CRANFIELD UNIVERSITY

LETICIA FERNANDEZ SANCHEZ

AUTOMATIC NUMBER PLATE RECOGNITION SYSTEM USING MACHINE LEARNING TECHNIQUES

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Computational and Software Techniques in Engineering:
Computer and Machine Vision

MSc
Academic Year: 2017–2018

Supervisor: Dr Zeeshan Rana
August 2018

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Computational and Software Techniques in Engineering: Computer and Machine Vision

MSc

Academic Year: 2017–2018

LETICIA FERNANDEZ SANCHEZ

Automatic Number Plate Recognition System Using Machine Learning Techniques

Supervisor: Dr Zeeshan Rana
August 2018

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

Abstract

Automatic Number Plate Recognition (ANPR) systems are widely used on a wide range of applications nowadays. The proposed approach has been developed in order to recognise UK number plates from high resolution digital images making use of the latest Computer Vision techniques and Machine Learning methods. For this purpose, a comparison among the different existing Computer Vision techniques used in ANPR is carried out and a deep insight on the operation and mode of use of the most commonly used Machine Learning algorithms in ANPR is provided, being these: Support Vector Machines, Artificial Neural Networks and K-Nearest Neighbours. Besides, for the development of an efficient, fast and reliable ANPR application, a huge car images dataset is created from scratch, necessary both for training the Machine Learning algorithms and for evaluating the performance of the developed system. The global results obtained, which are equal to or above 90% of success with a response time of less than 3 seconds, prove that the system is able to compete with other recently developed ANPR systems of similar characteristics.

Keywords

Automatic Number Plate Recognition; Machine Learning; Artificial Intelligence; Computer Vision; Image Processing; Support Vector Machine; K-Nearest Neighbour; Optical Character Recognition.

Contents

List of Figures

List of Tables

List of Abbreviations

ANPR	Automatic Number Plate Recognition
SVM	Support Vector Machine
QP	Quadratic Programming
ANN	Artificial Neural Network
MLP	Multi-Layer Perceptrons
KNN	K-Nearest Neighbour
OCR	Optical Character Recognition
DSIP	Digital Signal and Image Processing

Acknowledgements

First of all, I would like to thank my supervisor, Dr Zeeshan Rana, for his continuous support and useful guidance since the beginning of the project. Secondly, I feel also really grateful to Dr Gilbert Tang, who provided me with very good advice when I was at a critical stage of the project. Besides, I would like to give special thanks my classmates, Astrid and Miguel, and my flatmate, Moni, who have constituted my strongest support pillars not only during the development of this Thesis, but also during the whole academic year. And finally, I know that this would not have been possible without the help and love of all my family and my boyfriend, Bosco, so I would really like to thank them too.

Chapter 1

Introduction

This chapter introduces the topic of Automatic Number Plate Recognition (ANPR). Section 1.1 describes both the fundamentals of this technology and its applications. Section 1.2 provides a review of the different types of number plates existing within the European Union, making particular emphasis on UK number plates. Section 1.3 establishes the aim and the most important objectives of the project. And finally, Section 1.4 summarises the structure that this Thesis report follows.

1.1 Background and motivation

Automatic Number Plate Recognition is a computer vision technology that efficiently identifies vehicle number plates from images without the need for human intervention. In recent years, it has become more and more important due to three main factors: the growing number of cars on the roads, the rapid development of image processing techniques and the great quantity of real-life applications that this technology offers [1].

Some of the most typical applications of ANPR systems are traffic law enforcement, automatic toll collection or parking lot access control. But this technology is also widely used

for other, perhaps, more inspiring purposes like crimes resolution, as it helps to identify the cars of the offenders [2]. In Figure 1.1, some other examples of these applications can be appreciated.

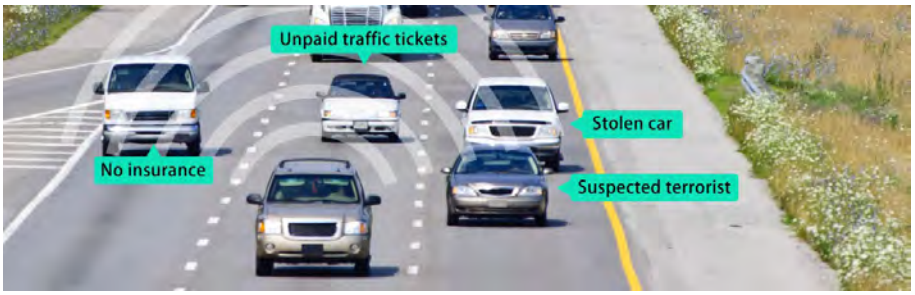

Figure 1.1: Examples of ANPR applications [3]

However, the development of ANPR systems is no easy task, since it faces numerous challenges due to environmental and number plate variations. As for the former, varying illumination or background patterns greatly affect number plate recognition. In effect, varying illumination can degrade the quality of the car image and background patterns add extra difficulty to the number plate location process. And, as for the latter, the location, quantity, size, font, colour or inclination of number plates constitute very challenging factors in the development of a consistent ANPR system, which is clearly illustrated in the following section [2].

1.2 Number plates in the European Union

A number plate or vehicle registration plate is a metal or plastic plate attached to a motor vehicle for its official identification. They contain a registration identifier, which is a numeric or alphanumeric code that uniquely identifies the vehicle within the database of the issuing authority [4].

The number plates of the different countries belonging to the European Union follow the common EU format, which was introduced in November 1998 and satisfies the re-

quirements of the Vienna Convention on Road Traffic. These requirements state that a distinguishing code for the country of registration of the vehicle must be displayed on the rear of the vehicle. This can be fulfilled either by directly incorporating this code into the vehicle number plate or by placing an oval sticker with the corresponding code next to it. This country code must be placed inside a blue section with the EU circle of stars, as it can be appreciated in Figure 1.2. Indeed, the common EU format also establishes that the number plates must be either white or yellow and that they must be wider than they are tall [5].

Figure 1.2: Common EU format [4]

Some non-EU European countries, like Norway, Iceland, Ukraine, Moldova or Turkey, have also implemented a similar number plate format to the common EU format, but replacing the circle of stars with their own symbol. Some examples of this are provided in Figure 1.3.

Figure 1.3: Common EU format adopted by some non-EU European countries [4]

As for the registration identifier itself, in order to avoid duplications that could lead to administrative problems, apart from using differing numbering schemes and text fonts, each country tried to develop a different alphanumeric system for their number plates. However, in practice, this was not completely achieved, as some countries share very similar registration identifiers. In the Figures 1.4 to 1.15, the number plates of the different EU member states together with the ones of some other European countries are classified

according to their similarity [4].

- Germany, Cyprus, Finland, Sweden, Hungary, Lithuania, Malta, Moldova and Belgium

Figure 1.4: Number plate scheme: AAA 111 [4]

- Slovakia, Austria, France, Italy, Serbia and Georgia

Figure 1.5: Number plate scheme: AA 111 AA [4]

- Denmark, Norway and Poland

Figure 1.6: Number plate scheme: AA 11111 [4]

- Greece and Spain

Figure 1.7: Number plate scheme: AAA 1111 and vice-versa [4]

- The Netherlands and Portugal

Figure 1.8: Number plate scheme: AA/11-AA-11 [4]

- Luxembourg and Latvia

Figure 1.9: Number plate scheme: AA 1111 [4]

- Romania and United Kingdom

Figure 1.10: Number plate scheme: AA 11 AAA [4]

- Montenegro and Slovenia

Figure 1.11: Number plate scheme: AA AA 111 [4]

- Ireland

Figure 1.12: Number plate scheme: 11-A-111111 [4]

- Bulgaria

Figure 1.13: Number plate scheme: AA 1111 AA [4]

- Estonia

Figure 1.14: Number plate scheme: 111 AAA [4]

- Czech Republic

Figure 1.15: Number plate scheme: 1 AA 1111 [4]

1.2.1 Number plates in the United Kingdom (UK)

The current vehicle registration format of the UK was introduced in 2001 and it consists of the following parts [6].

- **Two letters:** they refer to the region in the country where the vehicle was first registered. Letters I, Q and Z cannot appear in this group of characters.
- **Two numbers:** they indicate the date when it was issued.
- **Three letters:** they are randomly chosen among all the alphabet letters except for the letters I and Q.

As for the dimensions, UK number plates have a standard height of 112 mm and a standard width of 524 mm. The characters appearing on them measure 50 x 79 mm, except for character 1, which is much thinner than the rest, measuring 14 x 79 mm [6].

All the above described aspects can be appreciated in Figure 1.16.

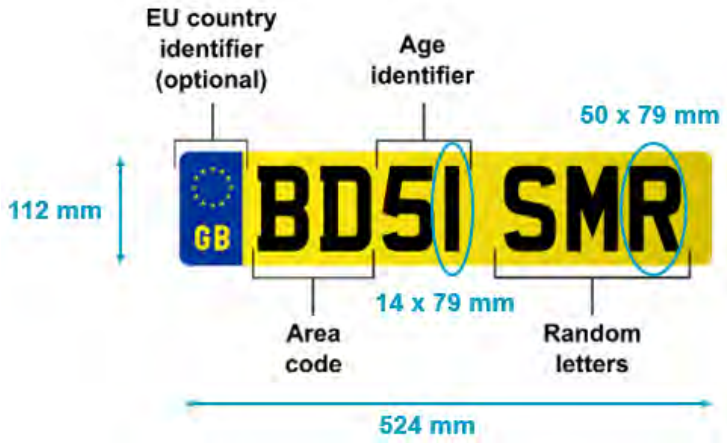

Figure 1.16: Number plate format of the UK [7]

1.2.1.1 Rules for UK number plates

All UK number plates must fulfill the following rules [6]:

1. All number plates should be made from a reflective material.
2. Front plate: it should display black characters on a white background.
3. Rear plate: it should display black characters on a yellow background.
4. Number plates should not have a background pattern.
5. Characters on a number plate need to be a certain height and size.
6. Characters on a number plate can be 3D.
7. Motorcycles and motor tricycles registered on or after 1 September 2001 must only display a number plate at the rear of the vehicle. If they were registered before this date, they can also display a number plate at the front but they do not need to.

8. Motorcycle and motor tricycle number plate should be on two lines.

9. When towing a trailer with a car, it must display the same number plate as the car.

1.2.1.2 Flags and national identifiers

The following flags, shown in Figures 1.17 to 1.20, with identifying letters can be displayed on the left-hand side of British number plates.

- **Union flag:**

Figure 1.17: Union flag [7]

Associated to the United Kingdom, the letters with which this flag can be accompanied are: GREAT BRITAIN, Great Britain, GB, UNITED KINGDOM, United Kingdom or UK.

- **Cross of St. George:**

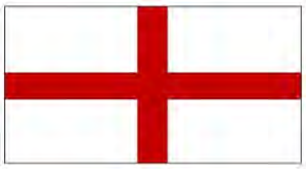

Figure 1.18: Cross of St. George [7]

Associated to England, this flag can be accompanied with the letters: ENGLAND, England, ENG or Eng.

- **Cross of St. Andrew:**

Figure 1.19: Cross of St. Andrew [7]

Associated to Scotland, the letters with which this flag can be accompanied are: SCOTLAND, Scotland, SCO or Sco.

- **Red Dragon of Wales:**

Figure 1.20: Red Dragon of Wales [7]

And finally, associated to Wales, this flag can be accompanied with the letters: WALES, Wales, CYMRU, Cymru, CYM or Cym.

All these flags must be displayed above the identifier and nor the flags neither the letters can be on the number plate margin. Indeed, they cannot be more than 50 mm wide.

When travelling in Europe, a GB sticker is needed if one of these national flags and identifiers is displayed [6].

1.2.1.3 Euro symbol

The Euro symbol must follow a series of rules:

1. It must be a minimum height of 98mm.
2. It must have a width between 40mm and 50mm.

3. It must have a reflective blue background with 12 reflecting yellow stars at the top.
4. It must show the member state (GB) in reflecting white or yellow.

If the Euro symbol is displayed on the number plate together with Great Britain (GB) national identifier, a separate GB sticker is not needed when travelling within the European Union [6].

1.2.1.4 Personalised number plates

In the UK, it exists the possibility of buying personalised number plates for the people who wishes to own a more personal or unique number plate. However, when personalising a number plate, the official rules for number plates should never be broken [6].

1.3 Aim and objectives

The aim of this project is to be able to develop a system for automatically recognising UK number plates from High Resolution (HR) digital images by combining advanced Computer Vision techniques with some of the most powerful Machine Learning algorithms.

Hence, the most important objectives within the project are:

- To analyse the different existing Computer Vision techniques used in ANPR in order to select the most efficient and appropriate ones. To learn to use these techniques and make the most of their capabilities.
- To analyse the different Machine Learning methods and choose the most suitable ones for ANPR. To learn how they work and to adapt them to the ANPR application.
- To develop an efficient, fast and reliable ANPR application that is able to compete with other recently developed ANPR systems of similar characteristics.

- Apart from the ANPR application itself, to develop another separate application that quantitatively analyses the performance of the ANPR system.
- To create a full car images dataset in order to be able to both train and test the Machine Learning algorithms and to evaluate the performance of the developed system.

1.4 Thesis structure

This Thesis is divided into six different chapters. Chapter 1 explains the motivation for the project, describes different types of number plates and establishes the objectives of the project. Chapter 2 summarises and reviews the different already existing Computer Vision techniques used in ANPR and provides a deep explanation of three most commonly used Machine Learning algorithms in ANPR. Chapter 3 describes in great detail the methodology carried out for developing the ANPR application. Chapter 4 analyses and discusses the experimental results obtained with the developed system. Chapter 5 summarises the most important conclusions extracted from the development of the project. And finally, Chapter 6 provides some ideas for improving or increasing the functionalities of the developed system that could be implemented in the future.

Chapter 2

Literature Review

This chapter summarises the Literature Review that has been carried out for the development of this project. Section 2.1 provides a review of the different existing Computer Vision techniques used in Automatic Number Plate Recognition, explaining the main stages of an ANPR system. And, apart from this, in Section 2.2 three Machine Learning algorithms that are lately being used in this kind of systems are thoroughly described, which are: Support Vector Machines (SVM), Artificial Neural Networks (ANN) and K-Nearest Neighbours (KNN).

2.1 Computer Vision techniques used in Automatic Number Plate Recognition

As mentioned in Chapter 1, the notably growing use of Automatic Number Plate Recognition systems has made it become a widely researched field in recent years. In effect, numerous and very different Computer Vision methodologies and techniques have been developed for this purpose. However, all the ANPR methodologies share four common stages that are explained in the next section together with the different approaches upon

which they can be based.

