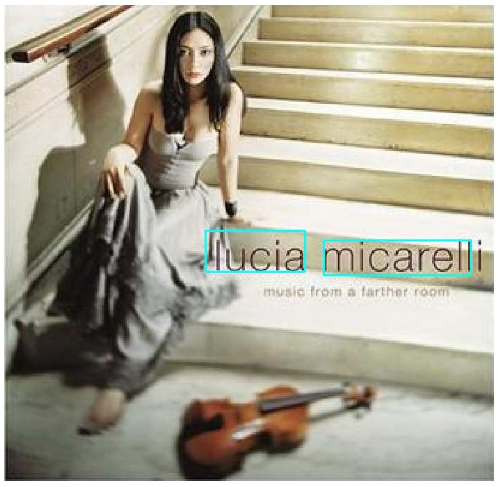
(c)



(d)

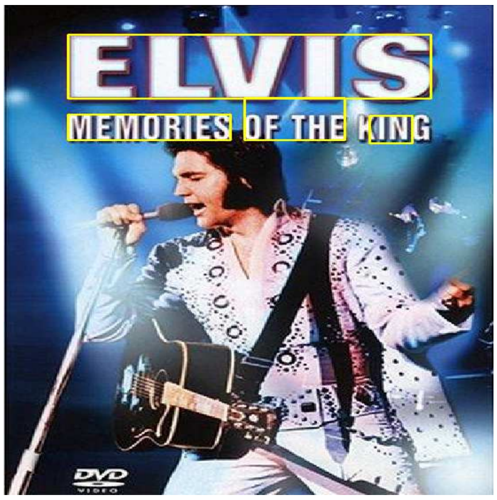


(e)

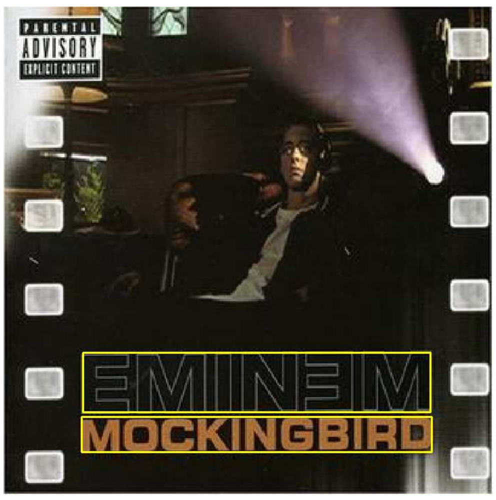


(f)

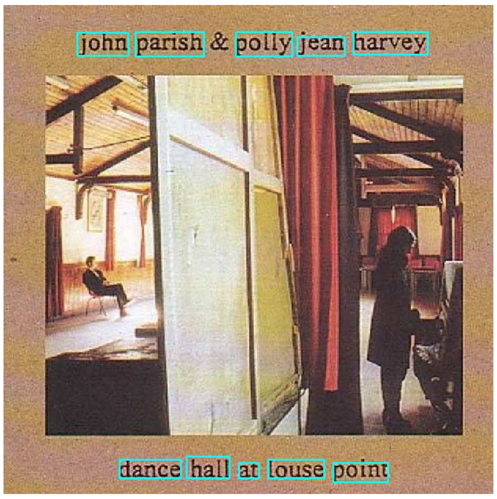
Figure A.6: Text detection results on some images from the CoverDB dataset.



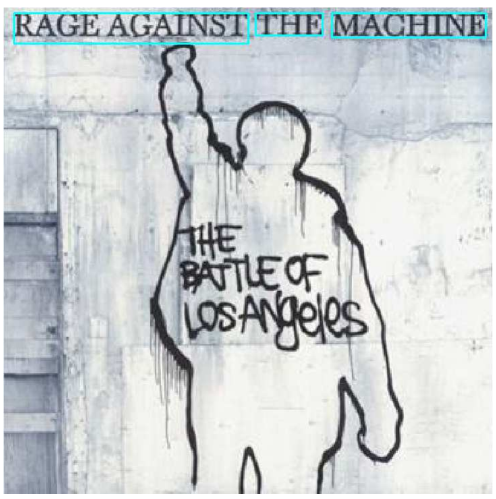
(a)



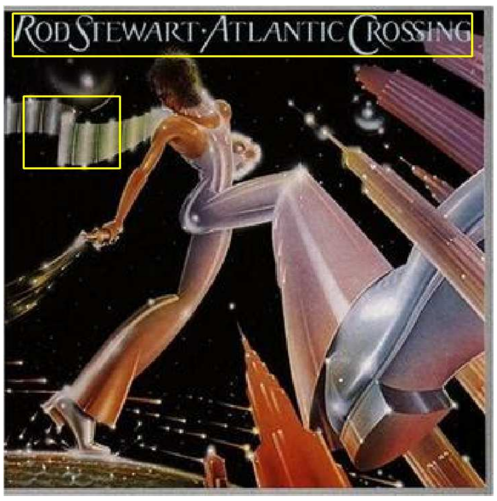
(b)



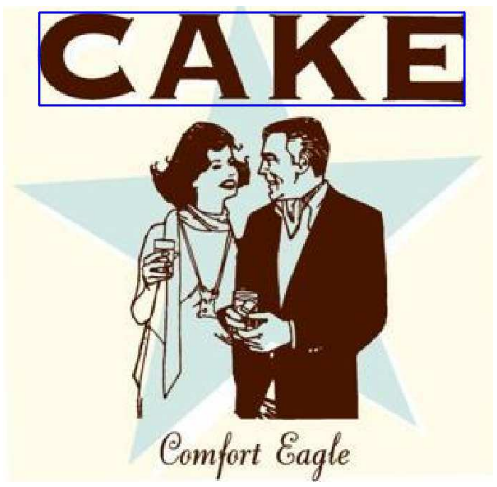
(c)



(d)



(e)



(f)

Figure A.7: Text detection results on some images from the CoverDB dataset.

