ANPR

AUTOMATIC NUMBER PLATE RECOGNITION

ABSTRACT

This final project develops an algorithm for automatic number plate recognition (ANPR). ANPR has gained much interest during the last decade along with the improvement of digital cameras and the gain in computational capacity.

The text is divided in four chapters. The first, introduces the origins of digital image processing, also a little resume about the following algorithms that are needed for develop the system ANPR. The second chapter presents the objectives to be achieved in this project, as well as, the program used for his development. The following chapter explains the different algorithms that compound the system, which is built in five sections; the first is the initial detection of a possible number plate using edge and intensity detection to extract information from the image. The second and the third step, thresholding and normalization, are necessary to use the images in the following stages; the text of the plate is found and normalized. With the segmentation, each character of the plate is isolated for subsequent recognition. The last step reads the characters by correlation template matching, which is a simple but robust way of recognizing structured text with a small set of characters. It is evaluated the system's speed and his error rate. Finally, the conclusions and future works are shown in the chapter four.

The databases used consist of images under normal conditions and only Bulgarian's numbers plate.

ACKNOWLEDGMENTS

I dedicate especially this final project for my father, Roberto, who despite not being among us was the person who was more excited to see this project finished and my mother, Julia, for the effort they have made over of their lives to offer all the possibilities of which I enjoyed as well as education and values they have taught me since childhood to the present.

Also I want to dedicate the text to my older sister, Silvia, for the support that gives me, both professionally and academically. At my younger brother, Roberto, for the letters and nocturnal talks through Skype, which make me feel like I was at home. I extend this dedication to the rest of my immediate family, and friends who have accompanied me during this time.

Of course, acknowledge to half of myself, my boyfriend, Iñaki, thanks to him I decided to accept a scholarship Erasmus and taught me that there isn`t sufficient sacrifice for the reward that awaits me.

On the other hand, not least, to the Public University of Navarre by training and acceptance of scholarship Erasmus and my mentor here at the Technical University of Sofia, Milena Lazarova, that despite her great contribution and professional dedication always finds a place for my doubts.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 – INTRODUCTION

1.1 DIGITAL IMAGE PROCESSING (DIP)

It refers to process real world images digitally by a computer. It is a broad topic, which includes studies in physics, mathematics, electrical engineering, computer science. It studies the conceptual foundations of the acquisition and deployment of images and in detail the theoretical and algorithmic processing as such. It also aims to improve the appearance of the images and make them more evident in certain details that you want to note.

This chapter doesn't intend to provide a detailed explanation of digital image processing, but yes an overview of those concepts and methods more important for the realization of this project.

1.1.1 IMPORTANCE OF THE IMAGES

The human uses the senses to iterate with the world they live. The senses allow you to know reality. This way we grasp information about the world around us. We can feel objects, identify smells, hear sounds, detect flavors and most importantly we can see the space in which we live.

Of all the senses the most developed is in sight. It is the means by which we receive information. It allows us to perceive and understand the world around us and accounts for nearly seventy percent of the information we receive. Among this type of information include the identification of faces, reading, images, etc...

The scenes are often perceive three-dimensional (3D) and when we capture by devices (cameras or video, X-ray screens, etc...) we obtain two-dimensional images (2D). The human interacts with a three-dimensional world, when we want to capture a piece by some device usually we get two-dimensional images.

For all these reasons, the images are becoming more prominent role in our society. Personal photographs, video conferencing, real maps, movies, news and audio; all these elements have in common that store images. Therefore we are keen to investigate and develop good systems for image processing.

1.1.2 ORIGINS

The first known application of digital images was in the newspaper industry, where the images were sent through a cable between London and New York. The introduction of image transmission through the cable was in early 1920. During this period, the time for sending images was reduced from a week to less than three hours.

The history of PDI is directly related to the development and evolution of computers. His progress has gone hand in hand with the development of hardware technologies, requiring a high computational power and resources to store and process the images. Similarly, the development of programming languages and operating systems have made possible the continued growth of applications related to image processing, such as medical imaging, satellite, astronomical, geographical, archaeological, biological, industrial applications. The most have common goal to extract specific information from an image, whether for security, control, monitoring, identification, registration and monitoring, among others.

The early work on artificial vision dating from the early 1950. The initial enthusiasm was so great mainly due to greater confidence in the possibilities of computers. Years later, that enthusiasm disappeared due to the limited progress and the few existing applications. Although in the sixties developed algorithms that are used today, such as edge detectors Roberts (1965), Sobel (1970) and Prewitt (1970), its operation was limited to a small number of images and cases. That is why in the seventies there was a gradual abandonment in research.

Since the eighties we start to focus on feature extraction. So there is the detection of textures (Haralik, 1979), and obtain the shape through them (Witkin (1981)). In the same year, 1981, articles were published Stereo vision (Mayhew and Frisby), motion detection (Horn), interpretation of forms (Steven) and lines (Hanade) or corner detectors (Rosendfeld Kitchen (1982)).

The most important work of this decade is the book by David Marr (Vision: a Computational Investigation Into the human representation information and processing of casual information (1982)), which was addressed for the first time a complete methodology of image analysis by computer.

The main reasons for this growth is due in large part to a more realistic approach to solving the problem (for example, begins to be called computer vision rather than artificial vision), the development of computers (increased calculation capacity and decrease in the price) and specialization in processing hardware and imaging.

1.1.3 DIP PROCESSES

The **capture or acquisition** is the process through which a digital image is obtained using a capture device like a digital camera, video camera, scanner, satellite, etc...

The **preprocessing** includes techniques such as noise reduction, contrast enhancement, enhancement of certain details, or features of the image.

The **description** is the process that gets convenient features to differentiate one object from another type, such as: shape, size, area, etc...

The **segmentation** is the process which divides an image into objects that are of interest to our study.

The **recognition** identifies the objects, for example, a key, a screw, money, car, etc...

The **interpretation** is the process that associates a meaning to a set of recognized objects (keys, screws, tools, etc...) and tries to emulate cognition.

1.2 AUTOMATIC NUMBER PLATE RECOGNITION

1.2.1 INTRODUCTION TO THE TOPIC

Due to the mass integration of information technology in all aspects of modern life, there is a demand for information systems for data processing in respect of vehicles.

These systems require data to be archived or by a human or by a special team which is able to recognize vehicles by their license plates in real-time environment and reflect the facts of reality in the information system.

Therefore, several techniques have been developed recognition and recognition systems are license plates used today in many applications.

In most cases, vehicles are identified by their license plate numbers, which are easily readable by humans but not machines. For machines, a registration number plate is just a dark spot that is within a region of an image with a certain intensity and luminosity. Because of this, it is necessary to design a robust mathematical system able to perceive and extract what we want from the captured image.

These functions are implemented or mathematical patterns in what is called "ANPR Systems" (Automatic Numbers Plate Recognition) and mean a transformation between the real environment is perceived and information systems need to store and manage all that information.

The design of these systems is one of the areas of research in areas such as Artificial Intelligence, Computer Vision, Pattern Recognition and Neural Networks.

Systems of automatic recognition of license plates are sets of hardware and software to process a signal that is converted into a graphical representation such as static images or sequences of them and recognize the characters in the plate.

The basic hardware of these systems is a camera, an image processor, an event logger memory and a storage unit and communication. In our project we have relied on images of cars in which we can see their license plate.

The license plate recognition systems have two main points: The quality of license plate recognition software with recognition algorithms used and the quality of imaging technology, including camera and lighting.

Elements to consider: maximum recognition accuracy, achieve faster processing speed, handling as many types of plates, manage the broadest range of image qualities and achieve maximum distortion tolerance of input data.

Ideally, for extreme conditions and with serious problems of normal visibility, would have special cameras ready for such an activity, such as infrared cameras that are much better to address these goals and achieve better results. This is because the infrared illumination causes reflection of light on the license plate is made of special material which causes a different light in that area of the image relative to the rest of it, causing it to be easier to detect.

There are five main algorithms that the software needed to identify a license plate:

1. Location license plate, responsible for finding and isolating the plate in the image. It should be located and extracted from the image for further processing.
2. After the number plate is located and extracted from the image, it can be transformed into a standard format for brightness and contrast.
3. Orientation and plate size, offset angles that make the plate look "crooked" and adjust the size.
4. Segmentation of individual characters is present in plate.

5. Optical Character Recognition (OCR) for each image we segmented individual character. The output of the recognition of each character is processed as ASCII code associated with the image of the character. By recognizing all successive images of the characters are completely read the license plate.

The basic process takes place in the Optical Character Recognition is to convert the text on an image to a text file that can be edited and used as such by any other program or application that needs it.

Assuming a perfect image, namely, an image with only two gray levels, recognition of these characters will perform basically in comparison with patterns or templates that contain all possible characters. However, the actual images are not perfect; therefore Optical Character Recognition encounters several problems:

- The device that obtains the image can enter gray levels that do not belong to the original image.
- The resolution of these devices can introduce noise into the image, affecting the pixels to be processed.
- Connecting two or more characters in common pixels can also produce errors.

The complexity of each of these subdivisions of the program determines the accuracy of the system.

ANRP software must be able to face different potential difficulties, including:

- Poor image resolution, often because the plate is too far, or is the result of using a low quality camera.
- Blurred images, including motion blur and very often in mobile units.
- Poor lighting and low contrast due to overexposure, reflection or shadow.
- An object obscuring (part of) the license plate.
- Evasion techniques.
- Blurred images difficult to OCR; ANPR systems should have high shutter speeds to avoid motion blur.