2.1.1 Stages of an ANPR system

Every Automatic Number Plate Recognition system is constituted of the following stages [2]:

1. Image acquisition
2. Number plate extraction
3. Number plate segmentation
4. Characters recognition

In the following lines, these stages are explained one by one along with a review of the different existing approaches for each of them.

2.1.1.1 Image acquisition

The initial stage is called Image Acquisition and it simply consists in capturing images or videos of different vehicles as they pass along the road. The distance at which the images should be taken so that the number plate can be read is approximately 3 meters [2].

This can be done in two different ways:

- **Using an infrared camera**

This first approach takes advantage of the retro-reflective nature of the surface of number plates. Unlike scatter or angle reflection, retro-reflection causes light to be reflected back to the source, as it can be observed in the Figure 2.1.

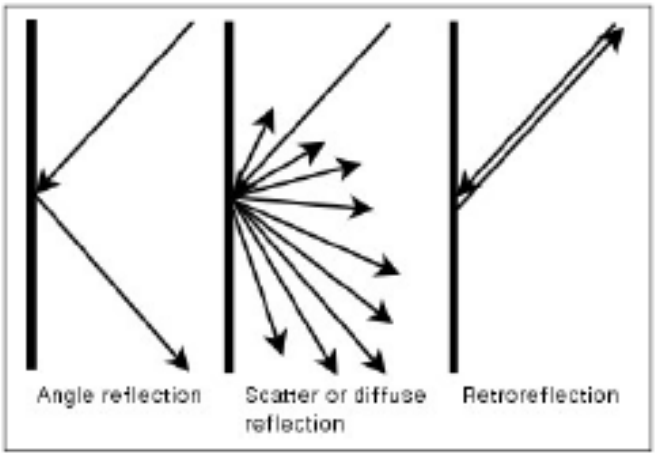

Figure 2.1: Types of reflections [8]

Hence, if a camera with a filter coupled with a structured infrared light projector is used, the infrared light only can be retrieved, thus obtaining an image where the number plate can be perfectly distinguished, like the one shown in Figure 2.2.

Figure 2.2: Car image obtained with an infrared camera [8]

The fact of obtaining an image like this greatly facilitates the development of the ANPR system, since the location and consequent extraction of the number plate from the image becomes a practically direct process. This way, the rest of the problem is reduced to the stages of number plate segmentation and character recognition [8].

- **Using a High Resolution (HR) Digital Camera**

Since infrared cameras are not available to everybody, the use of HR digital cameras for ANPR purposes is also widespread. The images obtained through this method are the common images regularly seen in daily life, which applied to a car results in a picture like the one shown in Figure 2.3.

Figure 2.3: Car image obtained with a HR digital camera [8]

For this purpose, a digital camera with a remarkably good resolution is needed. Apart from this, it is worth noting that the different configurations of the parameters of the camera as well as the lighting and weather conditions greatly affect the acquired image. Moreover, unlike the first one, this second approach presents the disadvantage of implying further image processing stages in order to be able to correctly locate and extract the number plate from the image [8].

2.1.1.2 Number plate extraction

Once the image of the vehicle is obtained, the next step is to extract the number plate from the it. For this purpose, this second stage is based on the identification of number plate features within the image, such as its colour, shape or characters, in order to detect its position and thus be able to extract it [2].

There is a wide range of different methods for number plate extraction, which, according to the number plate feature that they employ for extracting it, can be classified as follows:

- **Using edge features**

Since one of the most relevant characteristics of number plates is their rectangular shape, trying to identify the edges of this rectangle is a very common and efficient way of locating and extracting number plates.

In effect, these are the simplest ones of all the number plate extraction methods, being very fast and straightforward. However, in order for these methods to provide reliable results, the edges of the number plates must always be continuous and the images must not be very complex, that is, they must not contain too many unwanted edges that could be confused with number plates.

Some examples of these methods are: Sobel filter, Vertical Edge Detection Algorithm (VEDA), Generalised Symmetry Transform (GST) or Block-based method [2].

- **Using global image features**

Another common approach for number plate extraction is using global image features. This kind of methods aim at finding a connected object within the image whose dimension is similar to the one of a number plate.

They constitute very straightforward methods as well, which are completely independent of where the number plate is located in the image. Nonetheless, they are considered to be quite time-consuming and they can present problems with bad quality images, since these might generate broken objects that can difficult the location of the number plate.

Some of these methods would be: Connected Component Analysis (CCA), Contour detection algorithm or 2D cross-correlation with a pre-stored number plate template [2].

- **Using texture features**

Another typical characteristic of number plates is their frequent colour transition from the background colour, white or yellow, to the colour of the characters, generally black.

Hence, several methods have been developed in order to take advantage of these texture features.

Indeed, they constitute very robust techniques, being able to detect number plates even if their boundaries are deformed. However, they are computationally complex, especially when the image presents many edges.

Some examples of these methods are: Scan-line techniques, Sliding Concentric Window (SCW), Discrete Fourier Transform (DFT) or Gabor filters [2].

- **Using colour features**

This other type of methods is based on the fact that most of the number plates share the same colours, which are usually black and white or black and yellow. Thus, they focus on this feature, that is, they analyse the colours in the images, in order to detect number plates.

Thanks to this, it is possible to detect even inclined or deformed number plates. Nevertheless, these techniques depend a lot on the limitations of the colour scheme used. For example, RGB is greatly affected by the illumination conditions and HLS is very sensitive to noise. Besides, wrong detection can take place if the number plate colours are also present in other parts of the image.

Some of these methods would be: Colour edge detectors, Pixel classification based on the HLS or HSI colour models, Segmentation of colour images by the mean shift or Colour barycentre hexagon model [2].

- **Using character features**

As everybody knows, the actual core of all number plates is the registration number that they show, which is a combination of alphanumeric characters. Hence, methods that use

character features take advantage of this fact in order to locate and extract the number plate from the image.

In effect, these are quite robust techniques that are not affected by the rotation of the number plate. However, they are usually time-consuming and wrong detection can sometimes occur if additional characters are present in the image.

Some examples of these methods are: using algorithms for finding character-like regions in the image, doing a horizontal scan of the image or using the width of the characters [2].

- **Using combined features**

If all the previous methods already produce remarkably good results on their own, when appropriately combined, the performance of the system can be boosted. In other words, combining the use of two or more features offers more reliable results, but at the expense of an increase in the computational cost and complexity of the system.

Some examples of this kind of methods could be: combining colour and texture features, combining colour and edge features or even combining the three of them. In addition to these, the latest and probably the most successful combinations are the ones that mix some of the previously explained techniques with Machine Learning classifiers [2].

2.1.1.3 Number Plate Segmentation

Once the number plate is extracted from the image, the third stage of an ANPR system is number plate segmentation and it consists in extracting from the number plate image each of the characters that appear on it.

For this purpose, first, a pre-processing step is usually carried out on the image in order to improve its quality and thus facilitate the extraction of the characters. For example, a commonly tackled problem in this pre-processing step is tilt correction. In general,

previous quality enhancement of the extracted number plate image is a key aspect to achieve a successful number plate segmentation. Then, once the pre-processing step is completed, the proper number plate segmentation is carried out [2].

Just like in the previous stage, in order to achieve this, numerous methods have been developed depending on which features of the number plate image are used. Below, a brief description of the different existing approaches is provided.

- **Using pixel connectivity**

As it can be appreciated in Figure 2.4, pixel connectivity is the way pixels in an image relate to the pixels that they are surrounded by, that is, their neighbour pixels. This feature can be employed for number plate segmentation.

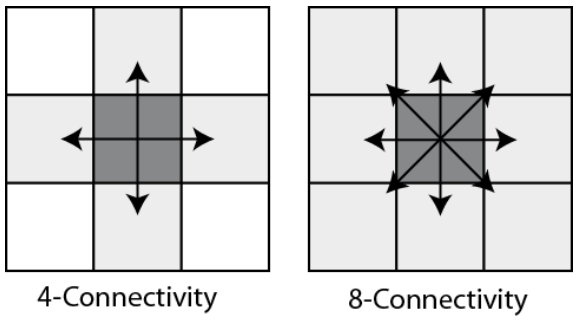

Figure 2.4: Examples of pixel connectivity [9]

This type of techniques are both simple and straightforward and, indeed, they result to be very robust to number plate rotation. Nevertheless, they usually fail to extract joined or broken characters, which can sometimes represent a problem.

An example of these techniques would be: Pixel labelling [2].

- **Using projection profiles**

Other methods for number plate segmentation are based on the extraction and analysis of the projection profiles of the number plate image. These represent the running sum of the

values of the pixels of an image in a certain direction, usually horizontal or vertical. Since all the number plates share the common structure of several black alphanumeric characters in a row on a white or yellow background, their projection profile has a characteristic shape that can be used to detect the position of the characters, as it can be observed Figure 2.5.

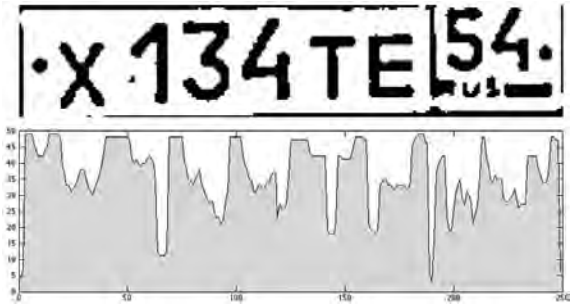

Figure 2.5: Operation of projection profiles methods [10]

This kind of methods present two main advantages. On the one hand, they are totally independent of the position of the characters and, on the other hand, they are able to deal with a certain degree of number plate rotation. However, they also present some disadvantages, like being greatly affected by noise and needing a prior knowledge of the exact number of characters appearing in the number plate, which usually varies from one country to another.

An example of these methods would be: Vertical projection of the binary extracted number plate [2].

- **Using prior knowledge of characters**

Methods that use prior knowledge of characters are based on the idea that if the positions and number of characters appearing in the number plate are previously known, it is not difficult to extract these characters by using a template.

In effect, they constitute very simple methods. Nonetheless, they are totally limited by

the prior knowledge of the number plate format and any change in it can affect the result of the number plate segmentation.

Some examples of these methods are: number plate scanning or number plate resizing into a known template size [2].

- **Using character contours**

Since the characters appearing on a number plate usually have a fixed size, contour modelling of these characters is another technique usually employed for number plate segmentation. What this kind of techniques usually do is to approximate the shape of the characters to a rectangle of fixed dimensions and, then, to look for the features in the number plate image that fit in this rectangle, hence finding the characters.

They are very simple, fast and straightforward techniques, which can even detect characters appearing in inclined number plates. Nevertheless, they can present problems with bad quality images, where some characters might not be very well distinguished, leading to a wrong detection of the contours.

Some examples of this kind of techniques are: Contours finding algorithms or Shape driven active contour model [2].

- **Using combined features**

Just like in the previous stage, combining several of the above described features offers more reliable segmentation results. But, once again, the price to pay is its computational complexity.

Some examples of this kind of methods could be: Adaptive morphology-based segmentation or Dynamic Programming (DP) [2].

2.1.1.4 Characters recognition

The final stage of every ANPR system is successfully recognising each of the previously extracted number plate characters. At this point, some new problems arise, such as different size and thickness of characters due to zoom factors, different character fonts for different countries, noisy or broken characters, etc [2].

In order to address all these issues, there are two main kinds of methods, which are based on the use of two different character image features.

- **Using raw data**

These methods, as their name indicates, use all the information in the extracted character image, that is, all the pixel values, and the operating principle that they use is template matching.

They are very simple and straightforward methods, but they can only correctly recognise single-font, non-rotated, non-broken and fixed-size characters. Moreover, the fact of processing all the pixels in the image, including non-important ones, makes these methods more time-consuming than the other ones.

Some examples of these methods are: Template Matching, Normalised cross-correlation or Lazy Machine Learning classifiers [2].

- **Using extracted features**

Unlike the previous one, this other type of techniques is based on the idea that not all the pixels in the character image are equally relevant for recognising the character.

Thus, by using this approach, the processing time is significantly reduced and image distortion affects less to the final result. Nonetheless, the process of feature extraction requires some additional time and non-robust features might degrade the recognition.

The main examples of these techniques are: Eager Machine Learning classifiers based on features vectors [2].

2.2 Machine Learning algorithms used in Automatic Number Plate Recognition

Nowadays, concepts like Machine Learning or Artificial Intelligence (AI) are literally booming everywhere, and this is not for no reason, because this kind of technology is actually delivering incredible advances in the field of computational sciences. In effect, Machine Learning algorithms offer innumerable applications, among which Automatic Number Plate Recognition can be included. In the following lines, a brief description of this kind of technology is provided.

As its definition states, learning is the activity of obtaining knowledge by studying something. In the particular case of Machine Learning, this learning activity is carried out by a computer, thus enabling the construction of computer programmes that automatically improve with experience.

There are three niches for Machine Learning [11]:

- **Data mining:** this kind of systems aim at using huge quantities of data which humans for themselves cannot handle in order to improve decisions. This offers, for example, a very useful application in the medical field, since it allows for the creation of medical knowledge based on medical records.
- **Software applications:** unbelievable as it may seem, not everything in this world can be programmed by humans. However, with the aid of Machine Learning techniques, these horizons can be broadened. For instance, this kind of techniques are currently being successfully applied in fields like autonomous driving, speech recognition, image recognition or, as in the case of this project, Automatic Number

Plate Recognition (ANPR).

- **Self-customising programmes:** although most people might not realise, almost everybody has contact with this last niche everyday. In effect, it is this kind of technology the one that is behind the news feeds that people usually receive according to their personal interests when surfing the Internet.

The way all this is achieved, that is, the way Machine Learning systems actually work is by means of using a series of labeled training examples based on which the algorithms are able to create general target functions that applied to a new, unseen dataset are able to correctly predict the expected result. These training labeled examples are called training dataset and the new, unseen dataset is called testing dataset. In particular, a sufficiently good and complete training dataset is crucial in this kind of applications, since a poor training always offers poor results [11].

Apart from the fundamentals that have just been explained, Machine Learning algorithms are so many and so varied that they have very little in common. In effect, even one single Machine Learning algorithm can be designed in infinite ways. Hence, for selecting the most appropriate design, a good evaluation is key, which is usually carried out by means of several statistics.

For all the previously described niches, three different types of classifications exist, each of them regarding a different aspect of Machine Learning algorithms.

- **According to the type of Machine Learning problem [11]:**
  - **Classification problem:** given a set of samples, each of them corresponding to a certain class or category of a known discrete set, this kind of problems aim at assigning the correct class label to each of the samples. To make it clearer, a direct application of this could be identifying different types of animals by analysing different pictures of them.