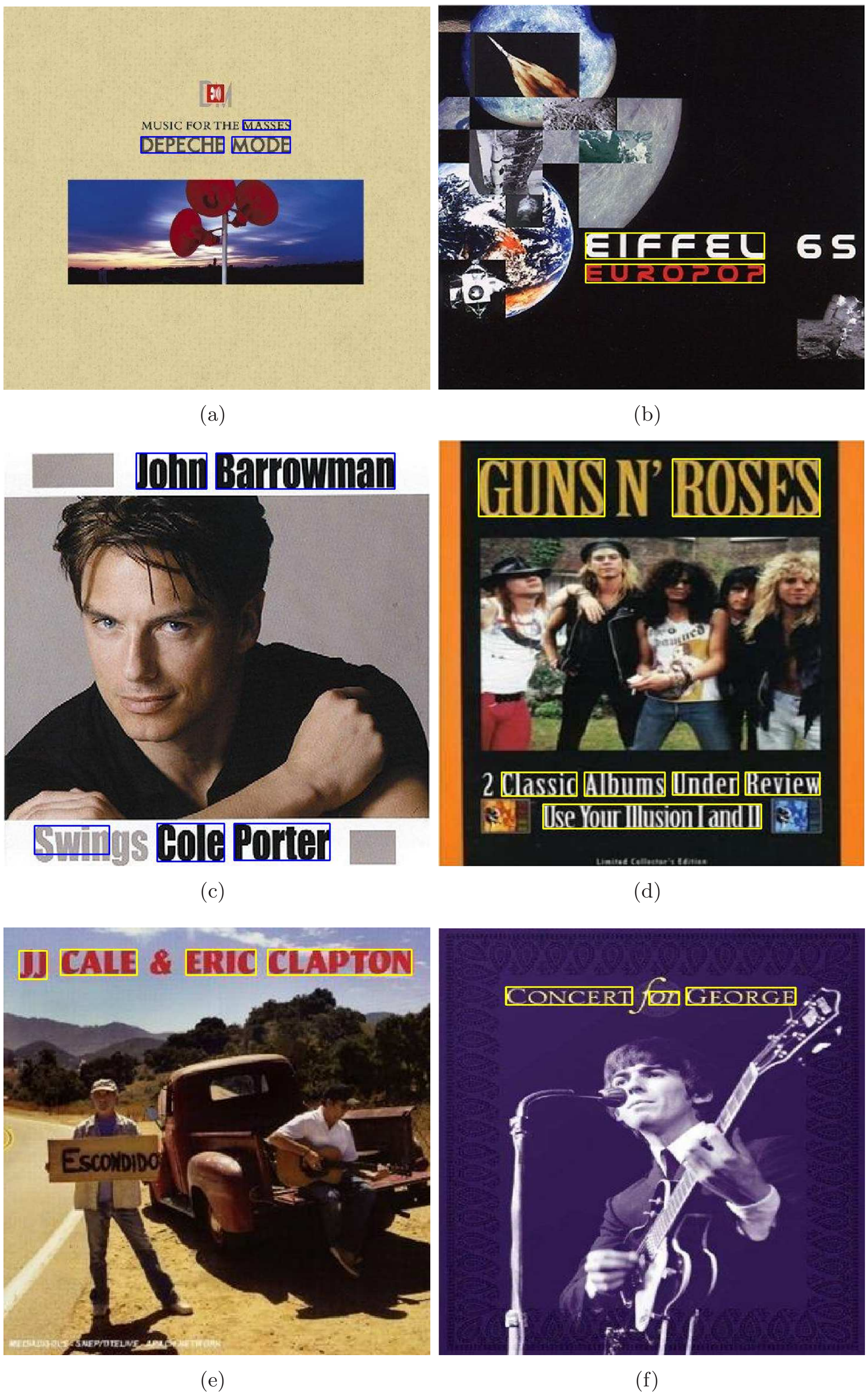
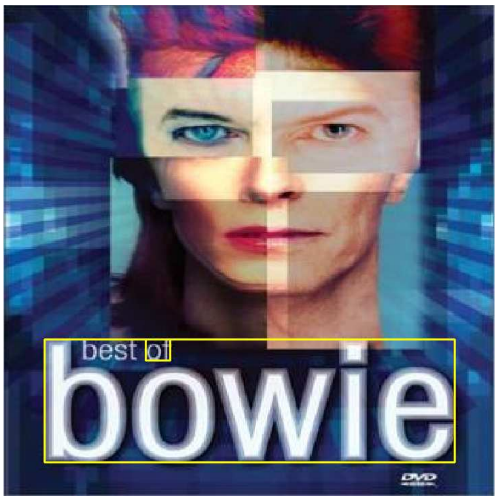
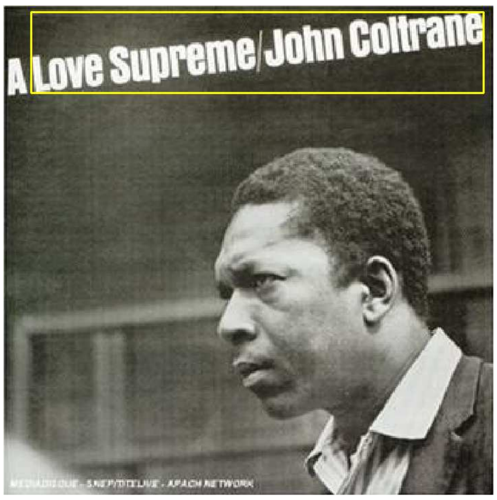


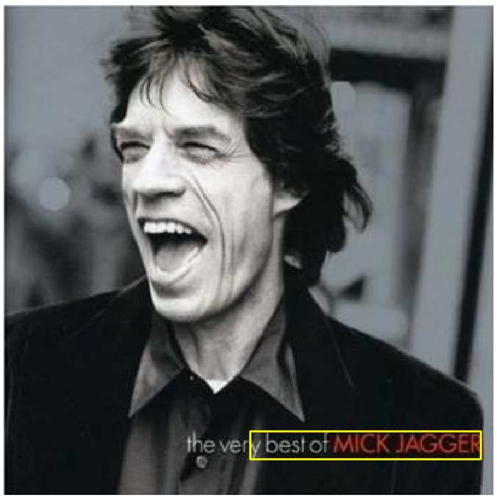
Figure A.8: Text detection results on some images from the CoverDB dataset.



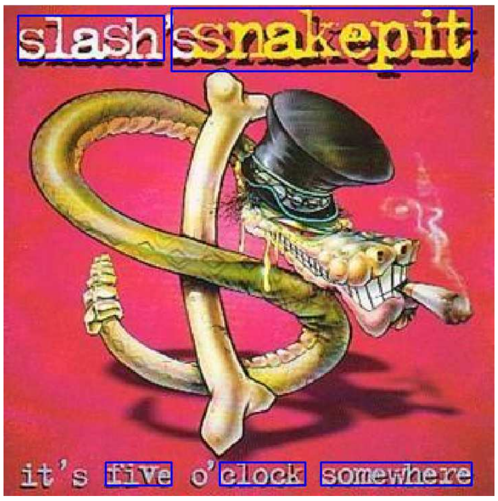
(a)



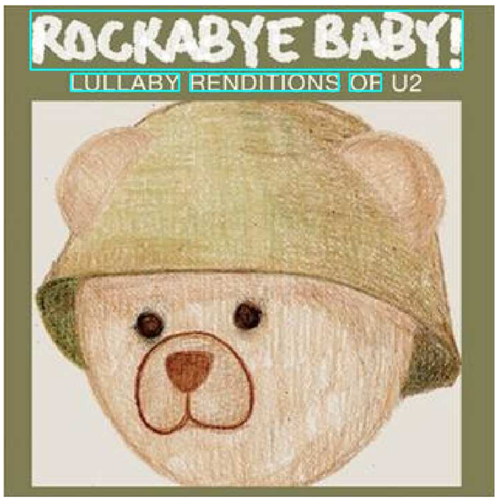
(b)



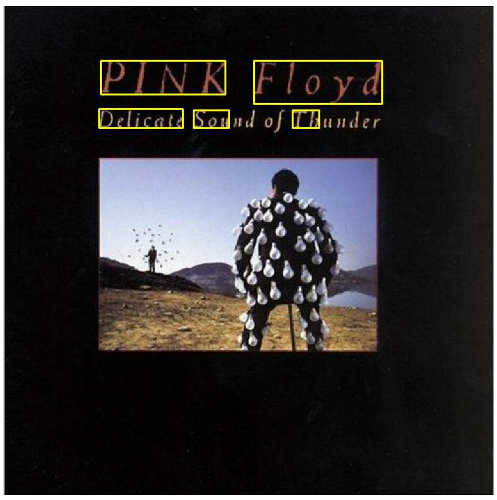
(c)



(d)

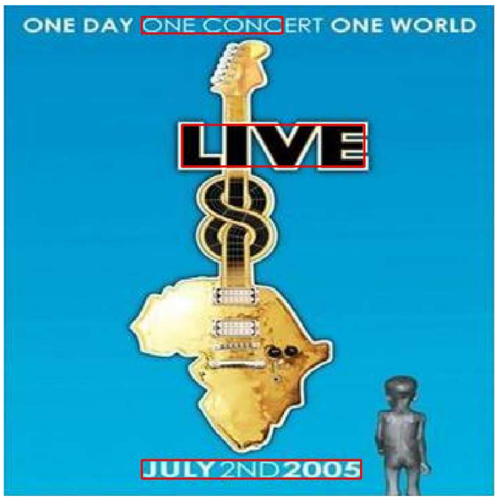


(e)