1.2.2 *APPLICATIONS*

In the parking, the recognition of license plates is used to calculate the duration in which the car has been parked. When a vehicle arrives at the entrance to the parking, the registration number is automatically recognized and stored in the database. When the vehicle leaves the parking later and reaches the door, the registration number of the plate is recognized again and compared to the first stored in the database. The time difference is used to calculate the cost of parking. This technology is used in some companies to grant access only to authorized personnel vehicles.

In some countries these systems are installed recognition throughout the city area to detect and monitor traffic. Each vehicle is registered in a central database and, for example, can be compared to a blacklist of stolen vehicles or congestion control access to the city during peak hours.

They can also be used to:

- Border crossing
- Service stations to keep track of drivers who leave the station without making payment.
- A marketing tool to track usage patterns.

1.2.3 *WORK IN THE MARKET*

There are many projects that develop this topic, such as:

- "Car license Plate Detection and Recognition System". This project develops one appliance of computer vision for the detection and recognition license plate of vehicles. It explains the different process and makes special mention of the tree method used for character recognition.
- "Moving Car License Plate Recognition". Shows the develop and techniques used of this project. The software was developed with MatLab and consolidated with C++.
- "A connectionist approach to vehicle license plate recognition". Uses traditional techniques of segmentation. In addition, it uses a multilayer perceptron neural network to estimate the probability of symbols using scalable features. Markov model is used on official nomenclature sequences plates containing all possible arrangements. In recognition Viterbi algorithm used to compute the most appropriate sequence of symbols, given their chance by the neural network.

- "Number plate reading using computer vision". Describes a new method based on a transverse line that cut horizontally the image looking for the characteristic shape of the plate.

- It should be noted the use of such systems by the London firm Transport for London, which uses more than 200 cameras to record vehicles traveling a particular area of the city. The company uses the information obtained for the collection of so-called Congestion charging (rate designed to avoid urban traffic congestion).

1.3 IMAGE PROCESSING ALGORITHMS

When we analyze an image we can have different objectives:

- Detection of objects: Given an image, select an object using different techniques.

- Extraction of 3D information of the scene, positions, angles, speeds, etc...

- Segmentation is the decomposition of an image in the different parts that compound it.

- Monitoring and correspondence: try to find the equivalence of points between two images.

- Delete noise, enhancement, restoration…

1.3.1 MATHEMATICAL MORPHOLOGICAL

Morphological image processing is a type of processing in which the spatial form or structures of objects within an image are modified.

The intensity map is a binary picture containing scattered pixels marking bright items and individual pixels that are noise. In addition, the edge map can contain much noise from small irrelevant edges. To get rid of some noise and to get the marked pixels around the plate connected, two algorithms called dilation and erosion are used. These algorithms are also called morphological operators. The algorithms take advantage of the fact that in the area of the plate, the marked pixels are closer to each other than in other parts of the picture. The order between these algorithms is very important since the reversed process would give a completely different result.

The process to perform first dilation and then erosion whit the same mask is called a morphological closing operation, the reverse is called opening operation. In this implementation, the masks are altered between dilation and erosion.

The **dilation** process uses a mask, or a structuring element, B, (typically is represented by a matrix of 0s and 1s), which can have different size and pattern for different tasks. The algorithm fills a pixel if the mask centered over that pixel covers one or more set pixels in the input picture, A. The result is that holes in fields of marked pixels are filled and areas of filled pixels grow. Areas closer to each other than the size of the mask are connected.

$$A \oplus B = \left\{ z \mid \left(\hat{B}\right)_z \cap A \neq \varnothing \right\} \forall z \in E \qquad \textbf{(1. 1)}$$

Where E is a Euclidean space or an entire grid and A is a binary image in E.

The **erosion** is the opposite of the dilation. It does also use a structuring element, B, but it only marks pixels if all pixels covered by the mask are set. In this combination, the erosion mask is slightly larger than the dilation mask. The result is that areas connected only by a thin line of pixels are disconnected, that small sets of pixels are erased and the areas of marked pixels shrink.

$$A \oplus B = \left\{ z \mid \left(\hat{B}\right)_z \cap A \neq \varnothing \right\} \forall z \in E \qquad \textbf{(1.2)}$$

Figure 1.1 (a) Image of squares of size 1, 3,5,7,9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

Morphological **opening** removes completely regions of an object that cannot contain the structuring element, smoothes object contours, breaks thin connections and removes thin protrusions. Like opening, morphological closing tends to smooth the contours of objects. Unlike opening, however, it generally joins narrow breaks, fills long thin gulfs, and fills holes smaller than the structuring element.

$$A \circ B = \left(A \Theta B \right) \oplus B \tag{1.3}$$

$$A \bullet B = \left(A \oplus B \right) \Theta B \tag{1.4}$$

Figure 1.2 (a) original image. (b) Opening of (a). (c) Closing of (a).

Other combination of dilate and erode is **Hit-or-Miss Transformation**. Often, it is useful to be able to identify specified configurations of pixels, such as isolated foreground pixels, or pixels that are end points of line segments. Consists of the intersection between one dilate and one erode, with different structuring elements.

$$A \otimes B = \left(A \Theta B_1 \right) \bigcap \left(A^c \Theta B_2 \right) \tag{1.5}$$

1.3.2 DETECTION OF DISCONTINUITIES

We defined edge as the boundary between two regions with different colors or have different regions. Edge detection is the most common approach for detecting meaningful discontinuities in intensity values. Such discontinuities are detected by using first and second order derivatives. The first and second order derivative of choice in image processing is the Gradient and the Laplacian respectively.

We represent an image as an array. In the case of a gray image each pixel is represented by an element of the array. We can detect pixels that have different values of those around them by applying different masks.

W1	W2	W3
W4	W5	W6
W7	W8	W9

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Figure 1.3 (a) Generic mask 3×3 (b) Generic image neighborhood

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \tag{1.6}$$

The values of W mask or Kernel depend on the operation you want performed in the region Z (neighborhood) and R is the result of sum of them.

1.3.2.1 GRADIENT OPERATOR

The gradient of a 2-D function, f (x, y), is defined as the vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{1.7}$$

By vectorial analysis is known of the gradient vector indicates the direction of the maximum variation of *f* in (x, y). A significant amount in the detection of edges is the modulus of this vector:

$$\nabla f = mag(\nabla f) = \left[G_x^2 + G_y^2 \right]^{1/2} \tag{1.8}$$

To simplify computation the magnitude of this vector using absolute values is:

$$\nabla f \approx |G_x| + |G_y| \tag{1.9}$$

The angle at which this maximum rate of change occurs is:

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x} \right) \tag{1.10}$$

The **Sobel edge detector** uses the masks in Fig. 1.2 to approximate digitally the first derivatives G_x and G_y . Each point in the image is convolved with both kernels. A kernel gives a maximum response at vertical edges and the other to a horizontal edge. The maximum value of two convolutions is taken as the output value for that pixel. The result is an edge magnitude image. In other words, the

gradient at the center point in a neighborhood is computed as follows by the Sobel detector:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 1.4 (a) Horizontal Sobel mask (b) Vertical Sobel mask

$$\nabla f = \sqrt{G_y^2 + G_x^2} = \sqrt{\left[\left(z_7 + 2z_8 + z_9\right) - \left(z_1 + 2z_2 + z_3\right)\right]^2 + \left[\left(z_3 + 2z_6 + z_9\right) - \left(z_1 + 2z_4 + z_7\right)\right]^2} \quad \textbf{(1.11)}$$

The **Prewitt edge detector** uses the masks in Fig. 1.3 The parameters of this function are identical to the Sobel parameters but Prewitt detector is slightly simpler to implement computationally than the Sobel detector and it tends to produce somewhat noisier results.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure 1.5 (a) Horizontal Prewitt mask (b) Vertical Prewitt mask

$$\nabla f = \sqrt{G_y^2 + G_x^2} = \sqrt{\left[\left(z_7 + z_8 + z_9\right) - \left(z_1 + z_2 + z_3\right)\right]^2 + \left[\left(z_3 + z_6 + z_9\right) - \left(z_1 + z_4 + z_7\right)\right]^2} \quad \textbf{(1.12)}$$

1.3.2.2 LAPLACIAN OPERATOR

The Laplacian of a 2-D function, f (x, y), is formed from second-order derivatives, as follows:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \quad \textbf{(1.13)}$$

The basic requirement in the definition of digital Laplacian is that the coefficients associated with the central pixel are positive and coefficients associated with the external pixels are negative. Because the Laplacian is a derivative, the sum of the coefficients must be zero. So the answer is zero whenever the point at issue and its neighbors have the same value.

The Laplacian responds to transitions in intensity, and this is rarely used for edge detection. For as is a second order derivative of the Laplacian is too sensitive to noise, produces double edges and is unable to detect the direction of the edge.

As the Laplacian serves as a detector to tell when a pixel is on the side of a bright or dark image.

Commonly is applies digitally in the form of two convolution kernels as shown in Fig. 1.4

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 1.6 Masks of Laplacian

$$\nabla^2 f = 4z_5 - \left(z_2 + z_4 + z_6 + z_8\right) \quad \textbf{(1.14)}$$

$$\nabla^2 f = 8z_5 - \left(z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8\right) \quad \textbf{(1.15)}$$

The Laplacian helps us to find the location of the edges using the zero-crossing property and also plays the role of detecting whether a pixel is light or dark side of an edge.