  - **Regression problem:** this second type tackles the problem of associating the appropriate numerical values or variables to the different input samples. Thanks to this, for instance, the distance to target based on shape features can be measured.
  - **Association or clustering problem:** the objective of association or clustering problems is to group a set of instances by attribute similarity. Thus, for example, they can be applied to image segmentation.

- **According to the modus operandi of the Machine Learning algorithm [11]:**

  - **Supervised:** this kind of Machine Learning algorithms are characterised by possessing knowledge of the output during the training phase, just like if they were learning with the presence of a teacher. In effect, the training data of Supervised Machine Learning algorithms is labelled with a class or certain value so that they are able to learn from them and thus they can later predict the corresponding class or label value when exposed to the testing set.
  - **Unsupervised:** unlike the previous type, Unsupervised Machine Learning algorithms do not have any knowledge of the output during the training phase, in other words, their training data is unlabelled. In this case, the goal of the algorithms is to determine unknown data patterns or groupings by means of a self-guided learning, which they achieve thanks to the performance of an internal self-evaluation against some criteria.

- **According to the kind of learning carried out by the algorithm [12]:**

  - **Eager Learning:** this first way of learning is based on generalisation, that is, learning from multiple examples and extracting a general hypothesis to approximate a target concept.
  - **Lazy Learning:** on the contrary, this other way of learning, also called Instance Based Learning, is characterised by the use of memorisation. In effect,

they do not extract any hypothesis from the learning examples, they just store them and then, they compare their similarity to the testing examples.

Finally, it is worth mentioning some of the most important Machine Learning algorithms, which are: Support Vector Machines (SVM), Artificial Neural Networks (ANN), K-Nearest Neighbours (KNN), Decision Trees, Random Forests, Genetic Algorithms or Bayesian Classifiers [11]. In the following sections, the first three of them are thoroughly explained, since they are the ones that are most commonly used in ANPR systems.

2.2.1 Support Vector Machines (SVM)

Support Vector Machines are a specific type of Linear Classifiers which belong to the group of Supervised Machine Learning algorithms and possess an Eager Learning approach. Given a set of points in the space in which each of them belongs to one of two different categories or classes, this kind of algorithms are able to predict whether a new, unknown point belongs to one class or another.

What SVM actually does in order to achieve this goal is to look for a hyperplane that separates the points of each class in an optimum way. This optimum separation is the one that assures the maximum distance or margin between the hyperplane and the nearest points to it. For this reason, these algorithms are also commonly known as 'maximum margin classifiers'. This way, the vector points located at one side of the hyperplane will belong to one category and the points located at the other side of the hyperplane will belong to the other category, which can be observed in Figure 2.6. The support vectors that these algorithms carry in their name are a sub-set of the training instances that define the decision boundary between classes [13].

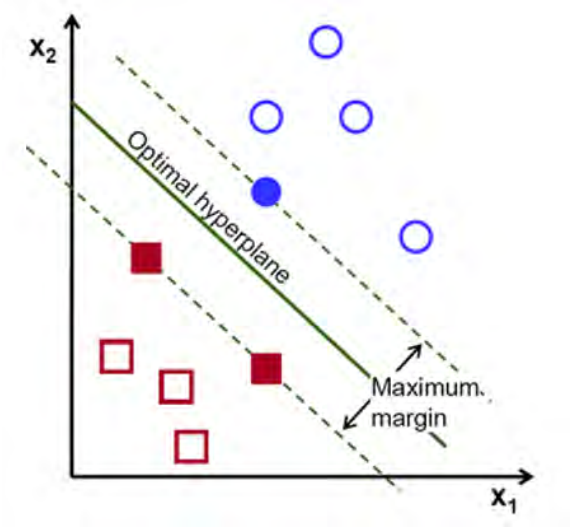

Figure 2.6: Operation of Support Vector Machines [14]

There are three principal reasons behind the use of the maximum margin in SVM. Firstly, that a small error in the boundary location provides least chance of causing misclassification. Secondly, that the model becomes immune to the removal of any non-support vector data points. And finally that, empirically, it works really well.

The way the optimal hyperplane is actually found is, of course, by means of several mathematical calculations. As it can be appreciated in Figure 2.7, this hyperplane is just a linear separator that follows the common equation of a line, $\vec{w}\vec{x}+b=0$, but in this case, considering w and x as vectors. Hence, to be able to define this equation and thus find the hyperplane, two unknowns must be found: $\vec{w}$ and b. The first one is the slope of the line and the second one is the y intercept.

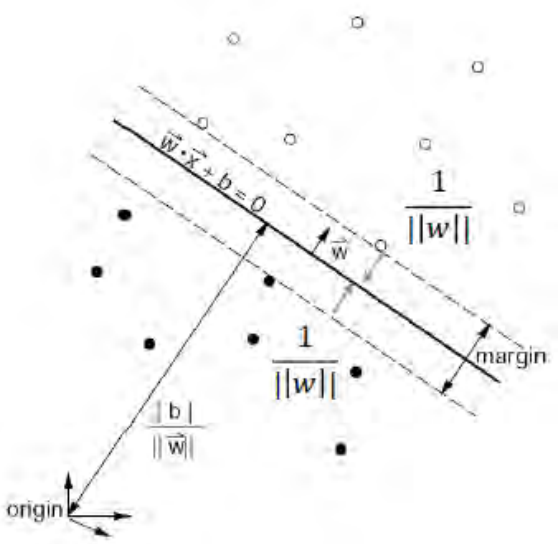

Figure 2.7: Optimal hyperplane calculation[14]

Thus, in order to find $\vec{w}$ and b, Equation 2.1 has to be maximised:

$$\arg\max_{wb} \frac{1}{||\vec{w}||} \tag{2.1}$$

Maximising the previous equation is equivalent to minimising Equation 2.2, which is a transform of Equation 2.1 made due to mathematic convenient.

$$\arg\min_{wb} \frac{||\vec{w}||^2}{2} \tag{2.2}$$

Therefore, the hyperplane is found by solving a constrained optimisation problem, which is done by means of Quadratic Programming (QP).

In case of having not linearly separable data and thus not being capable of finding the separating hyperplane, all the data should be projected to a higher dimension, for example, from 1D to 2D, where the data is separable [13].

To sum up, the operation of SVM algorithm is divided into two main steps [13]:

1. To find the optimum hyperplane with both the maximum margin and the minimum

number of training samples on the wrong side of the chosen boundary.

2. To solve the classification problem.

2.2.1.1 SVM concept extension: Multi-Class Learning

Support Vector Machines are, fundamentally, two-class classifiers and this constitutes a great limitation. For this reason, some methods for turning SVM into multi-class classifiers have been developed.

The most popular of these methods is called 'one-verses-the-rest' and it consists in constructing N separate Support Vector Machines, so that each of them learns to distinguish one of the classifications from all the remaining classifications. To make it more clear:

- **SVM 1 learns:** $Output = 1$ vs $Output \neq 1$
- **SVM 2 learns:** $Output = 2$ vs $Output \neq 2$
- ...
- **SVM N learns:** $Output = N$ vs $Output \neq N$

This way, the most positive classification of all of them is selected as the final result [13].

2.2.1.2 Advantages and disadvantages of Support Vector Machines

The main strengths of Support Vector Machines are that they appear to avoid overfitting and that they provide global optimisation, thus avoiding local minima, which represents a highly relevant aspect in Machine Learning and makes SVM stand among the best classifiers. Indeed, they offer a remarkably good performance on a wide variety of problems and, especially, on two-class problems [13].

However, these algorithms also present two main negative aspects. The first one is their high algorithmic complexity and extensive memory requirements in large-scale tasks due

to the use of Quadratic Programming. And the second one is that insight into kernel and kernel parameters choice is still needed [15].

2.2.1.3 Problems suited to Support Vector Machines

Support Vector Machines have proved to be among the best Machine Learning classifiers and, for these reason, they are widely used nowadays. Nevertheless, no classifier is the best one in absolutely every field, and SVM is not an exception to this. In effect, some classifiers are more suitable for solving some kinds of problems than other classifiers.

As for SVM, some of the cases in which this algorithm has proved to be suitable are the following ones [16]:

- When both the number of features and the quantity of training data are very large.
  - Examples: image classification or genes classification.
- When sparsity is very high, that is, when most of the features have zero value.
  - Examples: text classification or character recognition.

Use in Digital Signal and Image Processing (DSIP)

As it can be derived from the previously provided examples, Support Vector Machines can offer many different applications in Digital Signal and Image Processing. Indeed, many research studies have been carried out in this field using SVM, in which extremely good results have been obtained. Below, in Figure 2.8 and Figure 2.9, some graphic examples of this are presented:

- Satellite image classification with SVM:

Figure 2.8: Satellite image classification [17]

- Cell nuclei location with SVM:

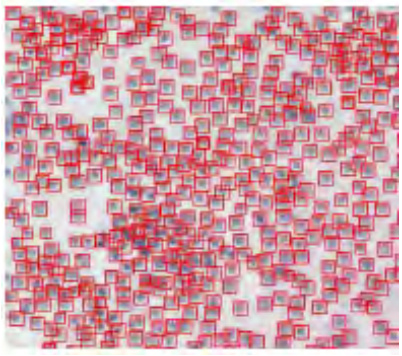

Figure 2.9: Cell nuclei location [14]

Apart from these two examples, SVM is also currently being widely used for many other DSIP applications, among which stands Automatic Number Plate Recognition. In this kind of applications, Support Vector Machines are mainly used for two purposes, which are number plate detection and character recognition. For both purposes, SVM offers remarkably good results. However, since SVM is a two-class classifier by nature, the application of the second purpose, which needs to distinguish among more than 20 characters, is computationally more expensive than the application of the first one.

2.2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) constitute another Supervised Machine Learning technique. They get their name from real neural networks, that is, the collection of neurons and synapses present in the human brain, shown in Figure 2.10. Altogether they are a powerful, adaptive and noise resilient pattern recognition system, combining both memorisation and generalisation. For this reason, they may be used for a wide variety of problems, which include classification, regression and association or clustering [18]. However, this section will mainly be focused on the classification problem, since it is the one applied to ANPR.

Figure 2.10: Representation of the biological neural network [19]

Their development took place around the 1950s and it was carried out by the scientists Warren McCulloch, Walter Pitts or Frank Rosenblatt, who tried to create a computing system inspired in the human brain, which resulted in the origin of Artificial Neural Networks [20]. To model artificial neurons, the basic nodes in the brain, they developed the concept of perceptron, and to model the connections between them, that is, the synapses, they made use of signals [21].

A perceptron is a quite simple mathematical function that provides an output value from an input one through a series of computations. As it can be appreciated in the general scheme shown in Figure 2.11, each of the components of an n-dimensional input vector are weighted and summed adding a bias value and then, the result enters an activation

function that provides the output of the perceptron, which is not a vector anymore but just a binary value, either '0' or '1'.

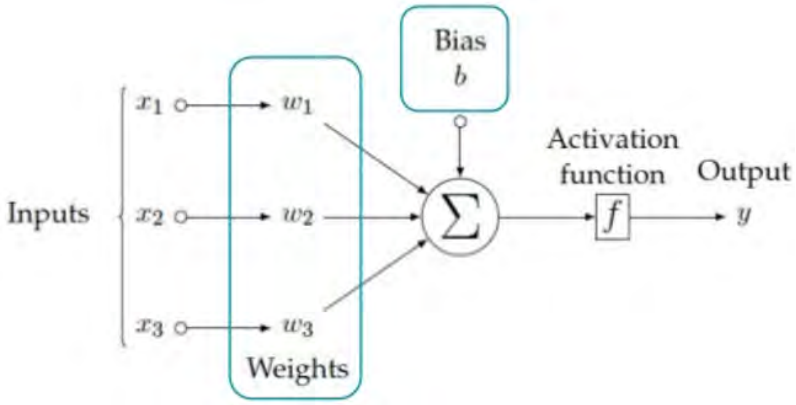

Figure 2.11: General scheme of a perceptron [23]

The blue rectangles in Figure 2.11 highlight the parameters that define the behaviour of the perceptron, which are the weights and the bias, so setting appropriate values for both of them is essential. The weights are real numbers that express the importance of each input to the output. As for the bias, it can be described as a measure of the easiness with which the perceptron outputs a '1'. The bigger the bias, the easier it is to get a '1' as output [20].

Nevertheless, the component that actually provides the output of the perceptron is the activation function. This is a predefined function, such as the sigmoid function, the rectifier function or the sign function, in charge of providing a smooth transition as input values change, so that a small change in the input produces a small change in the output [21].

To sum up, the general behaviour of a perceptron in mathematical terms can be appreciated in Equation 2.3.

Let ƒ be, for example, the Sign Function:

$$y = sign(\sum_{i=0}^{n} x_i w_i + b) \tag{2.3}$$

Although their real power comes from using them in connected combination, perceptrons are able to recognise simple patterns by themselves. For example, they can individually represent simple functions, like AND, OR, NAND, or NOR, that is, any classification function where the inputs are linearly separable. However, they cannot represent non-linear separable functions, such as XOR. In order to achieve this, layered networks of perceptrons must be used, which are known as Multi-Layer Perceptrons (MLP) and constitute a popular class of feedforward ANN [18].

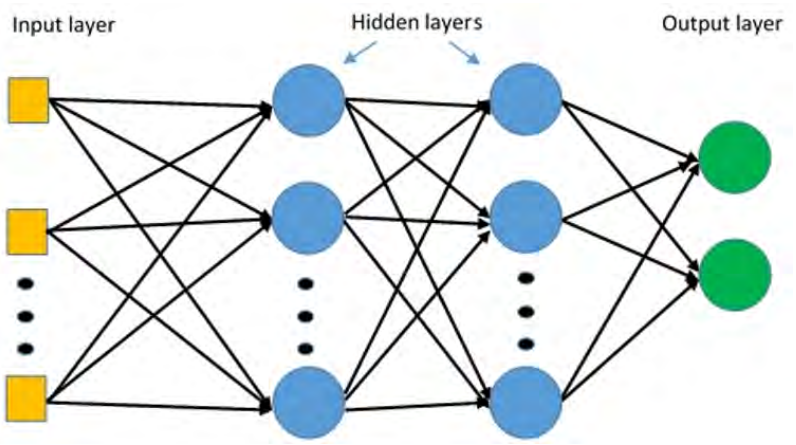

Figure 2.12: General scheme of Multi-Layer Perceptrons [21]

As it can be observed in Figure 2.12, a Multi-Layer Perceptron is composed of N layers of M perceptrons, each of them fully connected in graph sense to the perceptrons of the next layer. In other words, every node of layer N takes outputs of all M nodes of layer $N\text{-}1$. This way, the input to the network becomes a numerical attribute vector describing classification examples and the output of the network consists in a vector representing the classification results. An example of this output vector for classes A, B and C would be: $\{1, 0, 0\}$, $\{0, 1, 0\}$, $\{0, 0, 1\}$ [18].