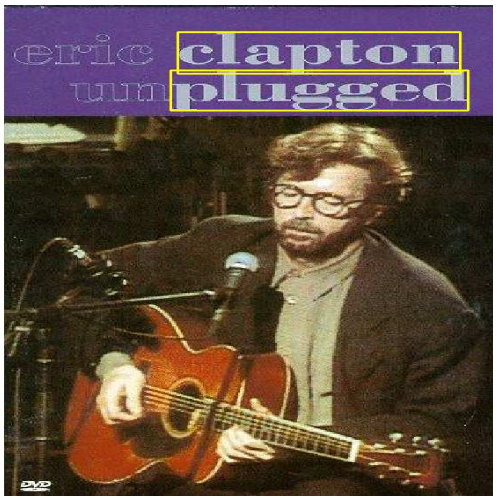


(f)

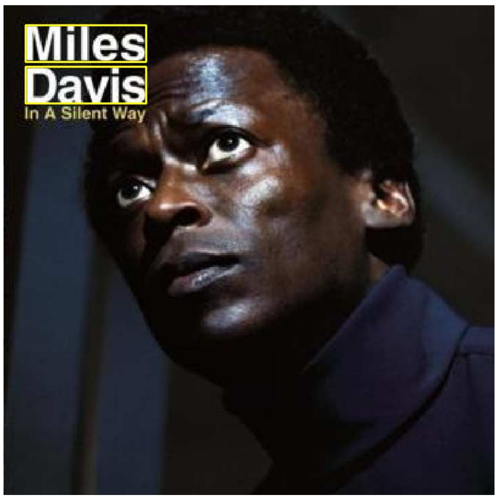
Figure A.9: Text detection results on some images from the CoverDB dataset.



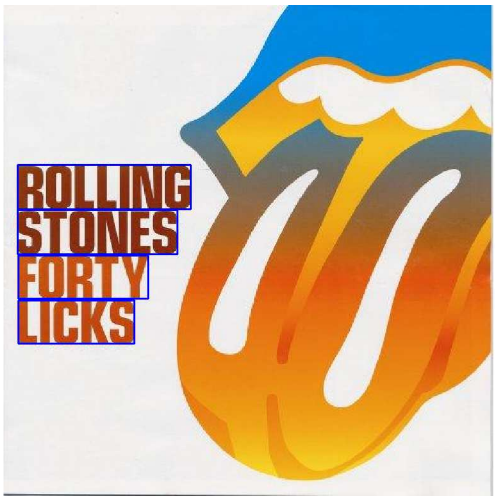
(a)



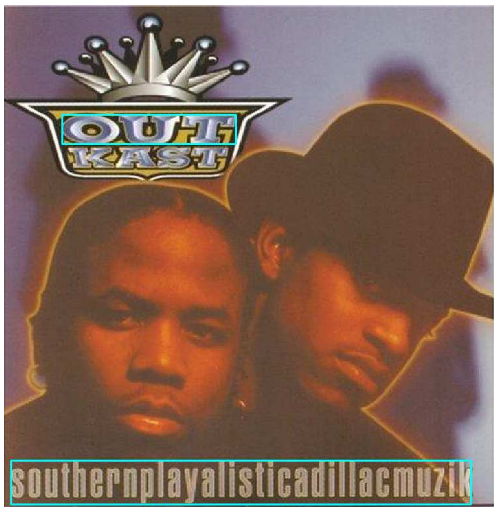
(b)



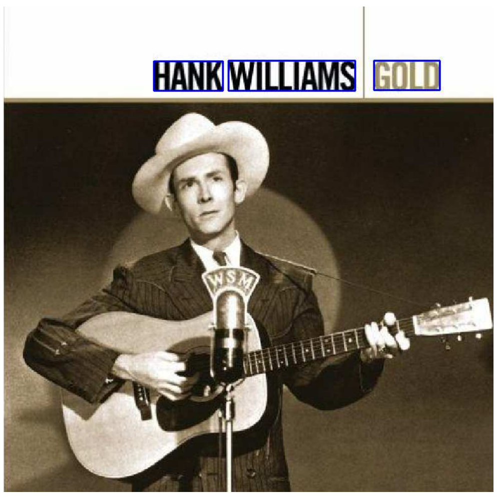
(c)



(d)

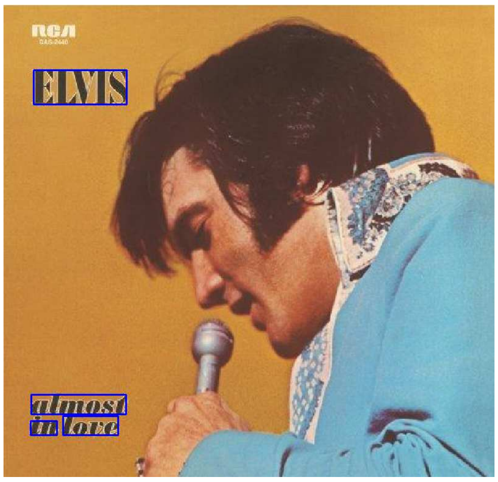


(e)

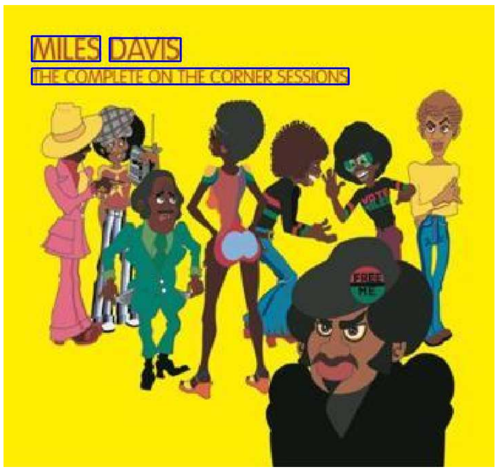


(f)

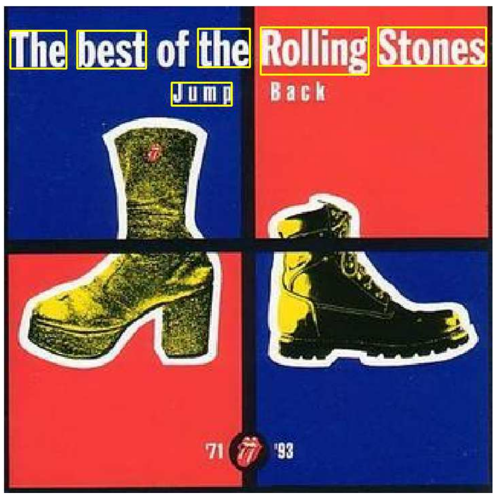
Figure A.10: Text detection results on some images from the CoverDB dataset.