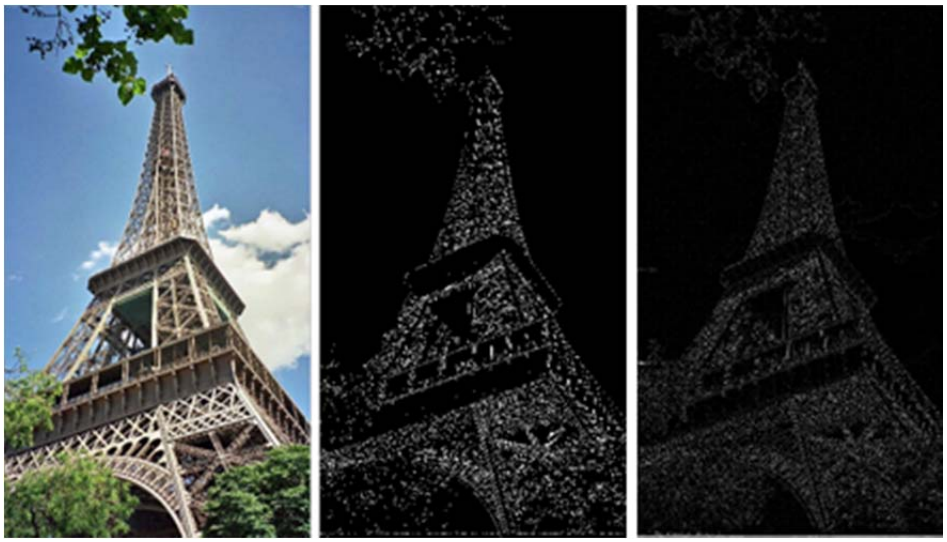


Figure 1.7 (a) Original image. (b) Filtered of (a) by Sobel masks. (c) Filtered of (a) by Laplacian mask.

1.3.2.3 *CANNY DETECTOR*

Canny algorithm is very efficient and gives a step in edge detection, detects significant edges.

This method is based on two criteria:

- The location criterion: It says that the distance between the actual position and the edge localized should be minimized.
- The criterion of a response that integrates the different responses to a single edge.

The method can be summarized as follows:

- The image is smoothed a Gaussian filter, g(m,n), with a specified standard deviation, σ, to reduce noise.

$$g(m,n) = e^{-\frac{m^2+n^2}{2\sigma^2}} \quad \textbf{(1.16)}$$

- The local gradient and edge direction are computed at each point of image.

$$g(m,n) = \sqrt{G_x^2(m,n) + G_y^2(m,n)} \quad \textbf{(1.17)}$$

$$\alpha(m,n) = \tan^{-1}\left(\frac{G_x(m,n)}{G_y(m,n)}\right) \quad \textbf{(1.18)}$$

- An edge point is defined to be a point whose strength is locally maximum in the direction of the gradient.
- The edge points determined in the before step, give rise to ridges in the gradient magnitude image. The algorithm then tracks along the tops of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as *nonmaximal suppression*. The ridge pixels are then threshold using two threshold, T1 and T2, with T1<T2. Ridge pixels with values greater than T2 are said to be "strong" edge pixels. Ridge pixels with values between T1 and T2 are said to be "weak" edge pixels.
- Finally, the algorithm performs edge linking by incorporating the weak pixels that are 8-connected to the strong pixels.

Figure 1.8 (a) Original image. (b) Filtered of (a) by Canny detector.

1.3.3 THRESHOLDING

The thresholding is one of the simplest segmentation algorithms. This technique consists of filter the pixels that form the image so that if they exceed a threshold is set to 0. Otherwise set to 1 or not change. The greatest difficulty lies in choosing the threshold. A variety of techniques have been proposed in this regard.

1.3.3.1 CHOICE OF THRESHOLD

In an ideal case, the histogram has a deep and sharp valley between two peaks representing objects and background, respectively, so that the threshold can be chosen at the bottom of this valley. However, for most real pictures, it is often difficult to detect the valley bottom precisely, especially in such cases as when the valley is flat and broad, imbued with noise, or when the two peaks are extremely unequal in height, often producing no traceable valley.

There have been some techniques proposed in order to overcome these difficulties. They are, for example, the valley sharpening technique, which restricts the histogram to the pixels with large absolute values of derivative (Laplacian or gradient), and the difference histogram method, which selects the threshold at the gray level with the maximal amount of difference. These utilize information concerning neighboring pixels (or edges) in the original picture to modify the histogram so as to make it useful for thresholding.

Another class of methods deals directly with the gray-level histogram by parametric techniques. For example, the histogram is approximated in the least square sense by a sum of Gaussian distributions, and statistical decision procedures are applied. However, such a method requires considerably tedious

and sometimes unstable calculations. Moreover, in many cases, the Gaussian distributions turn out to be a meager approximation of the real modes.

1.3.3.2 OTSU ALGORITHM

In image processing, Otsu's thresholding method (1979) is used for automatic binarization level decision, based on the shape of the histogram. The algorithm (nonparametric and unsupervised method of automatic threshold selection for picture segmentation) assumes that the image is composed of two basic classes: Foreground and Background. It then computes an optimal threshold value that minimizes the weighted within class variances of these two classes. It is mathematically proven that minimizing the within class variance is same as maximizing the between class variance.

We understand an image as a two-dimensional function *f* (x, y) = G. Where G is in the range [0 ... 255]. Pixels with gray value are represented as fi and the number of pixels as N. The probability of occurrence is:

$$p_i = \frac{f_i}{N} \tag{1.19}$$

If there are only two levels, is called binarization, in this case will be C1, with gray levels [1, ..., t] and C2 [t +1, ..., 255].

In the case of two-level thresholding of an image (sometimes called binarization), pixels are divided into two classes (background and objects): C1, with gray levels [1, ..., t] and C2, with levels of gray [t +1, ..., L]. Then, the probability distribution of gray levels for the two classes is:

$$C_1 : \frac{p_1}{\omega_1(t)}, ..., \frac{p_t}{\omega_1(t)} \tag{1.20}$$

$$C_2 : \frac{p_{t+1}}{\omega_2(t)}, \frac{p_{t+2}}{\omega_2(t)}, ..., \frac{p_L}{\omega_2(t)} \tag{1.21}$$

Where:

$$\omega_1(t) = \sum_{i=1}^{t} p_i \tag{1.22}$$

$$\omega_2(t) = \sum_{i=t+1}^{L} p_i \tag{1.23}$$

And the average for the class C1 and C2 class is:

$$\mu_1 = \sum_{i=1}^{t} \frac{i.p_i}{\omega_1(t)} \quad \textbf{(1.24)}$$

$$\mu_2 = \sum_{i=t+1}^{L} \frac{i.p_i}{\omega_2(t)} \quad \textbf{(1.25)}$$

μ_T is the average intensity of the entire image. It is easy to verify that:

$$\omega_1.\mu_1 + \omega_2.\mu_2 = \mu_T \quad \textbf{(1.26)}$$

$$\omega_1 + \omega_2 = 1 \quad \textbf{(1.27)}$$

Otsu defined the between-class variance as:

$$\sigma_B^2 = \omega_1.(\mu_1 - \mu_T)^2 + \omega_2.(\mu_2 - \mu_T)^2 \quad \textbf{(1.28)}$$

For two levels, the optimal threshold t* is chosen such that the between-class variance is maximum, that is:

$$t^* = \underset{1 \le t \le L}{Max}\left\{\sigma_B^2(t)\right\} \quad \textbf{(1.29)}$$

Figure 1.9 (a) Original image. (b) Gray scale image of (a). (c) Result to apply method's Otsu of (a).

1.3.4 HOUGH TRANSFORM

One approach that can be used to find and link line segments in an image is the Hough transform.

Given a set of points in an image (typically a binary image), suppose that we want to find subsets of these points that lie on straight lines. With the Hough transform we consider a point (x_i, y_i) and all the lines that pass through it. Infinitely many lines pass through (x_i, y_i), all of which satisfy the slope-intercept equation as $y_i = ax_i + b$ for some values of a and b. Writing this equation as $b = -x_i a + y_i$ and considering the ab plane (parameter space) yields the equation of a single line for a fixed pair (x_i, y_i).

Furthermore, a second point (x_j, y_j) also has a line in parameter space associated with it, and this line intersects the line associated with (x_i, y_i) at (a', b'), where a' is the slope and b' the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy-plane. In fact, all points contained on this line have lines in parameter space that intersect at (a', b').

The following figure illustrates these concepts:

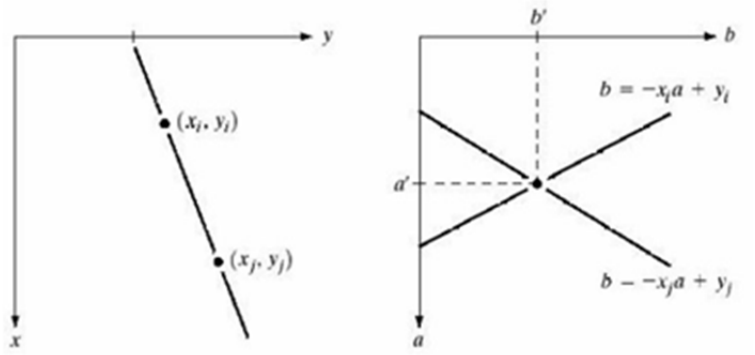


Figure 1.10 (a) xy-plane (b) Parameter space

In principle, the parameter-space lines corresponding to all image points (xi, yi) could be plotted, and then image lines could be identified by where large numbers of parameter-space lines intersect. A practical difficulty with this approach, however, is that a (the slope of the line) approaches infinity as the line approaches the vertical direction. One way around this difficulty is to use the normal representation of a line:

$$x\cos\theta + y\sin\theta = \rho \qquad \textbf{(1.30)}$$

The computational attractiveness of the Hough transform arises from subdividing the ρθ parameter space into so called accumulator cells, where (ρmin, ρmax) and (θmin, θmax) are the expected ranges of the parameter values. Usually, the maximum range of values is $-90^\circ \le \theta \le 90^\circ$ and $-D \le \rho \le D$, where D is the distance between corners in the image.