Hence, when designing an MLP, two main aspects must be determined. These aspects are the number of layers in the network, which should be at least three, and the number of hidden nodes, which is usually determined through experimentation. Besides, as mentioned before, the weights of the perceptrons are a key factor for the correct performance

of the system. In the case of a Multi-Layer network, their value can be set if the activation function is differentiable. Apart from this, the activation function must also provide an output that is a non-linear combination of the inputs. A commonly used function that fulfills both conditions is the Sigmoid Function, which follows the expression shown in Equation 2.4.

$$\sigma = \frac{1}{1 - e^{-x}} \tag{2.4}$$

By using this kind of function, an MLP can be trained to learn non-linear functions by means of 'backpropagation', a method used for training Neural Networks by applying the gradient descent algortihm and which is the most suitable for classification problems. It is called like that because it makes modifications to reduce the error in the backwards direction, that is, from the output layer, through each hidden layer down to the first hidden layer. The main steps carried out by this method are the following ones [19]:

1. Initialisation of weights to small random values.
2. Forward propagation of the inputs.
3. Backward propagation of the error with the consequent update of the weights and biases.
4. Terminating condition when the error is minimised or enough iterations have been performed.

2.2.2.1 Classes of Artificial Neural Networks

Throughout the years, many different variants of Artificial Neural Networks have been developed. In summary, some of the most relevant classes of ANN are [23]:

- **Feedforward Neural Networks:** this constitutes the first developed and simplest class of ANN, which has been explained in this section. A subtype of this class

that is currently booming in Computer Vision applications is 'Convolutional Neural Networks'.

- **Regulatory Feedback Neural Networks:** this second class of ANN are characterised by the use of 'Negative feedback', which is called like that because it is not used to find optimal learning but to find the optimal activation of nodes.
- **Radial Basis Function Neural Networks:** these are two-layer Artificial Neural Networks that employ a distance criterion with respect to a center and which are widely used nowadays.
- **Recurrent Neural Networks:** this other class of ANN works on the principle of saving the output of a layer and feeding this back to the input to help in predicting the outcome of the layer.
- **Modular Neural Networks:** these neural networks have a collection of different networks working independently and contributing towards the output.
- **Dynamic Neural Networks:** this last class has proved that ANN do not have to be fixed, since the evolution of their topology during training has delivered numerous advances.

2.2.2.2 Advantages and disadvantages of Artificial Neural Networks

Many scientists state that Artificial Neural Networks possess the ability to outperform nearly every other Machine Learning technique. In effect, they constitute very robust and efficient algorithms, which excel to learn and model non-linear and complex patterns. Moreover, they do not impose restrictions of any kind on the input variables like, for example, the distribution of these. And on top of that, their parallel processing nature prevents small errors in located elements of the network from damaging the general performance of the network [24].

Nevertheless, this Machine Learning technique also presents some weak points. The

best-known disadvantage is their 'black box' nature, which means that it is really hard to understand how and why neural networks come up with a certain output. Apart from this, both the duration of development and the amount of data needed in a neural network are extremely excessive compared to the ones of simpler methods that provide a similar performance. And finally, as it can be expected derived from the previous drawbacks, they are computationally expensive [25].

2.2.2.3 Problems suited to Artificial Neural Networks

The most appropriate problems to be solved using Neural Networks are the ones that fulfill the following conditions [18]:

- They have high dimensional discrete or real-valued inputs, such as signal samples or image pixels.
- The outputs are discrete or real valued vectors of one or more values.
- The input data might be noisy.
- The shape of the target function is unknown.

Some of the applications that fulfill these conditions are: character recognition, image classification, speech recognition, natural language processing or even forecasting.

Use in Digital Signal and Image Processing

As it can be seen, most of the applications of Artificial Neural Networks are related to Digital Signal and Image Processing and, in fact, they are providing lots of advances in this field. In Figures 2.13 and 2.14, some graphic examples of this can be appreciated.

- Road sign identification with ANN:

Figure 2.13: Road sign identification [18]

- Road text recognition with ANN:

Figure 2.14: Road text recognition [18]

Apart from the ones that have just been shown, one of the currently most popular applications of Artificial Neural Networks is Automatic Number Plate Recognition. In particular, they are mainly used for the task of recognising the characters appearing on the number plates, which is providing very good results.

2.2.3 K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN) is a Supervised Machine Learning technique mostly used for classification but also used for regression. It belongs to the category of 'Lazy Learning', which is characterised by the use of memorisation for classifying data. What this kind of algorithms do is simply to store multiple classification examples and then, answer queries by comparing their similarity to the stored examples [12].

In KNN, it is assumed that every instance or classification example can be represented as a point in a Euclidean space and similarity is measured as the distance between these points. For classifying a certain point into one class or another, the vote from its K nearest neighbours is taken into account and the class with most of the votes is selected. In Figure 2.15, the different classification results that the green circle would obtain for different K values can be appreciated. For example, for $K = 1$, the classification result would be a red triangle; for $K = 3$, it would be a red triangle again; but for $K = 5$, it would turn into a blue square. For this reason, the choice of the value of K, which must always be a positive odd number, is one of the most crucial aspects for achieving a correct performance, being experimentation the only way of finding the correct one for each application [25].

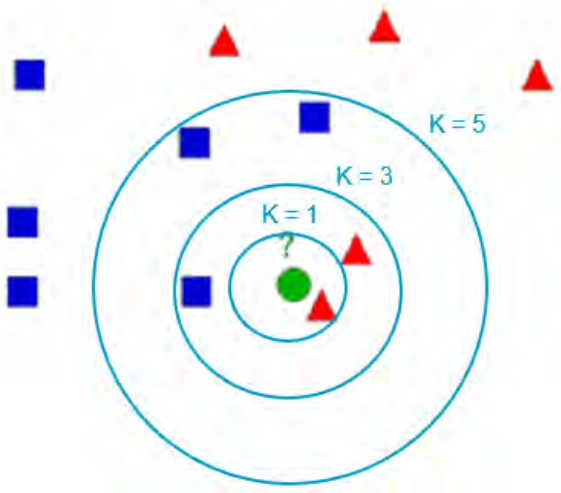

Figure 2.15: Examples of KNN classifications [26]

To describe this in a mathematical way, the training examples are vectors within a characteristic multidimensional space and each of them is described in terms of p attributes and considering q classes for classification. The values of the attributes of the i-th training example, where $1 \leqslant n \leqslant i$, are represented by the p-dimensional vector that can be observed in Equation 2.5 [27].

$$x_i = (x_{1i}, x_{2i}, ..., x_{pi}) \tag{2.5}$$

The space is partitioned in regions by locations and labels of the training examples. A point in space is assigned to class C if this is the most frequent class among the K nearest training examples. For this purpose, generally, Euclidean distance is employed, which is shown in Equation 2.6 [27].

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2} \tag{2.6}$$

To sum up, the operation of KNN is divided into two main phases or algorithms [26]:

- **Training algorithm:** the training phase consists in storing the vectors and the class labels of the different training examples. Mathematically, each example $< x, f(x) >$, is added to the structure that represents the training examples.
- **Classification algorithm:** the classification phase consists in analysing the distance between the vector representing the testing example and the different stored training vectors, selecting the K nearest neighbours and choosing the more frequent class within these.

2.2.3.1 Types of K-Nearest Neighbours

According to the value of K, the following two main types of nearest neighbours classifiers can be distinguished:

- ***K = 1***

This is the most intuitive type of nearest neighbours classifier. In effect, it aims at assigning a point x to the class of its closest nearest neighbour in the feature space. In order to know which one exactly this neighbour is, a decision surface is established, also called 'Voronoi diagram of instances'.

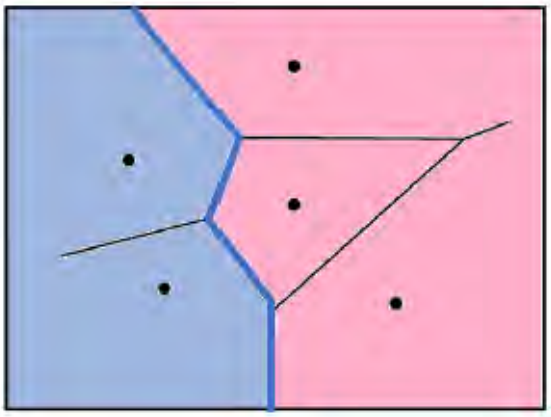

Figure 2.16: Voronoi diagram of instances [12]

As it can be observed in Figure 2.16, the convex polygon surrounding each training example indicates the region of instance space closest to that point. Hence, these are the areas within which each instance completely determines the classification [12].

As the size of the dataset approaches infinity, this type of KNN guarantees an error rate of no worse than twice the Bayes error rate, which is the minimum achievable error rate given the distribution of the data [27].

- ***K $\geqslant$ 1***

This other type of nearest neighbour classifier is very similar to the previous one, but taking more than one neighbour into account for making the classification, always considering odd numbers only. Besides, it is worth mentioning that this is done by taking the whole set of attributes into account, which can sometimes lead to the problem of biasing

the classification by many irrelevant attributes if, for example, only two attributes are the most relevant.

Therefore, in order to try to correct this, a new version of KNN was developed, called 'Distance Weighted KNN'. This version allows the application of a certain weight to the different attributes, thus making possible to give greater importance to the most relevant ones. Another similar version allows to set or adjust different weights with the known training examples. But, first, before assigning any weights, it is always highly recommended to identify and eliminate the non-relevant attributes [27].

2.2.3.2 Advantages and disadvantages of K-Nearest Neighbours

K-Nearest Neighbours is one of the most widely used Machine Learning algorithms in the industry, even more than SVM or ANN, and there are various reasons for this. In effect, its simplicity and easiness of implementation are two very important facts. Besides, it trains very fast and it does not lose any information in the process since it carries out memorisation instead of generalisation. And, what is more, it is able to learn really complex target functions, offering extremely competitive results.

However, a few disadvantages can also be found in this algorithm. As mentioned before, if the normal version of KNN is used, it can easily be fooled by irrelevant attributes. Apart from this, it can sometimes result a bit slow at query time. And, in addition, it can require plenty of storage space [12].

2.2.3.3 Problems suited to K-Nearest Neighbours

Finally, as it has been done with SVM and ANN, it is very important to know which kind of problems are suitable for being solved by means of KNN. In particular, this kind of problems should present the following characteristics [12]:

- There should be less than 20 attributes per instance.
- A huge quantity of training data should be required.
- They should be non-parametric problems.
- There should be no prior knowledge of the problem.

Some of the applications that fulfill all this are: classification, pattern recognition, prediction or text similarity measurement.

Use in Digital Signal and Image Processing

Once again, the described applications of this other Machine Learning algorithm are closely related to the field of Digital Signal and Image Processing. In order to see this in a more graphical way, some examples of possible DSIP applications are provided in Figure 2.17 and Figure 2.18.

- Map area classification with KNN:

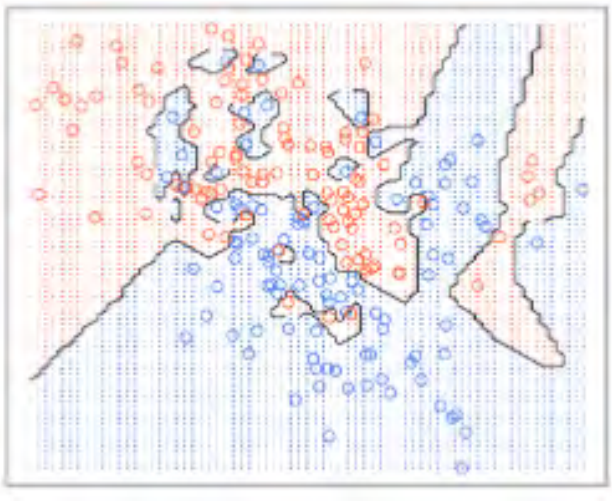

Figure 2.17: Map area classification [28]

- Cancer tumor detection with KNN:

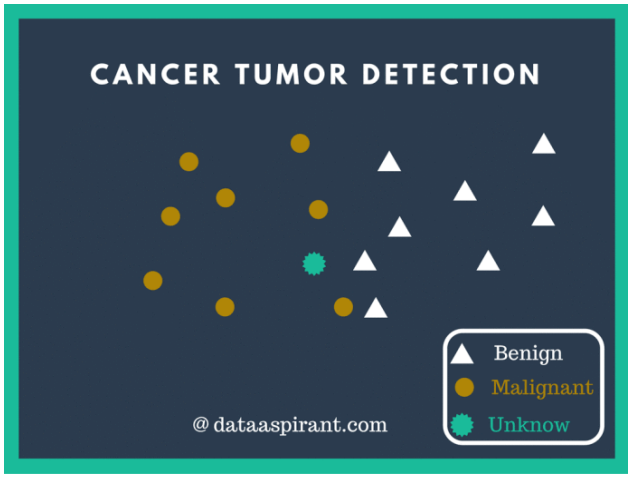

Figure 2.18: Cancer tumor detection [29]

In addition to these examples of applications, just like in the case of SVM and ANN, K-Nearest Neighbour is also commonly used in ANPR systems. The functionality that it offers in this kind of systems is exactly the same as the one provided by ANN, that is, characters recognition, and its results are stunning.

Chapter 3

Methodology

This chapter explains the Methodology that has been followed throughout the development of this project. Section 3.1 deeply explains how the Automatic Number Plate Recognition system has been developed, describing in high detail the implementation of each of the different stages of the system. And Section 3.2 explains how the developed ANPR system has been evaluated.

3.1 Development of the ANPR system

The Automatic Number Plate Recognition system designed in this project has been developed as a C++ application in the Integrated Development Environment (IDE) of Microsoft, Microsoft Visual Studio, which has been combined with the latest version (v3.4) of the OpenCV library in order to be able to make use of really powerful Computer Vision resources.

As explained in Chapter 2, ANPR systems consist of four main stages: image acquisition, number plate extraction, number plate segmentation and characters recognition. Nevertheless, in this project, in order to facilitate the development of the ANPR system,

a division of the second stage, number plate extraction, into three new stages has been made. These new stages are: image pre-processing, image segmentation and number plate detection. Therefore, the developed system is divided into six stages that are thoroughly explained one by one in the following sections. In Figure 3.1, a flowchart of this system can be appreciated.