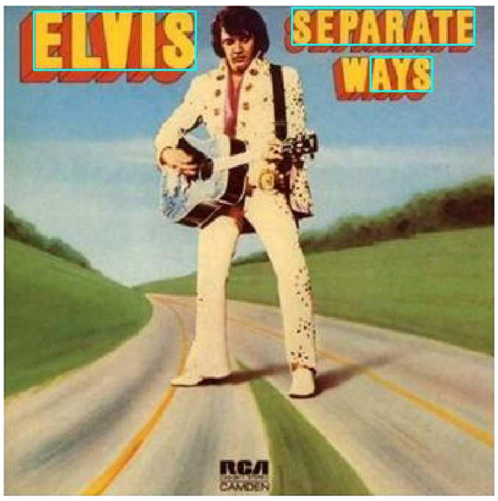
(a)



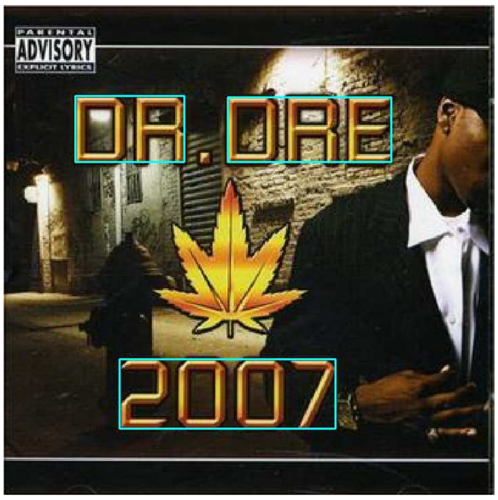
(b)



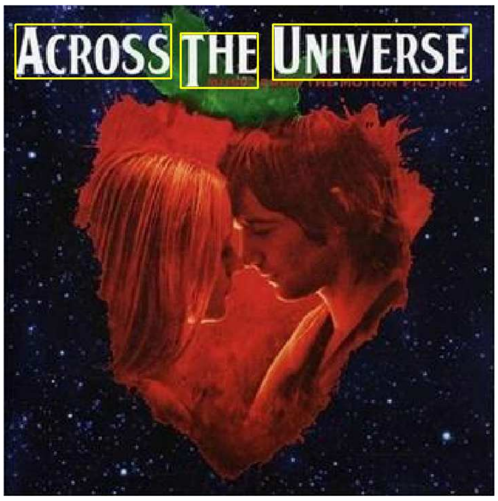
(c)



(d)



(e)



(f)

Figure A.11: Text detection results on some images from the CoverDB dataset.

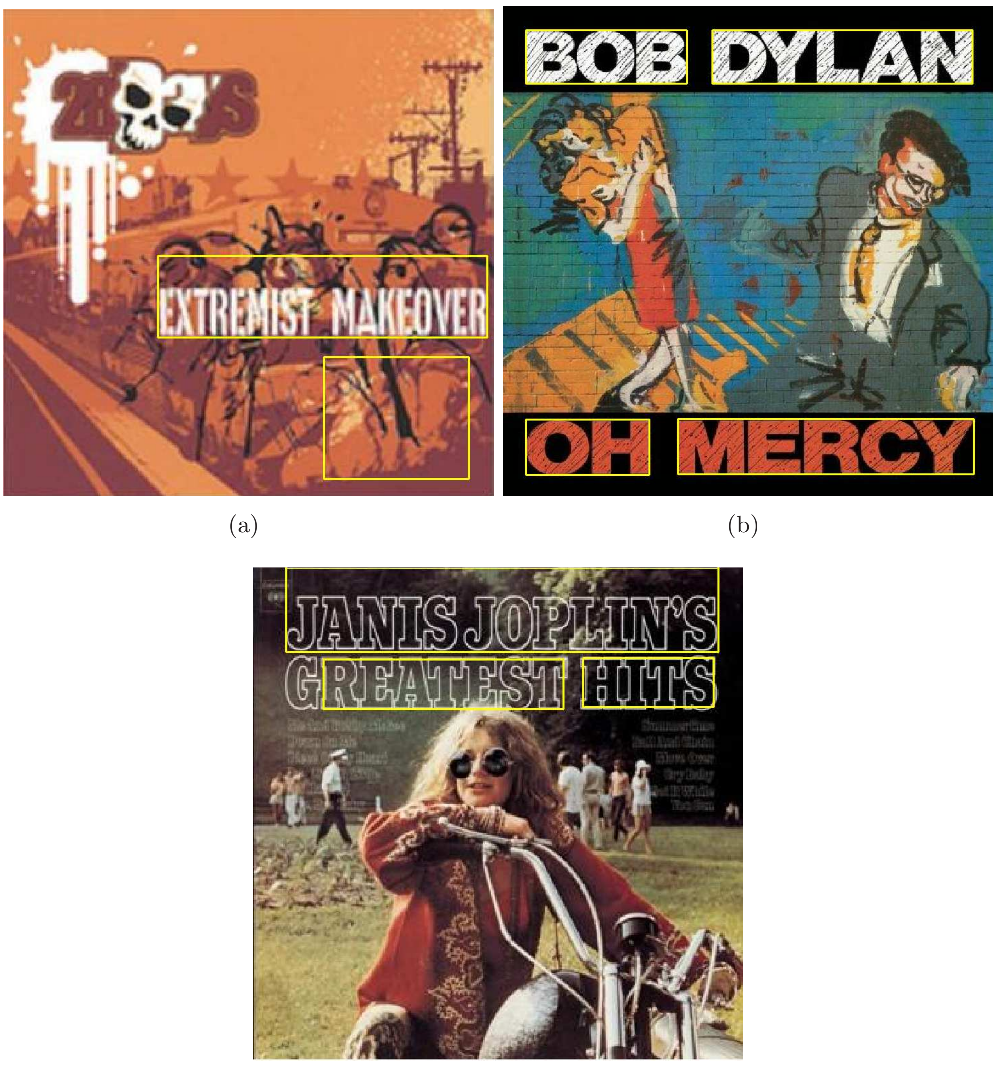


Figure A.12: Text detection results on some images from the CoverDB dataset.

Appendix B

The forward algorithm

The forward algorithm is an inference algorithm for Hidden Markov Models (HMMs) which computes the posterior marginals of all hidden state variables given a sequence of observations $O = O_1, O_2, \ldots O_T$. The algorithm makes use of the principle of dynamic programming to efficiently compute the values that are required to obtain the posterior marginal distributions in one pass.

A Hidden Markov Model is defined by the following parameters:

- N, the number of states in the model. The individual states are denoted as $S = S_1, S_2, \ldots, S_N$, and the state at time t as q_t.
- M, the number of distinct observation symbols per state, *i.e.* the discrete alphabet size. The observation symbols correspond to the physical output of the system being modeled. We denote the individual symbols as $V = V_1, V_2, \ldots, V_M$.
- The state transition probability distribution $A = a_{ij}$ where

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i] \qquad 1 \leq i, j \leq N \tag{B.1}$$

- The observation symbol probability distribution in state j, $B = b_j(k)$, where

$$b_j(k) = P[v_k \ at \ t | q_t = S_j] \qquad 1 \leq j \leq N, 1 \leq k \leq M \tag{B.2}$$

- The initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P[q_1 = S_i] \qquad 1 \leq i \leq N \tag{B.3}$$

Therefore, a complete specification of an HMM $\lambda = (N, M, A, B, \pi)$ requires five parameters. There are three basic problems of interest that must be solved for the model to be useful in real-word applications. These problems are the following:

1. Given the observation sequence $O = O_1, O_2, \ldots O_T$ and a model $\lambda = (N, M, A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model? This is solved using the forward algorithm.
2. Given the observation sequence $O = O_1, O_2, \ldots O_T$ and a model $\lambda = (N, M, A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1, q_2, \ldots q_T$ which is optimal in some meaningful sense (*i.e.*, best explains the observations)? This is solved using the Viterbi algorithm.

3. How do we adjust the model parameters $\lambda = (N, M, A, B, \pi)$ to maximize $P(O|\lambda)$? This is typically solved using the Baum-Welch algorithm.

In this appendix, we are showing the forward algorithm, as it is used in this thesis to compute the most likely word among all the words in a dictionary that may have generated a certain observed sequence.