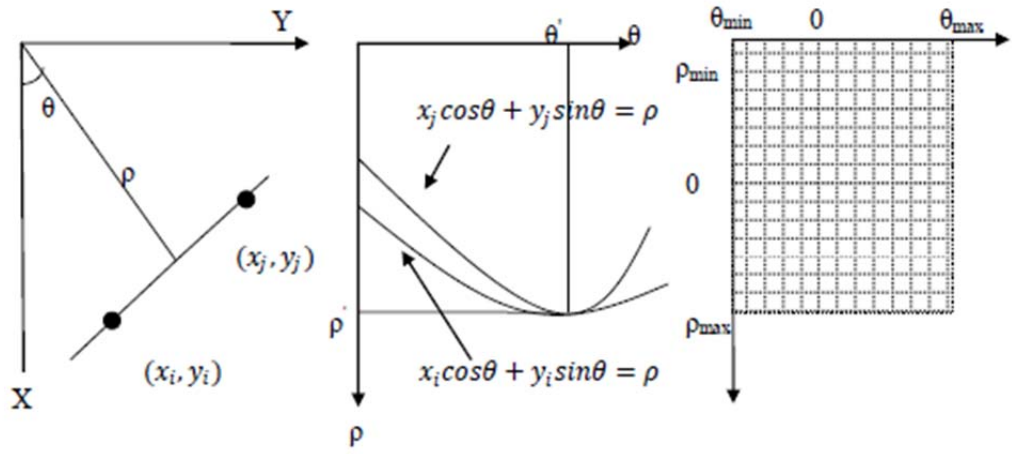


Figure 1.11 (a) (ρ, θ) parameterization of lines in the xy-plane. (b) Sinusoidal curves in the ρθ-plane; the point of intersection ρ', θ' correspond to the parameters of line joining (xi, yi) and (xj, yj). (c) Division of ρθ-plane into accumulator cells.

1.3.5 BOUNDARY AND REGIONAL DESCRIPTORS

When we classify images we analyze previously. The information extracted from an image can be qualitative which is trying to figure out which object is represented in the image, but can also be quantitatively analyzed, different things in images such as positions, colors, intensity, texture...

A region is a connected component (internal features), and the boundary (border or contour) of a region is the set of pixels in the region that have one or more neighbors that are not in the region (external features).

Generally external representation is chosen when the main objective focuses on the features of shape and internal representation when the main interest focuses on the reflective properties such as color and texture. In any case, the features selected as descriptors should be as insensitive as possible to changes in size, translation and rotation.

Some boundary descriptors are:

- Length of a boundary
- Diameter of a boundary
- Curvature
- Shape numbers
- Fourier descriptors
- Statistical Moments

And some regional descriptors are:

- Area
- Perimeter
- Compactness
- Circular ratio
- Mean, median, maximum and minimum intensity values
- Principal axes
- Topological descriptors
- Texture
- Moment invariants

1.3.6 *OBJECT RECOGNITION*

One of the most fundamental means of object detection within an image field is by template matching, in which a replica of an object of interest is compared to all unknown objects in the image field. If the template match between an unknown object and the template is sufficiently close, the unknown object is labeled as the template object.

A template match is rarely ever exact because of image noise, spatial and amplitude quantization effects, and a priori uncertainty as to the exact shape and structure of an object to be detected. Consequently, a common procedure is to produce a difference measure D(m, n) between the template and the image field at all points of the image field where $-M \leq m \leq M$ and $-N \leq n \leq N$ denote the trial offset. An object is deemed to be matched wherever the difference is smaller than some established level $L_d\ (m,n)$. Normally, the threshold level is constant over the image field. The usual difference measure is the mean-square difference or error as defined by:

$$D(m,n) = \sum_j \sum_k \left[F(j,k) - T(j-m, k-n) \right]^2 \qquad \textbf{(1.31)}$$

Where *F(j,k)* denotes the image field to be searched and *T(j,k)* is the template. The search, of course, is restricted to the overlap region between the translated template and the image field. A template match is then said to exist at coordinate *(m,n)* if:

$$D(m,n) < L_D(m,n) \tag{1.32}$$

Now, let equation 1.31 is expanded to yield:

$$D(m,n) = D_1(m,n) - 2D_2(m,n) + D_3(m,n) \tag{1.33}$$

Where:

$$D_1(m,n) = \sum_j \sum_k \left[F(j,k)\right]^2 \tag{1.34}$$

$$D_2(m,n) = \sum_j \sum_k \left[F(j,k) - T(j-m,k-n)\right] \tag{1.35}$$

$$D_3(m,n) = \sum_j \sum_k \left[T(j-m,k-n)\right]^2 \tag{1.36}$$

The term $D_3(m,n)$ represents a summation of the template energy. It is constant valued and independent of the coordinate (m,n). The image energy over the window area represented by the first term $D_1(m,n)$ generally varies rather slowly over the image field. The second term should be recognized as the cross correlation $R_{FT}(m,n)$ between the image field and the template. At the coordinate location of a template match, the cross correlation should become large to yield a small difference. However, the magnitude of the cross correlation is not always an adequate measure of the template difference because the image energy term $D_1(m,n)$ is position variant. For example, the cross correlation can become large, even under a condition of template mismatch, if the image amplitude over the template region is high about a particular coordinate (m,n). This difficulty can be avoided by comparison of the normalized cross correlation:

$$\tilde{R}_{FT}(m,n) = \frac{D_2(m,n)}{D_1(m,n)} = \frac{\sum_j \sum_k \left[F(j,k) - T(j-m,k-n)\right]}{\sum_j \sum_k \left[F(j,k)\right]^2} \tag{1.37}$$

To a threshold level $L_R(m,n)$. A template match is said to exist if:

$$\tilde{R}_{FT}(m,n) > L_R(m,n) \tag{1.38}$$

The normalized cross correlation has a maximum value of unity that occurs if and only if the image function under the template exactly matches the template.

One of the major limitations of template matching is that an enormous number of templates must often be test matched against an image field to account for changes in rotation and magnification of template objects. For this reason, template matching is usually limited to smaller local features, which are more invariant to size and shape variations of an object.

CHAPTER 2 – OBJECTIVES AND TOOLS USED

2.1 MATLAB, DEVELOPMENT TOOL

The name MATLAB comes from the contraction of the terms MATrix LABoratory. It is a computing environment and application development that integrates numerical analysis, matrix computation, signal processing and graphical display in a complete environment. In 2004 it was estimated that MatLab was employed for more than a million people and currently enjoys a high level of implementation in education institutions, as well as research and development departments of many national and international industrial companies.

It was created by Cleve Moler in 1984, raising the first version with the idea of using subroutine packages written in Fortran (a compound word derived from The IBM Mathematical Formula Translating System). MatLab has its own programming language (language M) created in 1970 to provide easy access to software LINPACK and EISPACK matrix without using Fortran. MatLab license currently is owned by MathWorks Inc. It is available for a wide range of platforms and operating under operating systems such as UNIX, Macintosh and Windows.

MatLab main window, called MatLab desktop is shown in the Figure 1.6 It can be seen 5 subwindows: the Command window, the workspace, the current directory, the command history, and one or more windows of figure shown only when the user wants to display a picture or graphic.

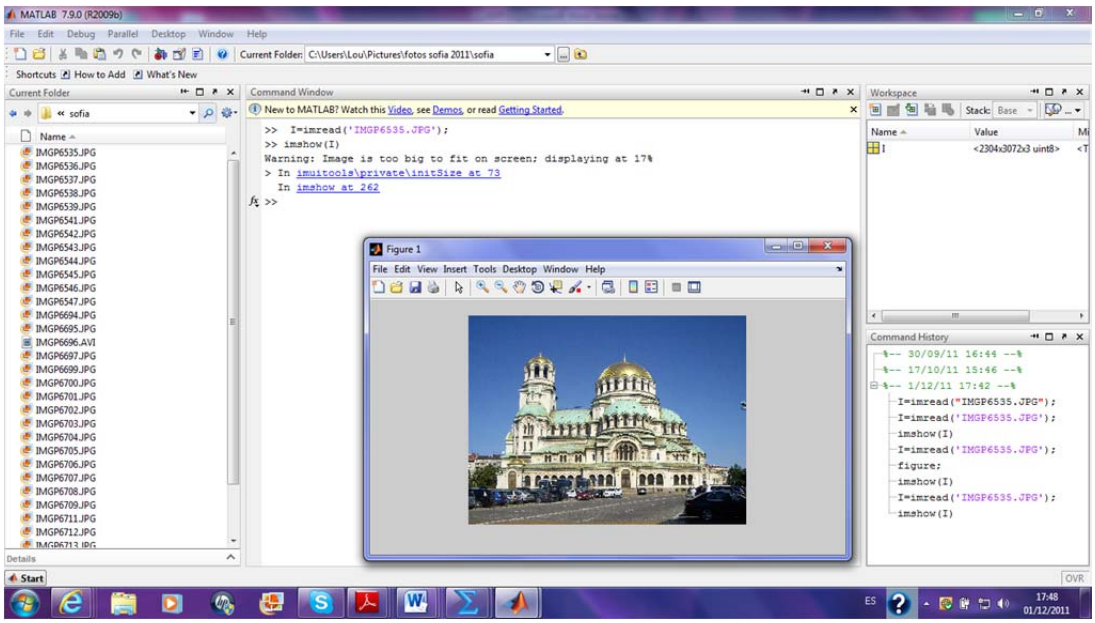


Figure 1.12 MatLab main window

MatLab also offers a wide range of specialized support programs, called Toolbox, which significantly extend the number of features built into the main program. These Toolboxes cover almost all major areas in the world of engineering and simulation, among them: image processing, signal processing, robust control, statistics, financial analysis, symbolic mathematics, neural networks, fuzzy logic, system identification, simulation of dynamic systems, Simulink (multidomain simulation platform), etc.., that is, the MatLab Toolbox provides a set of functions that extend the capabilities of the product for application development and new algorithms in the field of image processing and analysis.