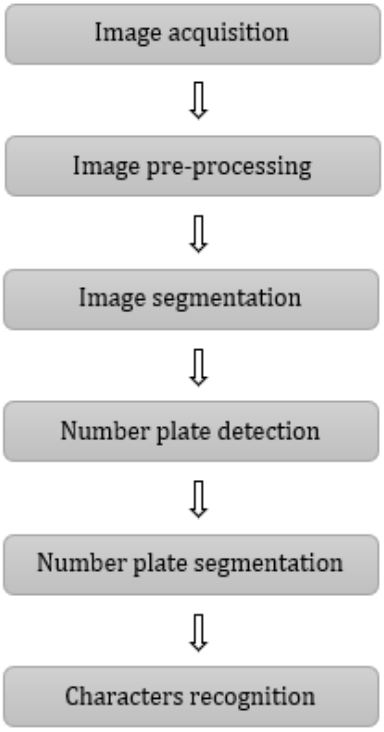

Figure 3.1: Flowchart of the developed ANPR system

Implementation

Since the application that implements this system is a quite large one that has more than 1000 lines of code and following the phylosophy of programming in distributed environments, when coding the application it has been divided into different classes, each of them carrying out some of the stages mentioned before. In Figure 3.2, the class diagram of the ANPR application and its relationships with the different stages is shown.

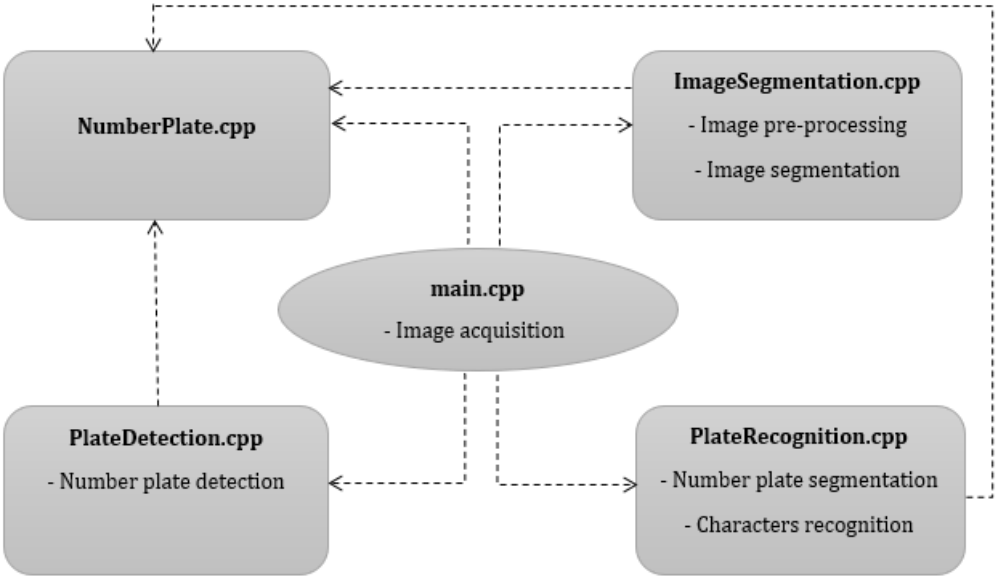

Figure 3.2: Class diagram of the ANPR application and relationships with the different stages

3.1.1 Image acquisition

Image acquisition is the first stage of every ANPR system and it consists in acquiring the image from which the number plate is going to be recognised. Thus, unlike the rest of the stages, it does not require almost any code to be typed, but it does require a good camera for taking high quality pictures of the cars. From the two available camera options, infrared camera and HR digital camera, in this project the second one has been selected.

In particular, the camera model that has been used is the Canon PowerShot SX50 HS, shown in Figure 3.3. It is professional camera that has the following main specifications: 12.1 Megapixels, a very powerful DIGIC 5 image processor with iSAPS technology and a 4x digital zoom.

Figure 3.3: Canon PowerShot SX50 HS

The images obtained with this camera have been taken at a distance of approximately 3 meters so that the number plate can be read from the image. As for their dimensions, an intermediate size of 800 x 600 pixels has been chosen.

Implementation

Once the image is obtained, the first step of the ANPR application is to load this image. For this purpose, when running the programme, the user must specify the desired image file in the command line, which is loaded into the application by means of the OpenCV command *imread()*. This function, which is called from inside the *main.cpp*, allows loading images in many different formats. In this case, the colour format is selected so that the original image can be shown to the user when he decides to run the programme. In Figure 3.4, an example of an input image is shown.

Figure 3.4: Example of input image

However, if no image is specified, the user is warned about it by showing a message in the command line and the programme finishes. This issue is detected by checking the number of input arguments in the command line, which need to be at least two. In a similar way, if the specified image file is corrupted or cannot be found, since an analysis of whether the loaded image is empty or not is carried out, the user is also warned about it.

3.1.2 Image pre-processing

The second stage is image pre-processing, which constitutes a crucial stage since it aims at adequately preparing the image so that the remaining stages can be successfully carried out. In the next paragraph, all these prior preparations of the image are explained.

In Image Processing, usually greyscale images are used due to their greater simplicity compared to colour images. Apart from this, in general, images usually present some undesired noise that needs to be removed. Besides, for this particular application, where number plates are the target, it is interesting to highlight within the image certain features that characterise number plates so that they can later be more easily detected. In this project, from all the different approaches that exist for this purpose, which have been explained Chapter 2, edge features have been chosen. This means that several image processing techniques need to be used in order to highlight the vertical lines appearing in the image and thus, facilitate the subsequent location of the number plate in the image.

Implementation

This stage is implemented in the class of the ANPR application *ImageSegmentation.cpp*, which is used inside the *main.cpp*.

According to the previous paragraph, the first thing to do at this stage is to convert the colour input image into a greyscale image. This is a totally straightforward step, since OpenCV provides a function, *cvtcolor()*, that carries out this kind of image type conversions. The result obtained using this command is shown in Figure 3.5.

Figure 3.5: Greyscale image

Secondly, the noise in the image needs to be removed, which is done by means of the OpenCV command *blur()*. This function blurs the image by using the 'normalised box filter', which makes that each pixel in the resulting image has a value equal to the average value of its neighbour pixels in the input image. This way, a smoothed version of the image is obtained [30], as it can be appreciated in Figure 3.6.

Figure 3.6: Blurred image

Then, in order to highlight the vertical lines or edges appearing in the image, which will contribute to the detection of the number plate, three different steps are carried out.

The first step consists in applying a 'Sobel filter' with the OpenCV function *sobel()*. This is done based on the principle that an edge is shown by the "jump" in intensity and that

this edge ”jump” can be seen more easily by taking the first derivative, as shown in Figure 3.7, which is calculated by means of this filter [31].

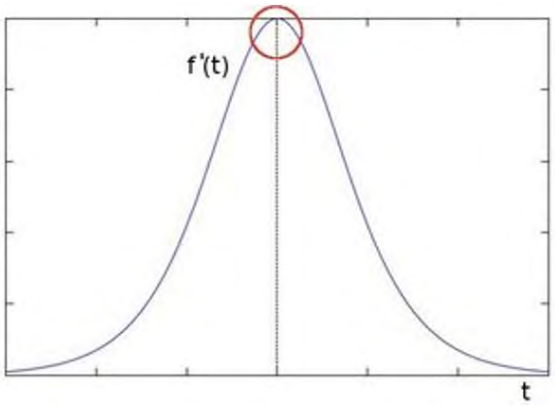

Figure 3.7: Representation of an image edge as a ”jump” in intensity and first derivative [31]

Therefore, by applying a ‘Sobel filter’, an image containing only information about the edges is obtained, as it can be seen in Figure 3.8.

Figure 3.8: Sobel filtered image

The second step is to threshold the image in order to obtain a binary image in which the previously obtained edges are much more highlighted than before. This is done by means of the OpenCV command *threshold()* and it is a very simple operation that assigns a value of 0 or 255 to each of the image pixels depending on whether their original value is above

or below a certain threshold. There are several ways of establishing this threshold value and in this application 'Otsu's method' has been chosen, which automatically calculates an optimal threshold value based upon the image histogram [32]. The image obtained by applying this image processing operation can be observed in Figure 3.9.

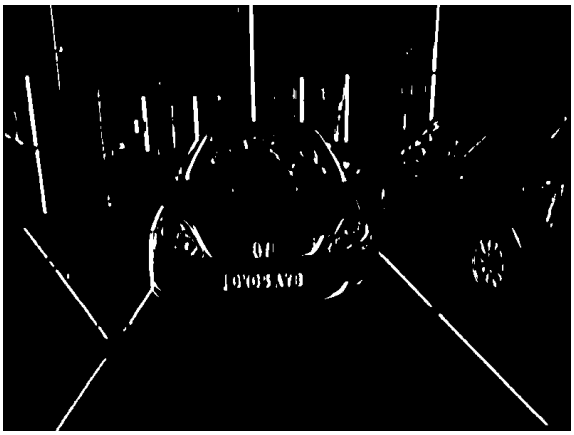

Figure 3.9: Thresholded image

Finally, the last step is to apply a closing morphological operation, which removes the small holes or dark regions present in the image and thus, helps to highlight even more the area of the image where the number plate is located. This is done by using the OpenCV function *morphologyEx()* and activating the option *CV_MOP_CLOSE* [33]. Figure 3.10 shows the result obtained after applying this closing operation. In it, it can be appreciated that the number plate has now become a white rectangle, which will be quite easy to be identified in the next stage: image segmentation.

Figure 3.10: Closed image

3.1.3 Image segmentation

Once the image is pre-processed, it is time to carry out image segmentation. This third stage consists in trying to locate and extract possible number plate patches within the image. It is a crucial stage because if it fails to locate and extract the real number plate, the whole ANPR system will fail. For this reason, it is very important to use the appropriate image processing techniques in order to create a robust image segmentation stage that prevents this from happening.

Since after the closing operation the number plate has adopted the shape of a white rectangle, as it has been shown in Figure 3.10, the clue for locating it is to look for rectangles in the image that adjust to the dimensions of a number plate. This number plate contour search provides the 'possible' location of the number plate within the image and thus, it enables its extraction from the input image.

Implementation

This stage is also implemented in the class of the ANPR application *ImageSegmentation.cpp*, which is used inside the *main.cpp*.

As mentioned in the previous paragraph, the first step of the image segmentation stage is

to carry out a contours search within the image that locates possible number plate regions. The library OpenCV provides a special command for doing this, which is *findContours()*. This function retrieves contours from binary images using an algorithm based on border following and it stores each of these contours as a vector of points [34].

Once the contours found are stored, a dimensions check of each of them is carried out in order to be able to remove the ones that do not fit into the dimensions of a number plate. For this purpose, the area and aspect ratio of UK number plates is taken into account, allowing a certain error. In Figure 3.11, the function developed for checking the dimensions of the possible number plates is shown. It is worth noting that for carrying out this check, the contours found need to be converted into an element called *RotatedRect*, which is done by means of the OpenCV command *minAreaRect()*.

```
bool ImageSegmentation::checkSize(RotatedRect candidate)
{
    //UK standard number plate dimensions: 524 x 112 mm
    //Set number plate aspect ratio
    const float aspect = 4.68;
    //Set a min and max number plate area
    int min = 15 * aspect * 15;
    int max = 125 * aspect * 125;

    //Compute the aspect ratio of the input rectangle
    float r = (float)candidate.size.width / (float)candidate.size.height;
    //Compute the area of the input rectangle
    int area = candidate.size.height * candidate.size.width;

    //Set an error margin
    float error = 0.4;
    //Apply the error margin to the aspect ratio
    float rmin = aspect - aspect * error;
    float rmax = aspect + aspect * error;

    //Check if the input rectangle fits the established parameters
    if (r < 1)
    {
        r = 1 / r;

        if ((area < min || area > max) || (r < rmin || r > rmax))
        {
            return false;
        }
        else
        {
            return true;
        }
    }
}
```

Figure 3.11: Function for checking the dimensions of the possible number plates

After filtering most of the erroneous number plates, the remaining possible number plate regions need to be cropped from the image. For doing this in an optimum way, 'Floodfill algorithm' is applied to each of these remaining regions thanks to the OpenCV command *floodFill()*. This algorithm, by taking three parameters (start node, end node and fill colour), is able to find all nodes in an array that are connected to the start node by a path of a target colour and to change these nodes to the fill colour. In effect, this is the algorithm behind the "bucket" fill tool provided by computer painting programmes like 'Paint' [35]. An example of the operation of this algorithm is shown in Figure 3.12.

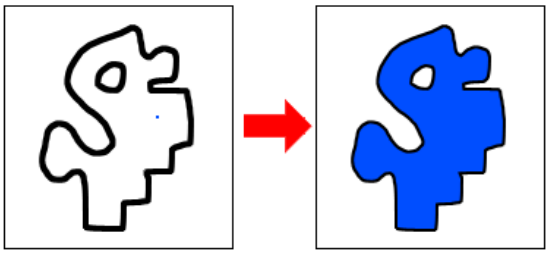

Figure 3.12: Operation of 'Floodfill algorithm' [36]

Once 'Floodfill algorithm' is applied, a new check of the dimensions of the obtained regions is carried out. The ones that fulfill this check, which in the car image example shown in Figure 3.13 are highlighted in red, are finally cropped from the input image by means of the OpenCV function *getRectsubPix()*, after having appropriately rotated the image with the OpenCV command *warpAffine()* in order to obtain a completely straight number plate image.

Figure 3.13: Final number plate regions found in the image

Finally, these possible number segments are resized to a fixed width and height by means of the OpenCV function *resize()* and stored in an object of the class *NumberPlate.cpp*, which saves two elements: the number plate image in the OpenCV format *Mat* and its position in the original image. Optionally, they can also be saved in *.jpg* format with the OpenCV command *imwrite()* for Machine Learning algorithms training purposes. Some examples of these images are shown in Figures 3.14 and 3.15.

Figure 3.14: Real number plate image

Figure 3.15: Non-number plate image

3.1.4 Number plate detection

The next stage of the developed ANPR system is number plate detection. In the previous stage, it has been possible to reduce the number of number plates candidates within the image to just a few and, now, the real one must be detected.

For this purpose, one of the Machine Learning algorithms described in Chapter 2 is employed. In particular, the one that can best perform this task is Support Vector Machine

since it provides stunning results at image classification, so this is the one selected in this project to distinguish real number plates from erroneous number plates.