Suppose that we wish to calculate the probability of the observation, $O = O_1, O_2, \dots O_T$, given the model $\lambda = (N, M, A, B, \pi)$, *i.e.* $P(O|\lambda)$. The most straightforward way of doing this is through enumerating every possible state sequence of length T (the number of observations). Consider one such fixed state sequence

$$Q = q_1, q_2, \dots q_T \tag{B.4}$$

where q_1 is the initial state. The probability of the observation sequence O for the state sequence of (B.4) is

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(O_t|q_t, \lambda) \tag{B.5}$$

where we assume statistical independence of observations. Thus, we get

$$P(O|Q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdot \ldots \cdot b_{q_T}(O_T) \tag{B.6}$$

The probability of such a state sequence Q can be written as

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T} \tag{B.7}$$

The joint probability of O and Q, *i.e.* the probability that O and Q occur simultaneously, is simply the product of the above two terms, *i.e.*

$$P(O, Q|\lambda) = P(O|Q, \lambda) P(Q|\lambda) \tag{B.8}$$

The probability of O (given the model) is obtained by summing this joint probability over all possible state sequences q giving

$$\begin{aligned} P(O|\lambda) &= \sum_{all\ Q} P(O|Q, \lambda) P(Q|\lambda) = \\ &= \sum_{q_1, q_2, \dots q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T) \end{aligned} \tag{B.9}$$

The interpretation of the computation in the above equation is the following. Initially (at time $t = 1$) we are in state q_1 with probability π_{q_1}, and generate the symbol O_1 (in this state) with probability $b_{q_1}(O_1)$. The clock changes from time t to time $t+1$ and we make a transition to state q_2 from state q_1 with probability $a_{q_1 q_2}$, and generate symbol O_2 with probability $b_{q_2}(O_2)$. This process continues until we make the transition at time T from state q_{T-1} to state q_T with probability $a_{q_{T-1} q_T}$ and generate symbol O_T with probability $b_{q_T}(O_T)$.

The forward algorithm calculates $P(O|\lambda)$ efficiently. Consider the forward variable $\alpha_t(j)$ defined as

$$\alpha_t(j) = P(O_1, O_2 \dots O_t, q_t = S_j|\lambda) \tag{B.10}$$

i.e., the probability of the partial observation sequence $O_1, O_2, \ldots O_t$ (until time t) and state S_j at time t, given the model λ. We can solve $\alpha_t(j)$ inductively, as follows:

1. Initialization:
$$\alpha_1(j) = \pi_j b_j(O_1), \qquad 1 \leq j \leq N \tag{B.11}$$

2. Induction:
$$\alpha_{t+1}(j) = \sum_{j=1}^{N} \pi_j b_j(O_{t+1}), \qquad 1 \leq t \leq T-1, 1 \leq j \leq N \tag{B.12}$$

3. Termination:
$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_T(j), \qquad 1 \leq t \leq T-1, 1 \leq j \leq N \tag{B.13}$$

Appendix C

Conversion of coordinates

Geographic coordinate systems enable every location on the Earth to be specified by a set of numbers and/or letters. There are many ways of representing the coordinates, but the two more popular systems are the one based on latitude and longitude and the one known as UTM (Universal Transverse Mercator). In this appendix, both coordinate systems are presented and the formulation to transform values from one system to the other one is detailed.

The most common coordinate system uses degrees of latitude and longitude to describe a location on the Earth's surface. Lines of latitude run parallel to the equator and divide the Earth into 180 equal portions from north to south. The reference latitude is the equator and each hemisphere is divided into 90 equal portions, each representing one degree of latitude.

In the northern hemisphere, degrees of latitude are measured from 0^o at the equator to 90^o at the north pole. In the southern hemisphere, degrees of latitude are measured from 0^o at the equator to 90^o at the south pole. To simplify the digitalization of maps, degrees of latitude in the southern hemisphere are often assigned negative values (0^o to -90^o). Wherever you are on the Earth's surface, the distance between lines of latitude is the same (60 nautical miles), so they conform to the uniform grid criterion assigned to a useful grid system.

On the other hand, lines of longitude do not stand up so well to the standard of uniformity. Lines of longitude run perpendicular to the equator and converge at the poles. The reference line of longitude (the prime meridian) runs from the north pole to the south pole through Greenwich, England. Subsequent lines of longitude are measured from 0^o to 180^o east or west (values west to the prime meridian are assigned negative values for use in digital mapping applications) of the prime meridian.

At the equator, the distance represented by one line of longitude is equal to the distance represented by one degree of latitude. As you move towards the poles, the distance between lines of longitude becomes progressively less until, at the exact location of the pole, all 360^o of longitude are represented by a single point.

To be truly useful, a map grid should be divided into small enough sections that can be used to describe, with an acceptable level of accuracy, the location of a point on the map. To accomplish this, degrees are divided into minutes and seconds. There are 60 minutes in a degree and 60 seconds in a minute (3600 seconds in a degree).

An alternative method of notation is the decimal degree system, in which the major units (degrees) are the same, but rather than using minutes and seconds, smaller increments are represented as a decimal of a degree.

The main problem of the geographic reference system based on latitude and longitude

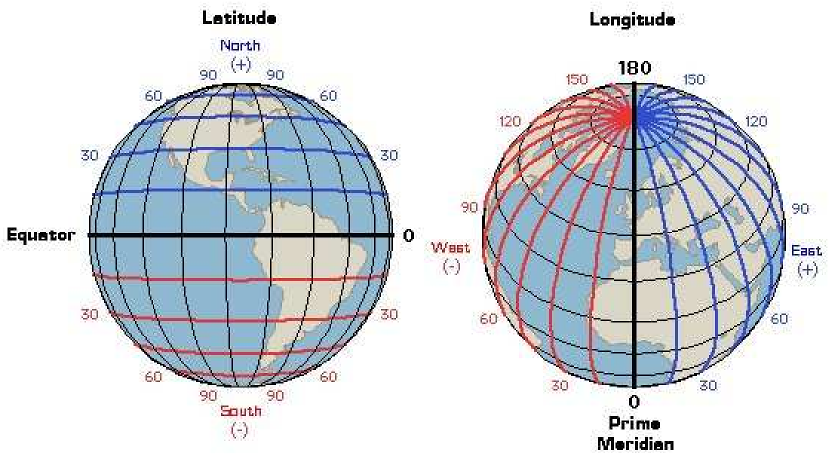


Figure C.1: Latitude and longitude of the Earth

is that the grid on a map is not constant from north to south. This problem is solved to some extent by the UTM coordinate system.

The UTM coordinate system uses a 2-dimensional Cartesian coordinate system to give locations on the surface of the Earth. It divides the Earth into 60 zones (UTM zones), each a six-degree band of longitude, and uses a Mercator projection in each zone. A Mercator projection is a pseudocylindrical conformal projection, which preserves shape.

The first UTM zone begins at the International Date Line. The zones are numbered from west to east, so zone 2 begins at 174°W and extends to 168°W. The last zone (zone 60) begins at 174ºE and extends to the International Date Line.

The zones are then further subdivided into an eastern and western half by drawing a line, representing a Mercator projection, down the middle of the zone. This line is known as the "central meridian" and is the only line within the zone that can be drawn between the poles and is perpendicular to the equator (in other words, it is the new "equator" for the projection and suffers the least amount of distortion). For this reason, vertical grid lines in the UTM system are oriented parallel to the central meridian. The central meridian is also used in setting up the origin for the grid system.

Each zone is segmented into 20 latitude bands. Each latitude band is 8 degrees high, and is lettered starting from 'C' at 80°S, increasing up the English alphabet until 'X', omitting the letters 'I' and 'O' (because of their similarity to the numerals one and zero). The last latitude band, 'X', is extended an extra 4 degrees, so it ends at 84°N latitude, thus covering the northernmost land on Earth. Latitude bands 'A' and 'B' do exist, as do bands 'Y' and 'Z'. They cover the western and eastern sides of the Antarctic and Arctic regions respectively. A convenient mnemonic to remember is that the letter 'N' is the first letter in the northern hemisphere, so any letter coming before 'N' in the alphabet is in the southern hemisphere, and any letter 'N' or after is in the northern hemisphere (see figure C.2).

Any point can then be described by its distance east of the origin (its "easting" value). By definition, the central meridian is assigned a false easting of 500,000 meters. Any easting value greater than 500,000 meters indicates a point east of the central meridian. Any easting value less than 500,000 meters indicates a point west of the central meridian. Positions in the UTM system are measured in meters, and each UTM zone has its own origin for east-west measurements.