The Image Processing toolbox handles four basic types of images: indexed images, intensity images (grayscale), binary images and RGB images.

MatLab stores most images as two-dimensional arrays (matrices) in which each element of the matrix corresponds to the intensity of a pixel in the image. Some images, such as color images (RGB), require a three-dimensional array where the first plane in three dimensional space represents the red intensity of the pixels, the background represents the intensity of green pixels and the third plane represents the intensity of blue pixels.

To reduce the required memory space for storing images, MatLab stores data in arrays of 8 or 16 bit integer, uint8 and uint16 class, respectively.

To better structure the code employs the use and creation of files. These are files with the extension ".m" used to work with MATLAB functions and scripts.

A script is a sequence of commands that can be run often and that can be saved in a file extension ".m" to avoid having to write again.

The functions are structured code block running when called and let you add additional features to MatLab, thus expanding the capacity of this program. For a long time there was criticism because MatLab is a proprietary product of The Mathworks, and users are subject to and locked to the seller. Recently it has provided an additional tool called MatLab Builder under the tools section "Application Deployment" to use MatLab functions as library files that can be used to build application environments .net or Java. But the downside is that the computer where the application has to be used needs MCR (MatLab Component Runtime), MatLab files to work properly. MCR can be freely distributed with the library files generated by MATLAB Compiler.

2.2 PROJECT OBJECTIVES

The project´s objective is to develop a character recognition system for a license plate of a car using the techniques and tools that best suit this purpose and allow doing it in an easy and comfortable way.

More specifically the project´s objectives are:

- Find a method with acceptable results for the correct location of the area of the license plate.
- Build a system that given an image region where plate can be found, to determine the character of the same.
- Recognize each character we have extracted above by OCR system.

Simultaneously, other objectives emerge as the theoretical study of the techniques of digital image processing and creation of databases for recognition by template matching.

2.3 SCHEMATIC OF THE SYSTEM

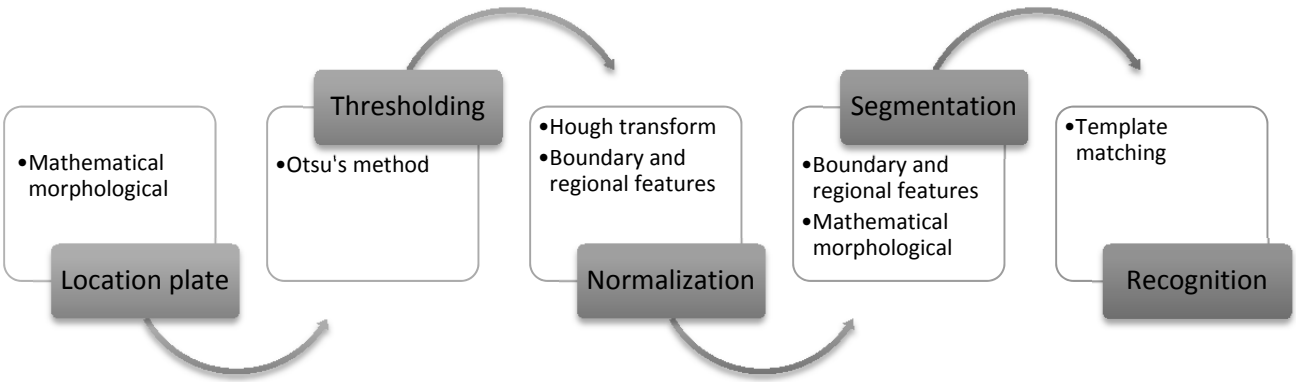


Figure 1.13 Schematic of the system.

CHAPTER 3 – DESIGN AND RESOLUTION

3.1 THE BASIS OF THE ALGORITHM

This chapter explains in detail how the system is implemented and how the algorithms are defined. The mathematical backgrounds and special considerations are described.

The program is designed for automatic number plate recognition of the following characteristic:

- Bulgarian cars
- Rectangle plate
- Single plate (one line of characters)
- White background and black characters
- Arrangement of letters and numbers, LL NNNN LL
- Different ambient conditions (garage, street, snow…)

When a picture is loaded into the program, it has a different resolution depending of the hardware. To reduce the required computational time, the size of the picture is decreased. The reduced smaller picture is used in the processing until the final ROI (Region Of Interest) is found.

During processing, the information about each pixel is sometimes reduced to only be the black/white intensity or a binary value, depending on what the next algorithm requires in order to save memory and improve efficiency.

The first processing step sequence aims at finding and cutting out a ROI, which is assumed to contain the license plate. During this step, intensity detection will be performed to extract feature data from the picture to modify them. And along with morphological operations, it refines the image until the location of registration (ImgPlate).

The next step is to divide the image 'ImgPlate', into subimages as many as characters are recognized. The image pre-processing prior to final segmentation is essential to not confuse impurities with possible characters and successfully return the number of characters with their corresponding images.

Finally, the Optical Character Recognition (OCR) algorithm works by comparing the picture 'ImgChar', with the predefined templates of the typeface used in the license plates of today. The templates are compared by calculating the normalized cross correlation.

At this point, the algorithm has met all its objectives. Remove the plate from the original image, to segment into characters that make up each image and to associate them to the proper character of the template.

3.2 MAIN FUNCTIONS

3.2.1 LOCATION PLATE

As input variable is an image I in color, whose size and resolution depending on the hardware used for acquisition. In order to process any image without a high computational cost we resize the image to 480 * 640 pixels.

We convert the image to grayscale and binary values to carry out the localization process, which is based on extracting the characteristics of the elements of the picture (regional descriptors) to modify them according to whether they fulfill certain conditions depending on case is being valued and go refining the image with morphological operations.

After applying the necessary changes and select the most favorable candidate, we extract it from the input image with the name 'ImgPlate' and display it on screen.

Particularly noteworthy is the case in which the final image has a greater height (65 pixels) or less (25 pixels) that standard measures of plate after this localization process. In this case I opted for a different mask to filter the image, whose function is to delete the background, i.e. modify the capture area.