It is important to remember that using a Machine Learning algorithm is not something straightforward, but it implies a prior training of the algorithm, for which a lot of training data is needed. Unfortunately, due to confidentiality issues, it does not exist any publicly available database of UK number plates. Hence, in order to be able to use SVM in the ANPR system, this training dataset has had to be created from scratch.

In effect, more than 500 car pictures have been taken around the campus of Cranfield University in order to be able to obtain a sufficient quantity of training data. In order to achieve this, they have all been processed exactly like in the image segmentation stage and the extracted patches, among which both real and erroneous number plates are found, have been saved as new images. This way, it has been possible to obtain the SVM training dataset shown in Table 3.1.

Table 3.1: SVM training dataset

Type of image	Number of images
Real number plate	500
Non-number plate	410

After using this dataset for training the SVM, the algorithm is ready for identifying real number plates.

Implementation

This stage is implemented in the class of the ANPR application *PlateDetection.cpp*, which is used inside the *main.cpp*.

Admittedly, the first step is to create the SVM itself. Doing this with the library OpenCV quite simplifies this task, since it already provides an SVM class that contains all the necessary functions for using this algorithm, being the function for creating it as intuitive as *create()*.

The second step consists in configuring the parameters of the SVM in an appropriate way. Some of the main SVM parameters are the type of SVM, the type of kernel and the termination criteria of the algorithm. As for the first one, by means of the OpenCV *setType()* function the type *C_SVC* has been chosen, which is the one most commonly used due to its ability to deal with non-linearly separable data and to classify more than two different classes. As for the second parameter, the OpenCV *LINEAR* kernel type has been selected with the function *setKernel()*, since it has been considered that there is no need to do any mapping to the training data. Finally, as for the termination criteria, two conditions have been established: to reach either 1000 iterations or a tolerance error of 0.01. This is done by means of the OpenCV command *setTermCriteria()* [37].

Once the SVM is correctly configured, it is time to train the algorithm using the OpenCV function *train()*. This function needs to receive both the training data and training classifications as parameters, after having loaded them into the programme as *.xml* files. Thus, before training the SVM, these files containing all the training data and training classifications need to be created. This has been done by means of creating a new, separate C++ application, called 'TrainSVM', in charge of reading all the real number plates and non-number plates image files, applying some necessary conversions to them and storing them in the mentioned *.xml* files. In Figure 3.16, the main core of this application is shown. When the SVM is trained, the SVM model is automatically built and it can be saved in another *.xml* file that can be directly loaded into the programme, without having to create the SVM from scratch every time the programme is run, in order to reduce the execution time of the ANPR application.

```
//Iterate over all the number plate images
for (int i = 1; i < numPlates + 1; i++)
{
    //Load image
    stringstream ss(stringstream::in | stringstream::out);
    ss << path_Plates << i << ".jpg";
    Mat img = imread(ss.str(), 0);
    //Apply necessary conversions to it to fulfill SVM requirements
    img = img.reshape(1, 1);
    //Save image and corresponding classification
    trainingImages.push_back(img);
    trainingLabels.push_back(1);
}

//Iterate over all the non-number plate images
for (int i = 1; i < numNoPlates + 1; i++)
{
    //Load image
    stringstream ss(stringstream::in | stringstream::out);
    ss << path_NoPlates << i << ".jpg";
    Mat img = imread(ss.str(), 0);
    //Apply necessary conversions to it to fulfill SVM requirements
    img = img.reshape(1, 1);
    //Save image and corresponding classification
    trainingImages.push_back(img);
    trainingLabels.push_back(0);
}

//Apply other necessary conversions to fulfill SVM requirements
Mat(trainingImages).copyTo(trainingData);
trainingData.convertTo(trainingData, CV_32FC1);
Mat(trainingLabels).copyTo(classes);

//Write saved data to XML file
FileStorage fs("SVM Training Data File.xml", FileStorage::WRITE);
fs << "TrainingData" << trainingData;
fs << "classes" << classes;
fs.release();
```

Figure 3.16: Code for creating *.xml* files from number plates and non-number plates images

Finally, once it is trained, the SVM can correctly identify the real number plate from among all the number plate candidates by means of applying the OpenCV method *predict()* to each of them. In Figure 3.17, a number plate identified by the SVM algorithm can be observed.

Figure 3.17: Number plate identified by SVM

3.1.5 Number plate segmentation

The next stage of the ANPR system is number plate segmentation, which consists in locating and extracting each of the characters appearing on the previously identified number plate.

The process carried out for achieving this is quite similar to the one carried out in the stage of image segmentation, since the same principle, segmentation, is applied in both stages. For doing so, in this case, the approach that has been used is doing a character contours search, which takes into account the dimensions of the characters of UK number plates.

Implementation

This stage is implemented in the class of the ANPR application *PlateRecognition.cpp*, which is used inside the *main.cpp*.

Following the philosophy of the second stage, image pre-processing, the first thing to do in order to be able to segment the number plate into its corresponding characters is to pre-process the image in order to facilitate the segmentation process. In this case, since the characters can already be clearly distinguished, it is not necessary to apply many image processing techniques. In effect, only applying a thresholding that binarises the number plate image is enough, which has already been explained in the stage of image pre-processing.

Then, as in the image segmentation stage, a contours search is carried out, checking that the dimensions of these contours fit the dimensions of the characters by means of analysing their aspect ratio and their height. It is worth noting that, in UK number plates, character ‘1’ is much thinner than the rest, which needs to be taken into account in the dimensions checking function, shown in Figure 3.18.

```
bool PlateRecognition::checkSize(Mat character)
{
    //British standard number plate characters dimensions: 50 x 79 mm
    //Set characters aspect ratio
    float theoreticalAspect = 50.0f / 79.0f;
    //British number plate character '1' dimensions: 14 x 79 mm
    //Set character '1' aspect ratio
    float theoreticalAspect1 = 14.0f / 79.0f;

    //Compute input character aspect ratio
    float charAspect = (float)character.cols / (float)character.rows;

    //Set error margin
    float error = 0.35;
    //Set min and max height values
    float minHeight = 15;
    float maxHeight = 30;

    //Apply the error margin to the aspect ratio taking into account special dimensions for character '1'
    float minAspect = theoreticalAspect1 - theoreticalAspect1*error;
    float maxAspect = theoreticalAspect + theoreticalAspect*error;

    //Check if the input character fits the established parameters
    if (charAspect > minAspect && charAspect < maxAspect && character.rows >= minHeight && character.rows < maxHeight)
        return true;
    else
        return false;
}
```

Figure 3.18: Function for checking the dimensions of the contours of the characters

Once the wrong contours are filtered, the remaining ones, an example of which can be appreciated in Figure 3.19, are cropped from the number plate image in the same way as in the image segmentation stage.

Figure 3.19: Final character contours found in the number plate image

Afterwards, both the cropped images of the characters and their corresponding position within the number plate are stored in an object of the subclass 'CharSegment'. Optionally, the images of the characters can also be saved in *.jpg* format for Machine Learning algorithms training purposes. Some examples of these images can be observed in Figure 3.20.

Figure 3.20: Character images extracted from the number plate image

3.1.6 Characters recognition

The final stage of the ANPR system is characters recognition, which, as its name indicates, consists in successfully identifying each of the previously segmented characters in order to be able to finally obtain the number plate registration number.

In the field of Image Processing, the process of recognising characters from digital images is commonly called 'Optical Character Recognition' (OCR) and many different techniques can be used for this purpose. In the developed ANPR system, a Machine Learning algorithm has been selected for doing so. This algorithm is K-Nearest Neighbours, which has been chosen due to its proved great performance at this particular task and its ease of use. Moreover, in particular, the approach of using the raw data of the character images for their recognition, described in Chapter 2, has been selected.

As explained in the stage of number plate detection, where SVM algorithm has been used, all Machine Learning algorithms need to experiment a training phase in order to be able to work. This implies the need for the creation of a new training dataset that can successfully train the KNN in order for it to distinguish all the different characters appearing in UK number plates. Hence, this training dataset must be composed of multiple images of all these characters, which are numbers 0 to 9 and all the alphabet letters except I and Q.

It is worth mentioning that, in this case, some publicly available databases of characters do exist. However, since the font of the characters of UK number plates is a singular one, better results can be obtained if the training dataset is created based upon the number plate images already available in the project.

For this purpose, all the real number plate images of the SVM training dataset have been processed exactly like in the number plate segmentation stage and then, the extracted character patches have been saved as new images. This way, it has been possible to obtain the KNN training dataset shown in Table 3.2.

Table 3.2: KNN training dataset

Character	Number of images
0	30
1	30
2	30
3	30
4	30
5	30
6	30
7	30
8	30
9	30
A	30
B	30
C	30
D	30
E	30
F	30
G	30
H	30
J	30
K	30
L	30
M	30
N	30
O	30
P	30
R	30
S	30
T	30
U	30
V	30
W	30
X	30
Y	30
Z	30

After using this dataset for training the KNN, the algorithm is ready for identifying the different characters.

Implementation

This stage is implemented in the class of the ANPR application *PlateRecognition.cpp*, which is used inside the *main.cpp*.

As it has been done with SVM, the first step is to create the KNN itself. Apart from the SVM class, the library OpenCV also provides a KNN class containing all the necessary functions for using this algorithm, being again the function for creating it as intuitive as *create()*.

In this case, no parameters need to be configured, which shows the extraordinary ease of use of KNN.

Therefore, the next step is to train the Machine Learning algorithm using the OpenCV command *train()*. Just as it happened with SVM, this function needs to receive both the training data and training classifications as parameters, after having loaded them into the programme as *.xml* files. Thus, before training the KNN, these files containing all the training data and training classifications need to be created. This has been done by means of creating a new, separate C++ application, called 'TrainOCR', in charge of reading all the character image files, applying some necessary conversions to them and storing them in the mentioned *.xml* files. In Figure 3.21, the main core of this application can be appreciated. When the KNN is trained, the KNN model is automatically built and it can be saved in another *.xml* file that can be directly loaded into the programme, without having to create the KNN from scratch every time the programme is run, in order to reduce the execution time of the ANPR application.

```
//Iterate over all the character images
for (int i = 0; i < numCharacters; i++)
{
    int numFiles = numFilesChars[i];
    for (int j = 1; j < numFiles + 1; j++)
    {
        //Load image
        cout << "Character " << strCharacters[i] << " file: " << j << endl;
        stringstream ss(stringstream::in | stringstream::out);
        ss << path << strCharacters[i] << "-" << j << ".jpg";
        img = imread(ss.str(), 0);

        //Apply necessary conversions to it to fulfill KNN requirements
        bitwise_not(img, imgWhite);
        imgWhite.convertTo(floatImgWhite, CV_32FC1);
        flattenImgWhite = floatImgWhite.reshape(1, 1);

        //Save image and corresponding classification
        allImgsWhite.push_back(flattenImgWhite);
        classifications.push_back(i);
    }
}

//Write saved data to XML file
FileStorage fs("KNN Training Data File.xml", FileStorage::WRITE);
fs << "TrainingData" << allImgsWhite;
fs << "Classifications" << classifications;
fs.release();
```

Figure 3.21: Code for creating *.xml* files from character images

Once it is trained, the KNN can correctly identify each of the different characters by means of applying the OpenCV method *findNearest()*, which takes the value of K as parameter. In this case, after carrying out some experimentation, it has been decided that the value $K = 1$ is the most appropriate one for this particular application.

After having successfully identified each of the characters, the resulting number plate registration number can finally be obtained.

For this purpose, first, the identified characters are ordered in the appropriate way, according to their position within the number plate. Due to the existence of some confusing characters, like B and 8, O and 0 or S and 5, once all the characters are correctly ordered a characters check based on the position of each of them is carried out in order to avoid this kind of confusions. This confusing characters check can be observed in Figure 3.22.

```
string correctConfusingChars(string numberPlate, int numDetectedChars)
{
    //Check condition of being a complete number plate, otherwise, confusing characters won't be corrected
    if (numDetectedChars == 7)
    {
        //Initialise variables
        char confusingChar0, confusingChar1, confusingChar2, confusingChar3,
        confusingChar4, confusingChar5, confusingChar6;

        confusingChar0 = numberPlate.at(0);
        confusingChar1 = numberPlate.at(1);
        confusingChar2 = numberPlate.at(2);
        confusingChar3 = numberPlate.at(3);
        confusingChar4 = numberPlate.at(4);
        confusingChar5 = numberPlate.at(5);
        confusingChar6 = numberPlate.at(6);
       //Correct confusing characters by checking the corresponding position
       //Numbers correction
       //Position 2
       if (confusingChar2 == 'O' || confusingChar2 == 'D')
       {
           numberPlate.at(2) = '0';
       }
       else if (confusingChar2 == 'B')
       {
           numberPlate.at(2) = '8';
       }
       else if (confusingChar2 == 'S')
       {
           numberPlate.at(2) = '5';
       }
       //Position 3
       if (confusingChar3 == 'O' || confusingChar3 == 'D')
       {
           numberPlate.at(3) = '0';
       }
       else if (confusingChar3 == 'B')
       {
           numberPlate.at(3) = '8';
       }
       else if (confusingChar3 == 'S')
       {
           numberPlate.at(3) = '5';
       }

       //Letters correction
       //Positions 0, 1, 4, 5 and 6
       //Correct B and 8
       if (confusingChar0 == '8')
       {
           numberPlate.at(0) = 'B';
       }
       if (confusingChar1 == '8')
       {
           numberPlate.at(1) = 'B';
       }
       if (confusingChar4 == '8')
       {
           numberPlate.at(4) = 'B';
       }
       if (confusingChar5 == '8')
       {
           numberPlate.at(5) = 'B';
       }
       if (confusingChar6 == '8')
       {
           numberPlate.at(6) = 'B';
       }
        //Correct S and 5 or 6
        if (confusingChar0 == '5' || confusingChar0 == '6')
        {
            numberPlate.at(0) = 'S';
        }
        if (confusingChar1 == '5' || confusingChar0 == '6')
        {
            numberPlate.at(1) = 'S';
        }
        if (confusingChar4 == '5' || confusingChar0 == '6')
        {
            numberPlate.at(4) = 'S';
        }
        if (confusingChar5 == '5' || confusingChar0 == '6')
        {
            numberPlate.at(5) = 'S';
        }
        if (confusingChar6 == '5' || confusingChar0 == '6')
        {
            numberPlate.at(6) = 'S';
        }
    }

    return numberPlate;
}
```

Figure 3.22: Function for checking and correcting confusing characters

Finally, once the number plate is checked, the result is shown to the user in two different ways: next to the number plate in the input image and printed in the command line. In Figures 3.23 and 3.24, these two ways of showing the obtained number plate registration number to the user are shown.