To eliminate the necessity for using negative numbers to describe a location, the east-

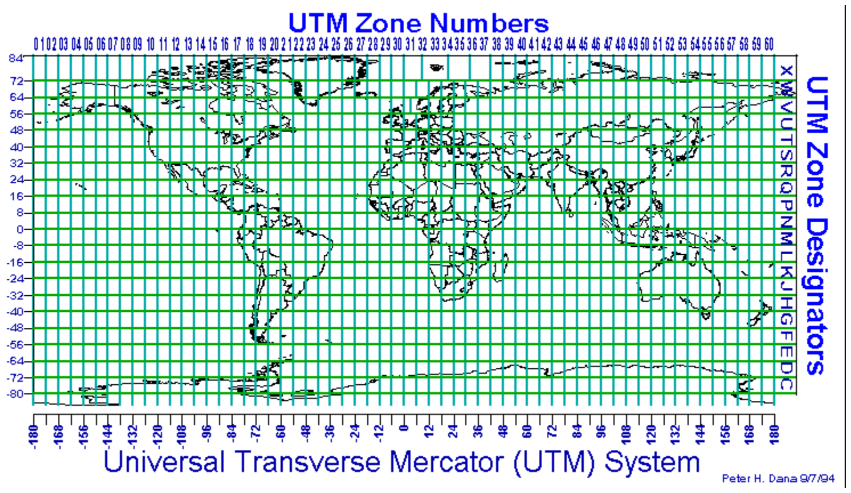


Figure C.2: UTM zones

west origin is placed 500,000 meters west of the central meridian. This is referred to as the zone's "false origin". The zone does not extend all the way to the false origin. The origin for north-south values depends on whether you are in the northern or southern hemisphere. In the northern hemisphere, the origin is the equator and all the distance north (or "northings") are measured from the equator. In the southern hemisphere, the origin is the south pole and all northings are measured from there. Once again, having separated origins for the northern and southern hemispheres eliminates the need for any negative values.

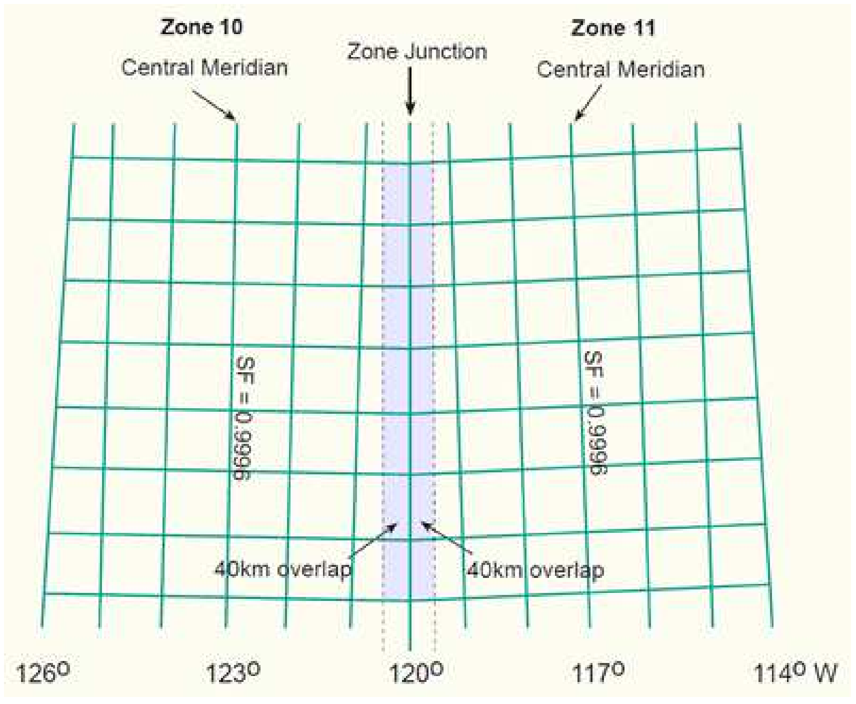


Figure C.3: An UTM zone

C.1 Converting Latitude and Longitude to UTM coordinates

Let $s_a = 6,378,137$ m and $s_b = 6,356,752.3142$ m be the radius of the Earth in the equator and in the pole, respectively, and $e = \frac{\sqrt{s_a^2 - s_b^2}}{s_b}$ and $c = \frac{s_a^2}{s_b}$ be the eccentricity of the Earth's elliptical cross-section and the radius of curvature of the Earth in the meridian plane, respectively.

Given the latitude lat and longitude lon of a point in radians, the UTM zone is computed using (C.1).

$$Zone = fix(\frac{lon \cdot \frac{180}{\pi}}{6} + 31) \tag{C.1}$$

where $fix()$ rounds to the nearest integer towards zero.

The designator of the UTM zone ('C'-'W') is given by the latitude value, as shown in figure C.2.

The following parameters are the core of the formulation that enable to compute the easting and northing values:

- $S = Zone \cdot 6 - 183$
- $\delta_S = lon - S \cdot \frac{\pi}{180}$
- $a = \cos(lat) \cdot \sin(\delta_S)$
- $\epsilon = 0.5 \cdot \log(\frac{1+a}{1-a})$
- $\nu = \arctan(\frac{\tan(lat)}{\cos \delta_S}) - lat$
- $v = 0.9996 \cdot \frac{c}{\sqrt{1+e^2 \cdot \cos^2(lat)}}$
- $t = \frac{e^2}{2} \cdot \epsilon^2 \cdot \cos^2(lat)$
- $a_1 = \sin(2 \cdot lat)$
- $a_2 = a_1 \cdot \cos^2(lat)$
- $j_2 = lat + \frac{a_1}{2}$
- $j_4 = \frac{3 \cdot j_2 + a_2}{4}$
- $j_6 = \frac{5 \cdot j_4 + a_2 \cdot \cos^2(lat)}{3}$
- $\alpha = 0.75 \cdot e^2$
- $\beta = \frac{5}{3} \cdot \alpha^2$
- $\gamma = \frac{35}{27} \cdot \alpha^3$
- $B_m = 0.9996 \cdot c \cdot (lat - \alpha \cdot j_2 + \beta \cdot j_4 - \gamma \cdot j_6)$

The easting and northing are computed using (C.2) and (C.3).

$$X = \epsilon \cdot v \cdot (1 + \frac{t}{3}) + 500,000 \tag{C.2}$$

$$Y = \nu \cdot v \cdot (1 + t) + B_m \tag{C.3}$$

In case we obtain a negative value for the northing ($Y < 0$), it means that the point is in the southern hemisphere, and a value of 10,000,000 must be added: $Y = Y + 10,000,000$.

C.2 Converting UTM coordinates to Latitude and Longitude

Let $s_a = 6,378,137$ m and $s_b = 6,356,752.3142$ m be the radius of the Earth in the equator and in the pole, respectively, and $e = \frac{\sqrt{s_a^2 - s_b^2}}{s_b}$ and $c = \frac{s_a^2}{s_b}$ be the eccentricity of the Earth's elliptical cross-section and the radius of curvature of the Earth in the meridian plane, respectively.

Given the easting X and the northing Y of a position in meters, as well as its UTM zone described by its number $Zone$ and its designator Des, the hemisphere (northern or southern) is given by Des according to figure C.2. Then, the easting value is corrected ($X = X - 500,000$), as well as the northing in case it corresponds to a position in the southern hemisphere ($Y = Y - 10,000,000$).