The code used for the execution of the function is attached in the following pages. Each step is described briefly

```
4  function [ImgPlate] = LocationPlate(I)
5
6  %% Cutting and resizing the original image %%
7
8  [rows columns]=size(I);
9  columns=columns/3;
10 xmin=round(0.20*rows);
11 ymin=round(0.20*columns);
12 width=round((0.85*columns)-(0.10*columns));
13 height=round((0.85*rows)-(0.15*rows));
14 Io=imcrop(I,[xmin ymin width height]);
15 Io=imresize(Io,[480 640]);
16 Io=rgb2gray(Io);
17 Io=imadjust(Io);
18
19 %% Image processing to focus the area of number plate %%
20 %% Smooth edges and contours to delete characters.
21 %% Subtracting the original image to obtain the information
22 %% previously deleted and thus stay with the characters.
23 %% Select the elements with a higher level of 85.
24 %% Pass a mask to the image to remove excess information common to
25 %% all images.
26
27 se=strel('rectangle',[6 30]);
28 Ic=imclose(Io,se);
29 Ic=imadjust(Ic);
30 tophat=Ic-Io;
31 Ibw1=(tophat>85);
32 Ibw=Ibw1 & im2bw(imread('marco.bmp'));
33
34 %% Remove the related elements with fewer than 70 pixels %%
35 %% Remove objects that are not plate %%
36
37 plate= bwlabel(Ibw,4);
38 obj= max(max(plate));
39 dim1 = regionprops(plate, 'area')';
40 dim=[dim1.Area];
41 dim(find(dim<70))=0;
42
43 for i=1:obj
44
45     index=find(plate==i);
46     if dim(i)==0
47         plate(index)=0;
48     else
49         plate(index)=1;
50     end
51
52 end
53
54 CC=bwconncomp(plate);
55 P=regionprops(CC,'all');
56 [rp cp]=size(plate);
```

```
for i=1:CC.NumObjects

    if P(i).MajorAxisLength>(2*cp/3)
        plate(P(i).PixelIdxList(:,1))=0;
    end

end
%% Remove objects that are not candidates for plate %%

se3=strel('rectangle',[30 70]);
r2=imclose(plate,se3);

se2=strel('rectangle',[5 30]);
r=imdilate(r2,se2);

CC=bwconncomp(r);
P=regionprops(CC,'all');

for i=1:CC.NumObjects

    if P(i).MajorAxisLength>(2*cp/3)
        r(P(i).PixelIdxList(:,1))=0;
    end

end

%% select the largest connected component after preprocessing, the
%%plate

plate1= bwlabel(r,4);
dim2= regionprops(plate1, 'area')';
dim1=[dim2.Area];
f=max(dim1);
indMax=find(dim1==f);
plate1(find(plate1~=indMax))=0;

%% cutting of original image %%

[cuty, cutx] = find( plate1 > 0);
up = min(cuty);
   down = max(cuty);
left = min(cutx);
  right = max(cutx);

  img_cut_v = Io(up:down,:,:);
  img_cut_h = img_cut_v(:,left:right,:);

  ImgPlate = img_cut_h;

  %% different mask for location plate %%

  [r c]=size(ImgPlate);
```

```
110  if r<25 || r>65
111
112      [rows columns]=size(I);
113      columns=columns/3;
114      xmin=round(0.20*rows);
115      ymin=round(0.20*columns);
116      width=round((0.85*columns)-(0.10*columns));
117      height=round((0.85*rows)-(0.15*rows));
118      Io=imcrop(I,[xmin ymin width height]);
119      Io=imresize(Io,[480 640]);
120      Io=rgb2gray(Io);
121      Io=imadjust(Io);
122
123      se=strel('rectangle',[6 30]);
124      Ic=imclose(Io,se);
125      Ic=imadjust(Ic);
126      tophat=Ic-Io;
127      Ibw1=(tophat>85);
128      mask=zeros(480,640);
129
130      for i=40:370
131          for j=40:575
132              mask(i,j)=1;
133          end
134      end
135
136      Ibw=Ibw1 & im2bw(mask);
137      plate= bwlabel(Ibw,4);
138      obj= max(max(plate));
139      dim1 = regionprops(plate, 'area')';
140      dim=[dim1.Area];
141      dim(find(dim<70))=0;
142
143      for i=1:obj
144          index=find(plate==i);
145          if dim(i)==0
146              plate(index)=0;
147          else
148              plate(index)=1;
149          end
150      end
151
152      CC=bwconncomp(plate);
153      P=regionprops(CC,'all');
154      [rp cp]=size(plate);
155
156      for i=1:CC.NumObjects
157        if P(i).MajorAxisLength>(cp/3)
158          plate(P(i).PixelIdxList(:,1))=0;
159        end
160      end
161
162      se3=strel('rectangle',[30 70]);
```

```
163       r2=imclose(plate,se3);
164       se2=strel('rectangle',[5 30]);
165       r=imdilate(r2,se2);
166
167       plate1= bwlabel(r,4);
168       dim2= regionprops(plate1, 'area')';
169       dim1=[dim2.Area];
170       f=max(dim1);
171       indMax=find(dim1==f);
172       plate1(find(plate1~=indMax))=0;
173
174       [cuty, cutx] = find( plate1 > 0);
175       up = min(cuty);
176       down = max(cuty);
177       left = min(cutx);
178       right = max(cutx);
179       img_cut_v = Io(up:down,:,:);
180       img_cut_h = img_cut_v(:,left:right,:);
181       ImgPlate = img_cut_h;
182
183   end
184
185   %% Representation %%
186
187   % figure(1);
188   % imshow(I);
189   % subplot(2,2,1);imshow(I);
190   % subplot(2,2,2);imshow(Ic);
191   % subplot(2,2,3);imshow(plate);
192   % subplot(2,2,4);imshow(plate1);
193
194    figure(2);
195    imshow(img_cut_h);title('output location plate');
196
197   end
```

3.2.2 SEGMENTATION

The input variable is the output from Location Plate, 'ImgPlate', corresponding to the image of the gray-scale plate (only plate), and returns the number of characters 'Objects', and a matrix images size [100 100 N] 'ImgChar', that contains the subpictures of the characters found for subsequent recognition.

The procedure can be summarized in five stages:

Apply the method of Otsu to work with a binary image. This procedure selects the optimal level for thresholding depending on the intensity levels of each image.

Eliminate the inclination of the binary image using features of higher object found in the image. I note this orientation and I apply rotation with this angle in the full picture.

Remove impurities larger and smaller than the measurements of a character through the characteristics of each region and morphological operations until there are only eight objects (maximum characters in a plate).

Divide those characters that are together because of previous operations or conditions of the image and original registration. In this loop must be especially careful because dividing the object, the number of them grows, which must be extended 'ImgChar' (100 * 100 * N) in the right place to show and recognize the characters in order and not to alter the sequence of registration.

Finally, remove impurities that were created by segmenting characters, not to return to the next function an array of erroneous images.

```
4   function [Objects,ImgChar]=Segmentation(ImgPlate)
5
6   %% Binarize the image %%
7
8   level = graythresh(ImgPlate);
9   Ibw =(im2bw(ImgPlate,level));
10
11  %% Select the orientation of the largest object in the image.
12  %% Turn this angle at the picture.
13  %% Plate cutting to segment the characters that compose %%
14
15  Fl=bwlabel(Ibw);
16  Statsbf=regionprops(Fl,'all');
17  Flmax=find([Statsbf.Area]==max([Statsbf.Area]));
18  angle=Statsbf(Flmax).Orientation;
19  F2=imrotate(Fl,-angle);
20
```

```
L=bwlabel(F2);
Statsbf=regionprops(L,'all');
maxi=find([Statsbf.Area]==max([Statsbf.Area]));
BB=Statsbf(maxi).BoundingBox;
F2=imcrop(F2,[BB(1,1) BB(1,2) BB(1,3) BB(1,4)]);

%% First three and last three rows to zero.
%% First two and last two columns to zero.
%% So remove connectivity between characters and background %%
%% Remove small impurities %%

L4=not(F2);
[r c]=size(L4);
L4(1,:)=0;
L4(2,:)=0;
L4(3,:)=0;
L4(r,:)=0;
L4(r-1,:)=0;
L4(r-2,:)=0;
L4(:,1)=0;
L4(:,2)=0;
L4(:,c)=0;
L4(:,c-1)=0;

L4b=bwlabel(L4);
Stats3=regionprops(L4b,'all');
sarea3=[Stats3.Area];
G=find(sarea3<70);

for cv=1:length(G)
    G1=find(L4b==G(cv));
    L4(G1)=0;
end
[r c]=size(L4);
CC=bwconncomp(L4);
L=bwlabel(L4);
ind2=max(L(:,c-2));
P=regionprops(CC,'all');

%% Remove objects smaller and larger than a character %%

i=1;
if  (max(P(i,1).PixelList(:,1))-min(P(i,1).PixelList(:,1)))<(c/13)
         L4(CC.PixelIdxList{1,i})=0;
end

for i=1:CC.NumObjects

if (max(P(i,1).PixelList(:,1))-min(P(i,1).PixelList(:,1)))>(2*c/3)
        L4(CC.PixelIdxList{1,i})=0;
end

  if (max(P(i,1).PixelList(:,2))-min(P(i,1).PixelList(:,2)))<(r/3)
```

```
74          L4(CC.PixelIdxList{1,i})=0;
75      end
76
77      if (max(P(i,1).PixelList(:,1))-min(P(i,1).PixelList(:,1)))<(c/8)
78          L4(find(L==ind2))=0;
79      end
80
81  end
82
83  L4=imclose(L4,strel('disk',1));
84  L4=imopen(L4,strel('disk',1));
85  figure(4);
86  imshow(L4);
87  L4b=bwlabel(L4);
88  Stats3b=regionprops(L4b,'all');
89
90  N=length(Stats3b);
91
92  while N>8
93      L4=imdilate(L4,strel('disk',1));
94      L4b=bwlabel(L4);
95      Stats3b=regionprops(L4b,'all');
96      N=length(Stats3b);
97  end
98
99  L4b=bwlabel(L4);
100 Stats3b=regionprops(L4b,'all');
101 ImgChar=zeros(100,100,N);
102
103 %% Dividing characters which are connected %%
104 %% Remove objects that have been listed as characters but are not%
105 %% Show every character in the correct position %%
106
107 cont=0;
108 cont1=0;
109
110 for i=1:N
111
112     [r1 c1]=size(Stats3b(i,1).Image);
113
114     if c1>round(c/6)
115         cont1=cont;
116         Stats3b(i,1).Image(:,round(c1/2))=0;
117         L5=Stats3b(i,1).Image;
118         CC=bwconncomp(L5);
119         CC1=regionprops(CC,'all');
120
121         for j=1:CC.NumObjects
122             [r2 c2]=size(CC1(j,1).Image);
123
124             if c2>round(c/7)
125                 CC1(j,1).Image(:,round(c2/2))=0;
126                 L6=CC1(j,1).Image;
```

```
127                 LL=bwconncomp(L6);
128                 CC2=regionprops(LL,'all');
129 
130                 for k=1:LL.NumObjects
131                    CC2(k).Image=imresize(CC2(k).Image, [100 100]);
132                    figure;imshow((CC2(k).Image))
133                    ImgChar(:,:,i+cont1)=not(CC2(k).Image);
134                    cont1=cont1+1;
135                 end
136             cont=cont+1;
137 
138             else
139 
140                CC1(j).Image=imresize(CC1(j).Image, [100 100]);
141                figure;imshow((CC1(j).Image))
142                ImgChar(:,:,i+cont1)=not(CC1(j).Image);
143                cont1=cont1+1;
144             end
145 
146         end
147     cont=cont+1;
148 
149     else
150         Stats3b(i).Image=imresize(Stats3b(i).Image, [100 100]);
151         figure;imshow((Stats3b(i).Image));
152 
153         if cont~=0
154             ImgChar(:,:,i+cont)=not(Stats3b(i).Image);
155         else
156             ImgChar(:,:,i)=not(Stats3b(i).Image);
157         end
158     end
159 End
160 
161 %% Remove spurious %%
162 
163 [x y Objects]=size(ImgChar);
164 
165     for p=1:Objects
166 
167         if min(min(not(ImgChar(:,:,p))))==1;
168             l=p;
169             while l~=(Objects-1)
170                  ImgChar(:,:,l)=ImgChar(:,:,l+1);
171                  l=l+1;
172             end
173             ImgChar(:,:,l)=ImgChar(:,:,l+1);
174             Objects=Objects-1;
175         end
176     end
177 end
```

3.2.3 *RECOGNITION*

Again, this function takes as input the output variables of the function that precedes it, 'Objects' as number of characters and 'ImgChar' as an array of subimages. The output variable 'strPlate' whose content is a set of char type elements corresponding to the identity of each character.