Figure 3.23: Obtained number plate registration number shown in the input image

```
PLATE:
Number of characters: 7
============================================
Recognised Number Plate: OW05AYB
============================================
```

Figure 3.24: Obtained number plate registration number printed in the command line

3.2 Evaluation of the ANPR system

When developing any kind of system, its evaluation is just as important as its development, since a wrong evaluation produces a wrong system. Indeed, due to the nature of the applications of ANPR systems, which are related to security, accuracy in this kind of systems is crucial. For these reasons, in this project, an exhaustive evaluation has been carried out.

In order to achieve this, the developed application has been tested with as many images as possible so that a highly realistic evaluation of its performance could be obtained. As mentioned in the previous section, for training both the SVM and the KNN algorithms, many pictures of cars had already been taken. However, when using Machine Learning, training and testing datasets must never be the same.

Therefore, a new car images dataset has been created only for software validation purposes. These pictures have been taken during several days with different weather conditions and at different hours within the day, which has allowed to have a wide variety of images that has enabled the evaluation of the robustness of the developed ANPR application. This testing dataset can be observed in Table 3.3 and some examples of the images contained in it are shown in Figures 3.25 and 3.26.

Table 3.3: Main testing dataset

Type of car image	Number of images
Front-view	150
Rear-view	150

Figure 3.25: Front-view car image

Figure 3.26: Rear-view car image

Apart from these images, some 'special' car pictures have also been taken, like cars with personalised number plates, inclined images of cars, foreign cars, images where no number plates are shown or images where multiple cars appear. Due to limited time and to the limited area where permission to take pictures had been awarded, not many of these images have been collected. Thus, they cannot constitute a proper dataset upon which reliable conclusions can be extracted. However, several indicative tests have been carried out based on them in order to obtain an idea of how the developed system would perform in those special situations. Table 3.4 shows this complementary testing dataset and Figures 3.27 to 3.31 provide some examples of the images contained in it.

Table 3.4: Complementary testing dataset

Type of car image	Number of images
Personalised number plate	10
Foreign number plate	12
Inclined image	20
Multiple number plates	15
No number plate	20

Figure 3.27: Personalised number plate car image

Figure 3.28: Foreign number plate car image

Figure 3.29: Inclined car image

Figure 3.30: Multiple number plates car image

Figure 3.31: No number plate car image

Implementation

The evaluation of the developed ANPR system has been implemented as a new, separate C++ application. It has mainly been based on the principal ANPR application, but applying some changes to the *main.cpp* and adding a new class called *PerformanceEvaluation.cpp*. In Figure 3.32, the class diagram of this new application can be observed.

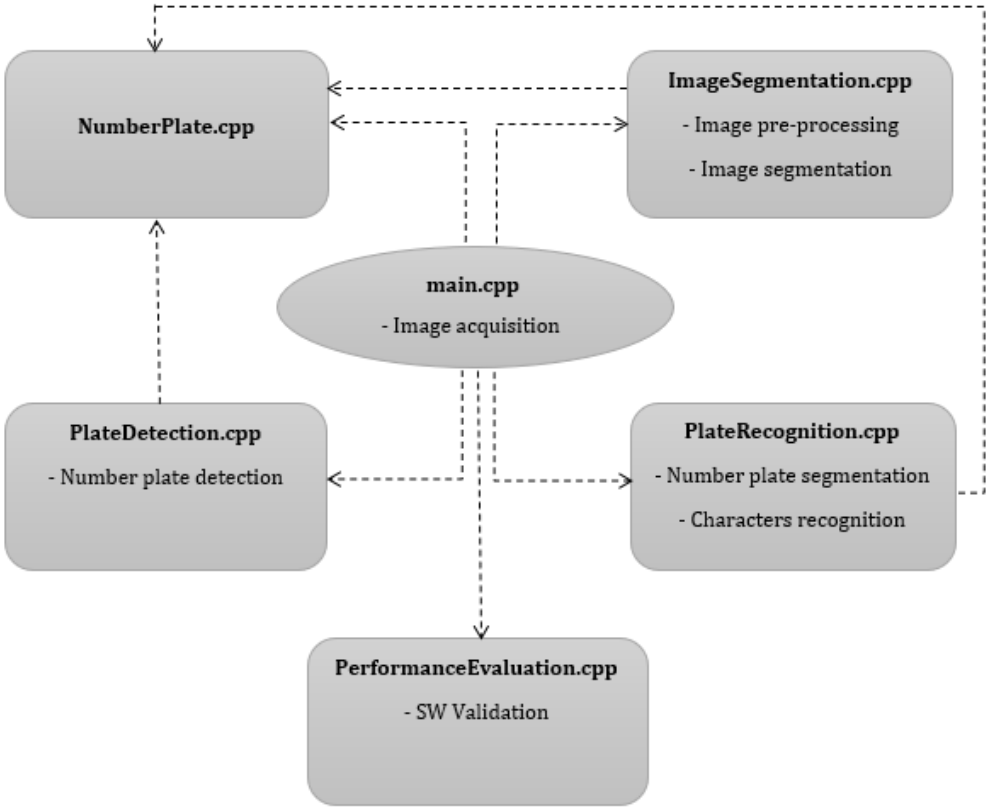

Figure 3.32: Class diagram of the software validation application and relationships with the different stages

As for the changes applied to the *main.cpp*, they have aimed at turning it into an iterative function, so that the all the images in the testing dataset could be tested on a row and thus, the average success rate could be obtained. This average success rate has been calculated in the class *PerformanceEvaluation.cpp* by means of comparing the obtained number plate registration numbers with the real ones, contained in the filename of each of the images, and computing the corresponding percentage.

In particular, three average success rates have been computed, each of them regarding a different aspect of the programme: Number Plate Detection (NPD), Optical Character Recognition (OCR) and the whole ANPR system. As for the first one, it evaluates the stages image segmentation and number plate detection, where the performance of SVM can be appreciated. The second one reflects the combined performance of the stages

number plate segmentation and characters recognition, where KNN algorithm is the main player. And finally, the last one shows the performance of the developed system as a whole, which is based on the combination of the previous two success rates.

In Figure 3.33 and 3.34, the two main functions that evaluate the performance of the developed ANPR system are shown.

```
int PerformanceEvaluation::checkPlate(string predictedPlate, string realPlate)
{
    int numErrors = 0;
    //Validate the number plate by checking its size and its resemblance to the real one
    if (predictedPlate.size() == realPlate.size())
    {
        //If it has been correctly recognised
        if (predictedPlate.compare(realPlate) == 0)
        {
            //Set the number of errors to 0
            numErrors = 0;
        }
        else
        {
            //If it has not been correctly recognised
            for (int i = 0; i < predictedPlate.size(); i++)
            {
                char aux1 = predictedPlate.at(i);
                char aux2 = realPlate.at(i);
                if (aux1 = !aux2)
                {
                    //Count the number of errors
                    numErrors++;
                }
            }
        }
    }
    else
    {
        //If the size does not fit, increase number of errors
        numErrors++;
        numErrors++;
    }
    return numErrors;
}
```

Figure 3.33: Function for checking the identified number plate

```
void PerformanceEvaluation::computeStatistics(int numHits, int totalNumErrors, int numImgs, int numPlatesNotDetected)
{
    //Compute ANPR success rate
    NPrecognitionSuccessRate = (100 * numHits) / numImgs;

    //Compute plate detection success rate
    NPdetectionSuccessRate = (100 * (numImgs - numPlatesNotDetected)) / numImgs;

    //Compute character recognition success rate
    float numChars = (numImgs - numPlatesNotDetected) * 7;
    float charsErrorRate = (100 * totalNumErrors) / numChars;
    charsSuccessRate = 100 - charsErrorRate;

    //Show results
    cout << endl <<endl << endl << "PERFORMANCE EVALUATION" << endl << endl;
    cout << "-------------------------------------------------------------------------" << endl;
    cout << "ANPR success rate: " << setprecision(2) << fixed << NPrecognitionSuccessRate << "%" << endl;
    cout << "-------------------------------------------------------------------------" << endl;
    cout << "Number plate detection success rate: " << setprecision(2) << fixed << NPdetectionSuccessRate << "%" << endl;
    cout << "Optical Character Recognition success rate: " << setprecision(2) << fixed << charsSuccessRate << "%" << endl;
    cout << "-------------------------------------------------------------------------" << endl;
}
```

Figure 3.34: Function for computing and printing the average success rates

Apart from the success rate, another very relevant aspect of ANPR systems is speed, since generally their ultimate goal is to be implemented in real-time. For this reason, a time counter has also been added to the software validation programme, particularly to the *main.cpp*, which has enabled the calculation and consequent evaluation of the time response of the developed system.

Chapter 4

Results and Discussion

This chapter describes and discusses the results that have been obtained with the developed ANPR system. Section 4.1 explains the organisation of the results. Section 4.2 provides and discusses the results obtained in the main experimentation. Section 4.3 provides and discusses the results obtained in the complementary experimentation. And finally, Section 4.4. discusses additional aspects regarding the results.

4.1 Results structure

As explained in the previous chapter, two different testing datasets have been created for evaluating the project. Hence, derived from them and taking into account that one of the datasets is more complete than the other one, the results obtained in this project have been classified into two different types: results obtained in the main experimentation and results obtained in the complementary experimentation.

On the one hand, the results obtained in the main experimentation are based on the main testing dataset. As it has been previously shown in Table 3.3, this dataset contains a total of 300 car images, being half of them front-view and the other half rear-view. This great

amount of images has enabled the collection of highly reliable quantitative results of the three available experiments, which are front number plate recognition, rear number plate recognition and global number plate recognition. These quantitative results correspond both to the success rates of Number Plate Detection (NPD), Optical Character Recognition (OCR) and ANPR and to the average execution time of the ANPR system, which have been explained in the previous chapter. In Table 4.1, a summary of the organisation of these results is provided.

Table 4.1: Summary of the results obtained in the main experimentation

Experiment	Type of analysis	Analysed aspects
Front number plates recognition	Quantitative	• OCR success rate • NPD success rate • ANPR success rate • Average execution time
Rear number plates recognition	Quantitative	• OCR success rate • NPD success rate • ANPR success rate • Average execution time
Global number plates recognition	Quantitative	• OCR success rate • NPD success rate • ANPR success rate • Average execution time

On the other hand, the results obtained in the complementary experimentation are based on the complementary testing dataset. This other dataset, which has been shown in Table 3.4, contains a total of 77 images, which correspond to five different types: images of cars with personalised number plates, images of foreign cars, inclined images of cars, images where multiple number plates appear and images where no number plates are shown. Due to the shortage of images of each type, it has not been possible to carry out a reliable quantitative analysis. Instead, a qualitative analysis of the performance of the developed ANPR system on the experiments that constitute each of the five previous types of images has been carried out. A summary of the organisation of these results can be observed in

Table 4.2.

Table 4.2: Summary of the results obtained in the complementary experimentation

Experiment	Type of analysis	Analysed aspects
Personalised number plates recognition	Qualitative	ANPR performance
Foreign number plates recognition	Qualitative	ANPR performance
Inclined number plates recognition	Qualitative	ANPR performance
Multiple number plates recognition	Qualitative	ANPR performance
No number plate detection	Qualitative	ANPR performance

4.2 Results obtained in the main experimentation

As it has been previously explained, in the main experimentation, three different experiments have been conducted upon which quantitative results have been obtained. These experiments are: front number plates recognition, rear number plates recognition and global number plates recognition. The following sections provide the results of these experiments together with their corresponding discussion.

4.2.1 Front number plates recognition

The results that have been obtained in this first experiment, front number plates recognition, regarding the three analysed success rates are shown in Table 4.3.

Table 4.3: Front number plates recognition success rates

Analysed aspect	Success rate
Number Plate Detection (NPD)	95%
Optical Character Recognition (OCR)	99.6%
Automatic Number Plate Recognition (ANPR)	94%

As it can be observed, the highest success rate of this experiment is the stunning 99.6% obtained for Optical Character Recognition, which gives an idea of the high competitiveness of the KNN classifier. As for Number Plate Detection, an also very high 95% of success has been obtained. The reason why this success rate is slightly lower than the previous one is not because of the performance of the SVM classifier, which actually works extremely well, but because of the complexity of the image segmentation process. Finally, the ANPR success rate, which is derived from the previous ones, has resulted in a remarkably good 94%.

The last analysed aspect, front number plates recognition average execution time, is shown in Table 4.4.

Table 4.4: Front number plates recognition average execution time

Average execution time	3.2*s*

This average execution time, which is as low as 3.2 seconds, proves the high speed of the developed ANPR system for recognising front number plates.

4.2.2 Rear number plates recognition

As for the second experiment, rear number plates recognition, the results that have been obtained regarding the three analysed success rates are shown in Table 4.5.

Table 4.5: Rear number plates recognition success rates

Analysed aspect	Success rate
Number Plate Detection (NPD)	89%
Optical Character Recognition (OCR)	98.7%
Automatic Number Plate Recognition (ANPR)	85%

In this case, a certain decrease in the analysed success rates compared to the ones of the first experiment can be appreciated. In effect, a 89% of success is obtained for Number Plate Detection, a 98.7% for Optical Character Recognition and an 85% for ANPR. This is mainly due to the general location of back number plates in cars, which are normally hidden below the boot lid. This causes a shadow to be permanently casted on the number plate, worsening the general performance of the ANPR system and, especially, of the image segmentation stage.

As for the average execution time of rear number plates recognition, it can be observed in Table 4.6.

Table 4.6: Rear number plates recognition average execution time

Average execution time	2.7*s*

This average execution time is very similar to the one obtained in the first experiment. In effect, it corresponds to 2.7 seconds, which proves that the developed ANPR system is also really fast at recognising rear number plates.

4.2.3 Global number plates recognition

The results that have been obtained in the last experiment, global number plates recognition, regarding the three analysed success rates are shown in Table 4.7.

Table 4.7: Global number plates recognition success rates

Analysed aspect	Success rate
Number Plate Detection (NPD)	92%
Optical Character Recognition (OCR)	99.2%
Automatic Number Plate Recognition (ANPR)	90%

Predictably, these success rates represent the average success rates of the previous two experiments, which are 92% for Number Plate Detection, 99.2% for Optical Character Recognition and 90% for ANPR. This indicates that the developed system is really competitive, since these success rates are within the range of the ones obtained in similar projects.

The last analysed aspect, global number plates recognition average execution time, is shown in Table 4.8.

Table 4.8: Global number plates recognition average execution time

Average execution time	2.9s

As it can be appreciated, this average execution time has resulted in 2.9 seconds, which is very fast and thus, it would enable the future implementation of the developed ANPR system as a real-time system. In effect, this average execution time is also within the range of the ones obtained in similar projects.