A series of parameters to compute the latitude and longitude are needed:

- $S = Zone \cdot 6 - 183$
- $L = \frac{Y}{6366197.724 \cdot 0.9996}$
- $v = 0.9996 \cdot \frac{c}{\sqrt{1 + e^2 \cdot \cos^2(L)}}$
- $a = \frac{X}{v}$
- $a_1 = \sin(2 \cdot L)$
- $a_2 = a_1 \cdot \cos^2(L)$
- $j_2 = L + \frac{a_1}{2}$
- $j_4 = \frac{3 \cdot j_2 + a_2}{4}$
- $j_6 = \frac{5 \cdot j_4 + a_2 \cdot \cos^2(L)}{3}$
- $\alpha = 0.75 \cdot e^2$
- $\beta = \frac{5}{3} \cdot \alpha^2$
- $\gamma = \frac{35}{27} \cdot \alpha^3$
- $B_m = 0.9996 \cdot c \cdot (L - \alpha \cdot j_2 + \beta \cdot j_4 - \gamma \cdot j_6)$
- $b = \frac{Y - B_m}{v}$
- $\xi = \frac{e^2 \cdot a^2}{2} \cdot \cos^2(L)$
- $\zeta = a \cdot (1 - \frac{\xi}{3})$
- $\eta = b \cdot (1 - \zeta) + L$
- $\sinh(\zeta) = \frac{\exp(\zeta) - \exp(-\zeta)}{2}$

- $\Delta\lambda = \arctan(\frac{\sinh \zeta}{\cos \eta})$
- $\tau = \arctan(\cos \Delta\lambda \cdot \tan \eta)$

The latitude and longitude in radians are computed according to (C.4) and (C.5), respectively.

$$lat = L + (1 + e^2 \cdot \cos^2(L) - \frac{3}{2} \cdot e^2 \cdot \sin(L) \cdot \cos(L) \cdot (\tau - L)) \cdot (\tau - L) \tag{C.4}$$

$$lon = \Delta\lambda + S \tag{C.5}$$

Bibliography

ABBYY (2009). ABBYY FineReader. [web page] `http://finereader.abbyy.com/`.

Abdel-Hakim, A. E. and Farag, A. A. (2006). Csift: A sift descriptor with color invariant characteristics. In *CVPR*, pages 1978–1983.

Bai, H., Sun, J., Naoi, S., Katsuyama, Y., Hotta, Y., and Fujimoto, K. (2008). Video caption duration extraction. In *Intl. Conf. on Pattern Recognition (ICPR)*, pages 1–4.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. J. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522.

Berg, A. C., Berg, T. L., and Malik, J. (2005). Shape matching and object recognition using low distortion correspondences. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 26–33.

Cano, J. and Perez-Cortes, J. C. (2003). Vehicle license plate segmentation in natural images. *Lecture Notes on Computer Science 2652*, pages 142–149.

Chan, W.-L. and Pun, C.-M. (2011). Robust character recognition using connected-component extraction. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 310–313.

Chen, H., Tsai, S. S., Schroth, G., Chen, D. M., Grzeszczuk, R., and Girod, B. (2011). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *Intl. Conf. on Image Processing (ICIP)*.

Chen, X. and Yuille, A. L. (2004). Detecting and reading text in natural scenes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 366–373.

Chowdhury, A. R., Bhattacharya, U., and Parui, S. K. (2012). Scene text detection using sparse stroke information and mlp. In *ICPR*, pages 294–297.

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

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893.

de Campos, T. E., Babu, B. R., and Varma, M. (2009). Character recognition in natural images. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 273–280.

Epshtein, B., Ofek, E., and Wexler, Y. (2010). Detecting text in natural scenes with stroke width transform. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2963–2970. IEEE.

Escalera, S., Baro, X., Vitria, J., and Radeva, P. (2009). Text detection in urban scenes. *Frontiers in Artificial Intelligence and Applications*, 202:35–44.

Fehli, M., Bonnier, N., and Tabonne, S. (2012). A skeleton based descriptor for detecting text in real scene images. In *ICPR*, pages 282–285.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Gómez-Moreno, H., Maldonado-Bascón, S., Gil-Jiménez, P., and Lafuente-Arroyo, S. (2010). Goal evaluation of segmentation algorithms for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 11(4):917–930.

Google (2010). Tesseract OCR. [web page] `http://code.google.com/p/tesseract-ocr/`.

Hanif, S. M. and Prevost, L. (2009). Text detection and localization in complex scene images using constrained adaboost algorithm. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1–5.

Holley, R. (2009). How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs.

Hua, X.-S., Yin, P., and Zhang, H. (2002). Efficient video text recognition using multiple frame integration. In *Intl. Conf. on Image Processing (ICIP)*, pages 397–400.

Jin, Y. and Geman, S. (2006). Context and hierarchy in a probabilistic image model. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2145–2152.

Karatzas, D. and Antonacopoulos, A. (2007). Colour text segmentation in web images based on human perception. *Image Vision Comput.*, 25(5):564–577.

Karatzas, D., Mestre, S. R., Mas, J., Nourbakhsh, F., and Roy, P. P. (2011). ICDAR 2011 robust reading competition - challenge 1: Reading text in born-digital images (web and email). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1485–1490.

Kulkarni, N. (2012). Color thresholding method for image segmentation of natural images. *International Journal of Image, Graphics and Signal Processing*, 4(1):28–34.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Intl. Conf. on Machine Learning (ICML)*, pages 282–289.

Lazebnik, S., Schmid, C., and Ponce, J. (2005). A sparse texture representation using local affine regions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1265–1278.

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

Lee, S. H., Cho, M. S., Jung, K., and Kim, J. H. (2010). Scene text extraction with edge constraint and text collinearity. In *Intl. Conf. on Pattern Recognition (ICPR)*.

Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML*, pages 4–15.

Li, Y. and Lu, H. (2012). Scene text detection via stroke width. In *ICPR*, pages 681–684.

Liu, C., Wang, C., and Dai, R. (2005). Text detection in images based on unsupervised classification of edge-based features. *International Conference on Document Analysis and Recognition (ICDAR)*, 0:610–614.

Liu, H. and Ding, X. (2005). Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 19–25.

Liu, X., Lu, K., and Wang, W. (2012). Effectively localize text in natural scene images. In *ICPR*, pages 1197–1200.

Liu, X. and Samarabandu, J. (2005). An edge-based text region extraction algorithm for indoor mobile robot navigation. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 2, pages 701–706.

Liu, Z. and Sarkar, S. (2008). Robust outdoor text detection using text intensity and shape features. In *Intl. Conf. on Pattern Recognition (ICPR)*, pages 1–4.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Intl. Conf. on Computer Vision (ICCV)*, pages 1150–1157.

Lucas, S. M. (2005). Text locating competition results. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 80–85.

Lucas, S. M., Panaretos, A., Sosa, L., Tang, A., Wong, S., and Young, R. (2003). Robust reading competitions. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Lucas, S. M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R., Ashida, K., Nagai, H., Okamoto, M., Yamamoto, H., Miyao, H., Zhu, J., Ou, W., Wolf, C., Jolion, J.-M., Todoran, L., Worring, M., and Lin, X. (2005). ICDAR 2003 robust reading competitions: entries, results, and future directions. 7(2-3):105–122.

Mancas-Thillou, C., Ferreira, S., Demeyer, J., Minetti, C., and Gosselin, B. (2007). A multifunctional reading assistant for the visually impaired. *EURASIP Journal on Image and Video Processing*.

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conf. (BMVC)*.

Merino, C., Lenc, K., and Mirmehdi, M. (2011). A head-mounted device for understanding text in natural scenes. In *Fourth International Workshop on Camera-Based Document Analysis and Recognition*, pages 29–41. Springer LCNS 7139.

Merino, C. and Mirmehdi, M. (2007). A framework towards real-time detection and tracking of text. *Second International Workshop on Camera-Based Document Analysis and Recognition (CBDAR)*, pages 10–17.

Mikolajczyk, K. and Schmid, C. (2001). Indexing based on scale invariant interest points. In *ICCV*, pages 525–531.