The procedure is called template matching, and it compares the image obtained from each sub-image of our program with a predefined template. These comparisons are based on finding the maximum correlation, i.e. the maximum similarity between the real image and the template. Each character is tested at each position and the highest probability is stored along with its position, the highest character match is selected. The position of that character is then used as reference to step to the next position and the process is repeated.

Template of the letters, I used a bitmap image, and template of the numbers, I have created a structure where images are entered for each of the 10 possible digits that are random outputs of the previous function because templates were found they are not realistic images resulting after all the process of locating and segmentation.

We perform the template matching differently if the number of characters is 6, 7 or 8. Also addresses the possibility of greater than 8 or less than 6.

If more than 8 we proceed to see if there is a character whose correlation is less than 0.3, which means it is an impurity thus remove that image and reduce the number of objects that compose the image.

Now, we know that if Objects=6, the arrangement of the plate is *L NNNN L*. We compare the first and last element with Baseletters and the rest with Basenumbers and avoid possible erroneous recognition.

If Objects=8, the arrangement of the plate is *LL NNNN LL*. The first and the last two characters are compared with Baseletters, the second with Baseletters1 to not confuse the o's with ‘q’ or ‘g’ (in Bulgarian license plates are not used these letters in this position) and the rest with Basenumbers.

If Objects=7, the arrangement of the plate is *L NNNN LL* or *LL NNNN L*. The objects number 3, 4 and 5 insurance are numbers, 1 and 7 are letters but 2 and 6 can be both.

If in this step the 'Objects' is greater than 8 or less than 6 the output variable will be all zeros.

```
4   function [strPlate] = Recognition(Objects,ImgChar)
5
6   %% Load databases numbers and letters for comparison %%
7
8   Baseletters=im2bw(uint8(imread('Letras.jpg')));
9   Baseletters=bwlabel(Baseletters);
10  L=regionprops(Baseletters,'all');
11
12  letters={'A','N','B','O','P','C','Q','D','R','E','S','F','T','G','
    U','H','V','I','J','W','K','X','L','Y','M','Z'};
13
14  letters1={'A','N','B','O','P','C','O','D','R','E','S','F','T','O',
    'U','H','V','I','J','W','K','X','L','Y','M','Z'};
15
16  for i=1:26
17      L(i).Image=imresize(L(i).Image,[100 100]);
18  end
19
20  N=struct('Image',{});
21  numbers={'0','1','2','3','4','5','6','7','8','9'};
22
23  N(1).Image=imresize(im2bw(uint8(imread('0.png'))),[100 100]);
24  N(2).Image=imresize(im2bw(uint8(imread('1.png'))),[100 100]);
25  N(3).Image=imresize(im2bw(uint8(imread('2.png'))),[100 100]);
26  N(4).Image=imresize(im2bw(uint8(imread('3.png'))),[100 100]);
27  N(5).Image=imresize(im2bw(uint8(imread('4.png'))),[100 100]);
28  N(6).Image=imresize(im2bw(uint8(imread('5.png'))),[100 100]);
29  N(7).Image=imresize(im2bw(uint8(imread('6.png'))),[100 100]);
30  N(8).Image=imresize(im2bw(uint8(imread('7.png'))),[100 100]);
31  N(9).Image=imresize(im2bw(uint8(imread('8.png'))),[100 100]);
32  N(10).Image=imresize(im2bw(uint8(imread('9.png'))),[100 100]);
33
34  %% Treat the image depending on the number of objects found %%
35
36  I=ImgChar;
37
38  %% Maximum of objects is 8, if more, you must remove the leftover%
39
40  if (Objects>8)
41
42      for i=1:Objects
43          char=I(:,:,i);
44          char=not(uint8(char));
45          list_corr=[];
46          for j=1:26
47              corr=corr2(L(j).Image,char);
48              list_corr=[list_corr corr];
49          end
50          for j=1:10
51              corr=corr2(N(j).Image,char);
52              list_corr=[list_corr corr];
53          end
54          f=max(list_corr);
```

```
55          if f<0.3
56              ImgChar(:,:,i)=0;
57          end
58      end
59
60      for p=1:Objects
61          if min(min(not(ImgChar(:,:,p))))==1;
62              for l=p:(Objects-1)
63                  ImgChar(:,:,l)=ImgChar(:,:,l+1);
64              end
65              ImgChar(:,:,l)=ImgChar(:,:,l+1);
66              Objects=Objects-1;
67          end
68      end
69  end
70
71  %% Distinguish between 6, 7 or 8 objects for correlation %%
72
73  if Objects==6
74
75      strPlate=[];
76      for i=1:Objects
77          char=ImgChar(:,:,i);
78          char=not(uint8(char));
79
80          if (i==1) || (i==6)
81              list_corr=[];
82              for j=1:26
83                  corr=corr2(L(j).Image,char);
84                  list_corr=[list_corr corr];
85              end
86              f=max(list_corr);
87              maxcorr=find(list_corr==f);
88              strPlate=[strPlate letters(maxcorr)];
89          end
90
91          if (i==2) || (i==3) || (i==4) || (i==5)
92              list_corr=[];
93              for j=1:10
94                  corr=corr2(N(j).Image,char);
95                  list_corr=[list_corr corr];
96              end
97              f=max(list_corr);
98              maxcorr=find(list_corr==f);
99              strPlate=[strPlate numbers(maxcorr)];
100         end
101     end
102 end
103
104 if Objects==8
105
106     strPlate=[];
107     for i=1:Objects
```

```
        char=ImgChar(:,:,i);
        char=not(uint8(char));

        if (i==1) || (i==7) || (i==8)
            list_corr=[];
            for j=1:26
                corr=corr2(L(j).Image,char);
                list_corr=[list_corr corr];
            end
            f=max(list_corr);
            maxcorr=find(list_corr==f);
            strPlate=[strPlate letters(maxcorr)];
        end

        if (i==2)
            list_corr=[];
            for j=1:26
                corr=corr2(L(j).Image,char);
                list_corr=[list_corr corr];
            end
            f=max(list_corr);
            maxcorr=find(list_corr==f);
            strPlate=[strPlate letters1(maxcorr)];
        end

        if (i==3) || (i==4) || (i==5) || (i==6)
            list_corr=[];
            for j=1:10
                corr=corr2(N(j).Image,char);
                list_corr=[list_corr corr];
            end
            f=max(list_corr);
            maxcorr=find(list_corr==f);
            strPlate=[strPlate numbers(maxcorr)];
        end
    end
end

if Objects==7

    strPlate=[];
    for i=1:Objects
        char=ImgChar(:,:,i);
        char=not(uint8(char));

        if (i==1) || (i==7)
            list_corr=[];
            for j=1:26
                corr=corr2(L(j).Image,char);
                list_corr=[list_corr corr];
            end
            f=max(list_corr);
            maxcorr=find(list_corr==f);
```

```
            strPlate=[strPlate letters(1,maxcorr)];
        end

        if (i==3) || (i==4) || (i==5)
            list_corr=[];
            for j=1:10
                corr=corr2(N(j).Image,char);
                list_corr=[list_corr corr];
            end
            f=max(list_corr);
            maxcorr=find(list_corr==f);
            strPlate=[strPlate numbers(1,maxcorr)];
        end

        if (i==2)
            list_corr=[];
            for j=1:26
                corr=corr2(L(j).Image,char);
                list_corr=[list_corr corr];
            end
            for j=1:10
                corr=corr2(N(j).Image,char);
                list_corr=[list_corr corr];
            end
            f=max(list_corr);
            maxcorr=find(list_corr==f);
            if maxcorr>26
                strPlate=[strPlate numbers(1,maxcorr-26)];
            else
                strPlate=[strPlate letters1(1,maxcorr)];
            end
        end

        if (i==6)
            list_corr=[];
            for j=1:26
                corr=corr2(L(j).Image,char);
                list_corr=[list_corr corr];
            end
            for j=1:10
                corr=corr2(N(j).Image,char);
                list_corr=[list_corr corr];
            end
            f=max(list_corr);
            maxcorr=find(list_corr==f);
            if maxcorr>26
                strPlate=[strPlate numbers(1,maxcorr-26)];
            else
                strPlate=[strPlate letters(1,maxcorr)];
            end
        end
    end
end
```

```
214 %% If there aren't between 6 and 8 objects, the number plate is
    wrong %%
215
216     if (Objects<6) || (Objects>8)
217         for i=1:8
218          strPlate{i}=0;
219         end
220     end
221
222 end
```

3.2.4 ANPR

It is main function, where the output is a list that contains all numbers plate recognized by set of previous functions. In this project the directory name 'image', contains a total of 200 pictures for the execution. All of them are realized for me in a garage of supermarket, in the street in Studentski Grad, Sofia center and Sofia region.

```
4   function [List s]=ANPR(directoryname)
5
6   close all;
7   clear all;
8
9   directory=dir('directoryname');
10  s=max(size(directory));
11  List=[];
12
13      for i=3:s;
14          close all;
15          I=imread(directory(i,1).name);
16          [ImgPlate] = LocationPlate(I);
17          [Objects,ImgChar]=Segmentation(ImgPlate);
18          [strPlate] = Recognition(Objects,ImgChar);
19          [r c]=size(strPlate);
20          if c==6
21              strPlate{7}=0;
22              strPlate{8}=0;
23          end
24          if c==7
25              strPlate{8}=0;
26          end
27          List=[List;strPlate];
28      end
29
30  end
```

3.3 RESULTS

In this chapter I show the tables with the all images executed, the success images and the error images. In this last section I present an image representative of each error and in brackets the total number of images that have this bug.