Although the global results obtained are remarkably good, the general performance of the system could probably be enhanced by changing the image segmentation approach, since this is the main stage that worsens a little bit the performance of the ANPR system. For this reason, it could be substituted by a more powerful image processing technique like, for example, RANSAC. However, this would be at the expense of a higher computational cost, which would produce a slower system response. Otherwise, the remaining solution

would be to try to reduce the influence of the image segmentation stage on the ANPR system, which would only be possible by means of changing the image acquisition approach to using an infrared camera instead of a HR digital camera.

4.3 Results obtained in the complementary experimentation

As it has been previously explained, in the complementary experimentation, five different experiments have been conducted upon which qualitative results have been extracted. These experiments are: personalised number plates recognition, foreign number plates recognition, inclined number plates recognition, multiple number plates recognition and no number plate detection. The following sections provide the results of these experiments described in a qualitative way.

4.3.1 Personalised number plates recognition

The first experiment of the complementary experimentation, personalised number plates recognition, has shown quite good results.

In effect, in most of the cases where the number plate is correctly extracted from the image, which has proved to be the greatest challenge of the developed ANPR system, the personalised number plates are successfully recognised. This makes sense since the font and size of the characters of personalised number plates are exactly the same as in normal number plates and the approach used in the developed system for extracting the different characters from the number plates focuses on these two aspects.

The only thing that changes in personalised number plates is the number of characters. This affects the confusing characters check of the developed system, which has been de-

signed based on the positions of the different characters in order to know if they constitute letters or numbers and thus, be able to correct them when necessary. Therefore, this reduces a little bit the success rate of personalised number plates recognition compared to normal number plates recognition. However, this decrease is not too heavy thanks to the excellent performance that KNN has shown, which minimises the number of character confusions.

There would be two possible ways of solving this issue. One of them would be to change the approach of the confusing characters check, which would completely solve the problem. And the other way would be to increase the KNN training data as much as possible in order to enhance even more the performance of KNN classifier. Nevertheless, this could not totally solve the issue and, indeed, extending the training dataset would would take a really long time.

4.3.2 Foreign number plates recognition

In the second experiment, foreign number plate recognition, a sharp decrease in the success rate takes place.

As it has been shown in Chapter 1, number plates vary a lot from one country to another. In effect, there are many different general schemes of number plates and also many different sizes and fonts of characters. For this reason, due to the way in which the ANPR system has been developed, it is very difficult for it to be able to correctly recognise a foreign number plate, which has been proved in the several tests carried out. These tests have shown that in most cases some of the characters are correctly identified, but in very few occasions is the full number plate successfully recognised.

A possible way of solving this would be to increase the training dataset in order to include numerous foreign car images so that they could be better recognised.

4.3.3 Inclined number plates recognition

As for the third experiment, inclined number plate recognition, intermediate results have been obtained.

The issue with this experiment is that the success rate totally depends on the angle of inclination of the number plate. If it presents just a low inclination, the number plate, usually, is successfully recognised. However, if it presents a higher inclination, the performance of the developed ANPR system is abruptly worsened.

In real life, road traffic cameras are generally situated on top of the road, from where straight car images are normally obtained unless one car is passing another one. Another common position of road cameras is right next to the road, where slightly inclined car images are obtained. Therefore, the fact that the developed ANPR system does not perform really well with very inclined images would not be very relevant from the point of view of real-life applications.

4.3.4 Multiple number plates recognition

As for the experiment of multiple number plates recognition, the results obtained have been quite positive.

In effect, the ANPR system has been designed to look for as many number plates as it can find in the input image and then, to iterate over them in order to recognise all of them. Hence, it normally succeeds in recognising multiple number plates in a single image. Nevertheless, as is normal, it sometimes fails when one of the cars is so far away that it cannot distinguish the different characters appearing on the number plate or even the proper number plate.

4.3.5 No number plate detection

Finally, the last conducted experiment, no number plate detection, has provided perfect results.

Although this was probably the easiest experiment, in all the empty images that have been tested, where no car appeared in any of them, it has not identified any erroneous number plate. Hence, this proves the reliability of the developed ANPR system regarding this aspect.

4.4 Additional discussion

Apart from the discussion of the results that has already been provided, there is an additional aspect that is also worth commenting. This aspect is the comparison of the performance at characters recognition between KNN and ANN.

At the beginning of the project, in order to know which of the two previously mentioned Machine Learning classifiers to use for the characters recognition stage, both approaches were coded and tested. In these tests, KNN showed that, without having to worry too much about choosing the right features of the characters in order for it to learn them, the results obtained were excellent. However, with ANN the opposite happened, being very complicated to find the correct features and thus, obtaining very poor results. For this reason, KNN was finally selected, which has undoubtedly constituted a totally right decision that the obtained results have been able to prove.

However, this does not mean that ANN performs badly at characters recognition, since actually it can perform really well, it just means that its implementation is more complex than the one of KNN. This complexity leads to a longer implementation time and, probably, to a higher computational cost. Therefore, since easier and computationally cheaper alternatives to ANN exist, like KNN in this case, which offer very similar results,

ANN is not the most commonly chosen algorithm when developing a Machine Learning application.

Chapter 5

Conclusion

This chapter summarises the main conclusions extracted from the development of the project, which are presented in the following lines.

In this project, an Automatic Number Plate Recognition (ANPR) system that makes use of two Machine Learning techniques, SVM and KNN, has been developed. This system receives a car image, processes it and analyses it by means of several Computer Vision techniques and the two mentioned Machine Learning algorithms and, finally, it identifies the number plate of the car appearing in the image.

This system has achieved its objective thanks to the good design and implementation of each of the different stages that it consists of. These stages are image acquisition, image pre-processing, image segmentation, number plate detection, number plate segmentation and characters recognition.

The main experimentation carried out has provided a global success rate of 90% with an average response time of 2.9 seconds, which is a clear indicator of the high competitiveness of the developed ANPR system. Achieving this would have been extremely difficult without the use of the Machine Learning classifiers Support Vector Machine and K-Nearest Neighbours, which have performed really well at their corresponding tasks,

number plate detection and characters recognition, respectively. In particular, it is worth highlighting the excellent performance of the developed Optical Character Recognition (OCR) system, in which KNN takes part, that has achieved a stunning 99% of success.

As for the complementary experiments that have also been carried out, although some aspects of the ANPR system would still need to be refined for achieving a sufficiently good performance on them, they have also shown generally positive results that give an idea of the robustness of the developed system.

Nevertheless, some limitations have also been found in the developed ANPR system. The main one is the relatively weak performance of the image segmentation stage. This issue could be solved by substituting the current image segmentation approach with RANSAC or other more powerful approaches. However, this would increase the computational cost and, unless a parallel processing approach was used, the speed of the system would be reduced. Therefore, it might be due to this issue that professional systems always use infrared cameras in the image acquisition stage, which eliminate the complexity of the image segmentation stage and, thus, the whole problem.

Chapter 6

Further Research

This last chapter describes the future work that could be done for improving the capabilities of the developed system, which is explained in the following lines.

Although the developed ANPR system has shown a remarkably good performance, there are still some aspects that could be improved if the necessary time was devoted.

The first one would be the enhancement of the performance of the image segmentation stage. As it has been previously explained, this could be done by means of using either a more powerful image segmentation approach or an infrared camera instead of a HR digital camera in the image acquisition stage. By using an infrared camera, two birds could be killed with one stone since this would also enhance the performance of the ANPR system in bad environmental conditions, like rain or mist.

The second ones are related to the limitations observed in the complementary experimentations. In particular, one would be the improvement of the method developed for confusing characters recognition so that the performance of personalised number plates recognition could be enhanced. And the other one would be greatly increasing the training datasets in order to also be able to recognise foreign number plates.

Moreover, additional functionalities could be added to the developed ANPR system. For

example, video could be implemented as input, instead of static images, in order to make the system more realistic or, also, vehicle manufacturer and model recognition could be implemented.

In the future, apart from the above mentioned challenges, research in the field of ANPR systems should also focus on the creation of a uniform evaluation method, based on a common testing dataset, in order to be able to equally analyse the performance of the different ANPR systems developed.

References

[1] Hogne Jorgensen. *Automatic License Plate Recognition using Deep Learning Techniques*. Norwegian University of Science and Technology, 2017.

[2] Shan Du, Mahmoud Ibrahim, Mohamed Shehata and Wael Badawy. *Automatic License Plate Recognition (ALPR): a State-Of-The-Art Review*. 2013.

[3] RoadMetric. *RoadMetric Integrates its ANPR with New Applications*. Online, accessed 07-July-2018. https://roadmetric.com/roadmetric-integrates-its-anpr-with-new-applications.

[4] Wikipedia. *Vehicle Registration Plates of Europe*. Online, accessed 17-April-2018. https://en.wikipedia.org/wiki/Vehicle_Registration_plates_of_Europe.

[5] Wikipedia. *Vehicle Registration Plates of the UE*. Online, accessed 18-April-2018. https://en.wikipedia.org/wiki/Vehicle_Registration_plates_of_the_European_Union.

[6] GOV.UK. *Driving Transport*. Online, accessed 20-April-2018. https://www.gov.uk/displaying-number-plates.

[7] Wikipedia. *Vehicle Registration Plates of the United Kingdom*. Online, accessed 19-April-2018. https://en.wikipedia.org/wiki/Vehicle_registration_plates_of_the_United_Kingdom.

[8] Daniel Llis Baggio, Shervin Emami, David Milln Escriv, Khvedchenia Ievgen, Naureen Mahmood, Jason Saragih, Roy Shilkrot. *Mastering OpenCV with Practical Computer Vision Projects* Packt Publishing Ltd, Birmingham, UK. December 2012.

[9] University of Alberta, Canada. *Image Analysis: Morphological Operations*. Online, accessed 02-August-2018. https://sites.ualberta.ca/ ccwj/teaching/image/morph/.

[10] Ivan Ozighanov. *Instant License Plate Recognition in iOS Apps with OpenCV & GPGPU*. Online, accessed 10-July-2018. https://www.azoft.com/blog/license-plate-recognition-ios/.

[11] Seemal Asif. *Introduction to Machine Learning Concepts and Conventions*. Cranfield University, 2018.

[12] Dr Gilbert Tang. *Instance Based Learning and Clustering*. Cranfield University, 2018.

[13] Dr Gilbert Tang. *Support Vector Machine*. Cranfield University, 2018.

[14] Dr Lance Elliot. *Support Vector Machines (SVM) for AI Self-Driving Cars*. Online, accessed 3-May-2018. https://aitrends.com/ai-insider/support-vector-machines-svm-ai-self-driving-cars/.

[15] Suykens et al. *Disadvantages of Support Vector Machines* Online, accessed 10-July-2018 http://www.svms.org/disadvantages.html

[16] Vinod Kumar Chauhan. *When we use Support Vector Machine for classification* Online, accessed 10-July-2018 https://www.researchgate.net/post/When_we_use_Support_Vector_machine_for_Classification

[17] Z. C. Zou and Xun Guo Lin *Geoinformatics Production for Urban Disasters Risk Reduction: a Zero Cost Solution* Online, accessed 26-July-2018 https://www.researchgate.net/figure/Comparison-of-original-satellite-image-SVM-labelled-result-and-colour-mapped-result_fig3_265014563

[18] Seemal Asif. *A Basic Introduction to Neural Networks*. Cranfield University, 2018.

[19] Tarun Agarwal. *Artificial Neural Networks and Different Types*. Online, accessed 28-April-2018. https://www.elprocus.com/artificial-neural-networks-ann-and-their-types/.

[20] Michael Nielsen. *Neural Networks and Deep Learning*. Online, accessed 26-April-2018. http://neuralnetworksanddeeplearning.com/chap1.html.

[21] Wikipedia. *Artificial Neural Networks*. Online, accessed 26-April-2018. https://en.wikipedia.org/wiki/Artificial_neural_network#Components_of_an_artificial_neural_network.

[22] Xavier Giro-i-Nieto. *Deep Learning for Artificial Intelligence*. Online, accessed 28-April-2018. https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence.

[23] Wikipedia. *Types of Artificial Neural Networks*. Online, accessed 26-July-2018. https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks.

[24] Deeplearningtrack - Online data science school. *Introduction to NEURAL NETWORKS, Advantages and Applications*. Online, accessed 26-July-2018. https://www.deeplearningtrack.com/single-post/2017/07/09/Introduction-to-NEURAL-NETWORKS-Advantages-and-Applications.

[25] Niklas Donges. *Pros and Cons of Neural Networks*. Online, accessed 26-July-2018. https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b.

[26] Tavish Srivastava. *Introduction to K-Nearest Neighbour: Simplified (with implementation on Python*. Online, accessed 29-July-2018. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.

[27] Wikipedia. *K-Nearest Neighbours Algorithm*. Online, accessed 29-July-2018. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[28] Kevin Zakka. *A Complete Guide to K-Nearest-Neighbors with Applications in Python and R*. Online, accessed 29-July-2018. https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/.

[29] Raul Saxena. *KNN SkLearn, K-Nearest Neighbor Implementation with Scikit Learn*. Online, accessed 29-July-2018. http://dataaspirant.com/2016/12/30/k-nearest-neighbor-implementation-scikit-learn/.

[30] Wikipedia. *Box Blur*. Online, accessed 31-July-2018. https://en.wikipedia.org/wiki/Box_blur.

[31] OpenCV. *Sobel Derivatives*. Online, accessed 31-July-2018. https://docs.opencv.org/3.4.1/d2/d2c/tutorial_sobel_derivatives.html.

[32] OpenCV. *Image Thresholding*. Online, accessed 31-July-2018. https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html.

[33] OpenCV. *More Morphology Transformations*. Online, accessed 31-July-2018. https://docs.opencv.org/2.4/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html.

[34] OpenCV. *Structural Analysis and Shape Descriptors*. Online, accessed 31-July-2018. https://docs.opencv.org/3.4.0/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf1

[35] Wikipedia. *Flood fill*. Online, accessed 31-July-2018. https://en.wikipedia.org/wiki/Flood_fill.

[36] Al Sweigart. *Recursion Explained with the Flood Fill Algorithm*. Online, accessed 31-July-2018. http://inventwithpython.com/blog/2011/08/11/recursion-explained-with-the-flood-fill-algorithm-and-zombies-and-cats/.

[37] OpenCV. *SVM Class Reference*. Online, accessed 31-July-2018. https://docs.opencv.org/3.3.0/d1/d2d/classcv_1_1ml_1_1SVM.html.