Minetto, R., Thome, N., Cord, M., Fabrizio, J., and Marcotegui, B. (2010). Snoopertext: A multiresolution system for text detection in complex visual scenes. In *Intl. Conf. on Image Processing (ICIP)*, pages 3861–3864.

Minetto, R., Thome, N., Cord, M., Stolfi, J., Precioso, F., Guyomard, J., and Leite, N. J. (2011). Text detection and recognition in urban scenes. In *Intl. Conf. on Computer Vision (ICCV)*, pages 227–234.

Ministerio de Fomento (2000). Norma 8.1-IC sobre Señalización Vertical de la Instrucción de Carreteras. *Boletín Oficial del Estado*, (25):4049–4106.

Ministerio de Presidencia (2003). Real decreto 1428/2003 sobre Reglamento General de Circulación. *Boletín Oficial del Estado*, (25):45684–45772.

Neumann, L. and Matas, J. (2010). A method for text localization and recognition in real-world images. In *Asian Conf. on Computer Vision (ACCV)*, pages 770–783.

Neumann, L. and Matas, J. (2012). Real-time scene text localization and recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Newell, A. J. and Griffin, L. D. (2011). Natural image character recognition using oriented basic image features. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 191–196.

Niblack, W. (1986). *An introduction to digital image processing.* Prentice-Hall.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66.

Oxford, U. (2007). British national corpus. [web page] `http://www.natcorp.ox.ac.uk/`.

Pal, U., Sharma, N., Wakabayashi, T., and Kimura, F. (2007). Off-line handwritten character recognition of devnagari script. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 496–500.

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

Parizi, S. N., Targhi, A. T., Aghazadeh, O., and Eklundh, J.-O. (2009). Reading street signs using a generic structured object detection and signature recognition approach. In Ranchordas, A. and Araújo, H., editors, *VISAPP 2009 - Proceedings of the Fourth International Conference on Computer Vision Theory and Applications, Lisboa, Portugal, February 5-8, 2009 - Volume 2*, pages 346–355. INSTICC Press.

Phan, T. Q., Shivakumara, P., and Tan, C. L. (2012). Text detection in natural scenes using gradient vector flow-guided symmetry. In *ICPR*, pages 3296–3299.

Plamondon, R. and Srihari, S. N. (2000). On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1).

Posner, I., Corke, P., and Newman, P. (2010). Using text-spotting to query the world. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010*.

Press release (2010). Word lens, an interactive real-time spanish/english translator app. [web page] http://www.bestappsite.com/2010/12/20/word-lens-an-interactive-real-time-spanishenglish-translator/.

Rabiner, L. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Reina, A. V., Sastre, R. J. L., Arroyo, S. L., and Jiménez, P. G. (2006). Adaptive traffic road sign panels text extraction. In *Proceedings of the 5th WSEAS International Conference on Signal Processing, Robotics and Automation*, ISPRA'06, pages 295–300.

Rodríguez, A. C., Kim, S., Kim, J. H., and Blanco Fernández, Y. (2009). English to Spanish Translation of Signboard Images from Mobile Phone Camera. In *IEEE SoutheastCon 2009*.

Sato, T., Kanade, T., Hughes, E. K., Smith, M. A., and Satoh, S. (1999). Video OCR: Indexing digital news libraries by recognition of superimposed captions. *Multimedia Syst.*, 7(5):385–395.

Shahab, A., Shafait, F., and Dengel, A. (2011). ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1491–1496.

Shao, Y., Wang, C., Xiao, B., Zhang, Y., Zhang, L., and Ma, L. (2010). Text detection in natural images based on character classification. In *Pacific-Rim Conference on Multimedia (PCM)*, pages 736–746.

Shi, X. and Xu, Y. (2005). A wearable translation robot. In *ICRA*, pages 4400–4405.

Shivakumara, P., Huang, W., and Tan, C. L. (2008). Efficient video text detection using edge features. In *Intl. Conf. on Pattern Recognition (ICPR)*, pages 1–4.

Sochman, J. and Matas, J. (2005). Waldboost - learning for time constrained sequential detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 150–156.

Tu, Z., Chen, X., Yuille, A. L., and Zhu, S. C. (2006). Image parsing: Unifying segmentation, detection, and recognition. In *Toward Category-Level Object Recognition*, pages 545–576.

van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1582–1596.

van de Weijer, J., Gevers, T., and Bagdanov, A. D. (2006). Boosting color saliency in image feature detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(1):150–156.

Varma, M. and Zisserman, A. (2002). Classifying images of materials: Achieving viewpoint and illumination independence. In *Eur. Conf. on Computer Vision (ECCV)*, pages 255–271.

Varma, M. and Zisserman, A. (2003). Texture classification: Are filter banks necessary? In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 691–698.

Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms, `http://www.vlfeat.org/`.

Wang, R., Jin, W., and Wu, L. (2004). A novel video caption detection approach using multi-frame integration. In *Intl. Conf. on Pattern Recognition (ICPR)*, pages 449–452.

Wang, X., Huang, L., and Liu, C. (2009). A new block partitioned text feature for text verification. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 366–370.

Weinman, J. J. and Learned-Miller, E. G. (2006). Improving recognition of novel input with similarity. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 308–315.

Wolf, C. and Jolion, J.-M. (2006). Object count/area graphs for the evaluation of object detection and segmentation algorithms. *Int. Jrnl. on Document Analysis and Recognition*, 8(4):280–296.

Wu, W., Chen, X., and Yang, J. (2005). Detection of text on road signs from video. *IEEE Transactions on Intelligent Transportation Systems*, 6(4):378–390.

Xu, C. and Prince, J. L. (1998). Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369.

Yamada, M., Budiarto, R., and Endo, M. (2004). Comic image decomposition for reading comics on cellular phones. *Computer*, E87D(6):1–8.

Yao, J.-L., Wang, Y.-Q., Weng, L.-B., and Yang, Y.-P. (2007). Locating text based on connected component and svm. In *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR '07. International Conference on*, volume 3, pages 1418 –1423.

Ye, Q., Jiao, J., Huang, J., and Yu, H. (2007). Text detection and restoration in natural scene images. *J. Visual Communication and Image Representation*, 18(6):504–513.

Yi, C. and Tian, Y. (2011). Text string detection from natural scenes by structure-based partition and grouping. *IEEE Transactions on Image Processing*, 20(9):2594–2605.

Yin, X., Yin, X.-C., Hao, H.-W., and Iqbal, K. (2012). Effective text localization in natural scene images with mser, geometry-based grouping and adaboost. In *ICPR*, pages 725–728.

Zhang, H., Berg, A. C., Maire, M., and Malik, J. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2126–2136.

Zhang, J., Chen, X., Yang, J., and Waibel, A. (2002). A PDA-based sign translator. In *Proc. the 4th IEEE Int. Conf. on Multimodal Interfaces, 2002*, pages 217–222.

Zhang, J. and Kasturi, R. (2010). Text detection using edge gradient and graph spectrum. In *Intl. Conf. on Pattern Recognition (ICPR)*, pages 3979–3982.

Zhang, Y. and Lai, J. (2012). Arbitrarily oriented text detection using geodesic distances between corners and skeletons. In *ICPR*, pages 1896–1899.

Zhao, X., Lin, K.-H., Fu, Y., Hu, Y., Liu, Y., and Huang, T. S. (2011). Text from corners: A novel approach to detect text and caption in videos. *IEEE Transactions on Image Processing*, 20(3):790–799.

Zhu, Y., Sun, J., and Naoi, S. (2012). Recognizing natural scene characters by convolutional neural network and bimodal image enhancement. In Iwamura, M. and Shafait, F., editors, *Camera-Based Document Analysis and Recognition*, volume 7139 of *Lecture Notes in Computer Science*, pages 69–82. Springer Berlin / Heidelberg.