3.3.1 TABLES

B 4507 PP	C 9601 MC	CA 3337 HT	CA 4650 MA
CA 0717 MX	CA 1357 PM	CA 2108 MX	C 9432 KA
A 7831 KT	CA 0605 MM	C 9601 MC	B 8909 PA
CH 9377 HH	C 6075 PX	CA 4868 PA	CO 0731 AP
CA 7767 MX	CA 6364 MK	C 5810 BA	CO 5491 PA
C 3311 HA	H 6123 AH	C 6592 XX	E 5238 BM
CA 4982 MX	CA 3215 PX	CA 1904 KH	CO 0979 AK
C 0266 XX	CA 1192 BP	CA 8725 KB	PA 8498 BB
BP 5580 BX	CA 2745 KT	CA 7488 PH	CO 2759 AB
KH 0002 AX	C 9768 MK	C 2643 HM	CO 1493 AK
T 5379 CT	C 5260 XK	C 6764 MK	CO 8418 AM
CA 6253 KA	CA 4200 BK	PA 5415 BK	CO 6210 AM
PK 3989 AC	CA 9187 PK	CO 4216 MA	CO 6388 KA
A 3102 KT	CA 6995 HM	CA 3386 MX	CO 4268 AK
CO 3171 CB	E 8350 BB	CO 4259 BB	CO 5471 AH
BT 9108 BH	C 4365 KP	CO 1563 AC	CA 1318 KM
CH 4304 KK	CA 9333 BK	E 4887 BP	CA 7319 CH
CO 3560 AP	PB 0416 MB	E 8350 BB	CO 3442 MA
KH 7188 AX	CA 2185 CP	CM 0427 MC	CA 1282 CK
CA 4855 PA	CA 9015 BX	CO 5942 AM	CO 0766 CB
CA 7849 BH	CA 3329 MA	CO 5127 BB	B 9527 PA
CO 0979 AK	BP 3527 BA	CO 9599 AC	CO 4236 AB
PA 4342 BM	CA 2936 CM	PB 3793 MB	CO 9599 AC
CA 9652 MP	CA 4374 MP	CO 8471 AC	CO 0934 AM

Table 1.1 Table of plates 1

CT 6328 KK	CT 8736 AB	CA 2848 CB	M 9842 AT
X 5876 BB	CT 5253 CP	CO 1185 AH	E 7694 BX
CH 5431 KK	CT 4118 CM	C 4525 HX	C 9588 XH
KH 5563 BA	PB 9763 MX	BP 3895 AT	E 5073 BC
CT 6280 KK	TX 0615 XB	B 2096 AT	EB 1271 AT
CA 3016 AX	BP 3657 AK	CA 0240 CP	CA 8294 AT
CA 2942 KB	CA 4779 PH	KH 7002 AP	CA 2292 PK
TX 3746 XB	C 6224 XT	EH 2876 BC	CA 1755 PA
KH 9315 AH	EH 1490 AK	CA 4112 AB	KH 3798 AT
CA 2942 KB	EH 8525 BP	T 1888 TT	C 7966 XK
CA 1841 PK	M 9842 AT	K 1458 AX	CA 5100 KT
M 3697 AM	KH 8930 AT	CA 8347 HX	CA 4674 KX
A 8624 KA	CA 6947 CB	CA 3595 KT	PA 8323 AT
EH 1323 BH	PK 0108 BA	C 0742 HM	BT 3416 AX
CA 8863 AB	C 4271 MP	CA 5470 HH	CA 1601 HX
C 2108 KA	CO 2048 AC	EH 6328 BP	A 9358 KM
PK 0526 BA	CC 8628 CK	CA 9348 KX	CA 0800 PB
PK 5718 AX	CT 2297 CC	K 1437 AC	P 9166 AP
X 5884 BM	CA 2514 HB	CM 5981 AC	PB 7741 MK
CM 0666 AM	C 9956 KA	K 6475 AC	C 1653 HA
KH 3365 BA	CC 2233 CK	KH 1217 AT	B 0440 KP
B 3645 PK	OB 1839 AX	CT 3742 KK	A 8301 BX
PK 2279 AX	M 0060 BA	X 9597 AP	CA 0833 PC
CO 2178 AP	CA 9761 KB	CT 2951 AB	C 0548 XM
PB 7640 TM	CT 3418 AT	PK 0108 BA	CT 5910 AK

Table 1.2 Table of plates 2

CO 4040 AH	CA 2108 MX	PA 6561 AH	CO 9958 BB
B 4507 PP	C 9601 MC	CA 3337 HT	CA 4650 MA
CA 0717 MX	CA 1357 PM	CA 2108 MX	C 9432 KA
A 7831 KT	CA 0605 MM	C 9601 MC	B 8909 PA
CH 9377 HH	C 6075 PX	CA 4868 PA	CO 0731 AP
CA 7767 MX	CA 6364 MK	C 5810 BA	CO 5491 PA
C 3311 HA	H 6123 AH	C 6592 XX	E 5238 BM
CA 4982 MX	CA 3215 PX	CA 1904 KH	CO 0979 AK
C 0266 XX	CA 1192 BP	CA 8725 KB	PA 8498 BB
BP 5580 BX	CA 2745 KT	CA 7488 PH	CO 2759 AB
KH 0002 AX	C 9768 MK	C 2643 HM	CO 1493 AK
T 5379 CT	C 5260 XK	C 6764 MK	CO 8418 AM
CA 6253 KA	CA 4200 BK	PA 5415 BK	CO 6210 AM
PK 3989 AC	CA 9187 PK	CO 4216 MA	CO 6388 KA
A 3102 KT	CA 6995 HM	CA 3386 MX	CO 4268 AK
CO 3171 CB	E 8350 BB	CO 4259 BB	CO 5471 AH
BT 9108 BH	C 4365 KP	CO 1563 AC	CA 1318 KM
CH 4304 KK	CA 9333 BK	E 4887 BP	CA 7319 CH
CO 3560 AP	PB 0416 MB	E 8350 BB	CO 3442 MA
KH 7188 AX	CA 2185 CP	CM 0427 MC	CA 1282 CK
CA 4855 PA	CA 9015 BX	CO 5942 AM	CO 0766 CB
CA 7849 BH	CA 3329 MA	CO 5127 BB	B 9527 PA
CO 0979 AK	BP 3527 BA	CO 9599 AC	CO 4236 AB
PA 4342 BM	CA 2936 CM	PB 3793 MB	CO 9599 AC
CA 9652 MP	CA 4374 MP	CO 8471 AC	CO 0934 AM

Figure 1.3 Table of plates 3

CT 6328 KK	CT 8736 AB	CA 2848 CB	M 9842 AT
X 5876 BB	CT 5253 CP	CO 1185 AH	E 7694 BX
CH 5431 KK	CT 4118 CM	C 4525 HX	C 9588 XH
KH 5563 BA	PB 9763 MX	BP 3895 AT	E 5073 BC
CT 6280 KK	TX 0615 XB	B 2096 AT	EB 1271 AT
CA 3016 AX	BP 3657 AK	CA 0240 CP	CA 8294 AT
CA 2942 KB	CA 4779 PH	KH 7002 AP	CA 2292 PK
TX 3746 XB	C 6224 XT	EH 2876 BC	CA 1755 PA
KH 9315 AH	EH 1490 AK	CA 4112 AB	KH 3798 AT
CA 2942 KB	EH 8525 BP	T 1888 TT	C 7966 XK
CA 1841 PK	M 9842 AT	K 1458 AX	CA 5100 KT
M 3697 AM	KH 8930 AT	CA 8347 HX	CA 4674 KX
A 8624 KA	CA 6947 CB	CA 3595 KT	PA 8323 AT
EH 1323 BH	PK 0108 BA	C 0742 HM	BT 3416 AX
CA 8863 AB	C 4271 MP	CA 5470 HH	CA 1601 HX
C 2108 KA	CO 2048 AC	EH 6328 BP	A 9358 KM
PK 0526 BA	CC 8628 CK	CA 9348 KX	CA 0800 PB
PK 5718 AX	CT 2297 CC	K 1437 AC	P 9166 AP
X 5884 BM	CA 2514 HB	CM 5981 AC	PB 7741 MK
CM 0666 AM	C 9956 KA	K 6475 AC	C 1653 HA
KH 3365 BA	CC 2233 CK	KH 1217 AT	B 0440 KP
B 3645 PK	OB 1839 AX	CT 3742 KK	A 8301 BX
PK 2279 AX	M 0060 BA	X 9597 AP	CA 0833 PC
CO 2178 AP	CA 9761 KB	CT 2951 AB	C 0548 XM
PB 7640 TM	CT 3418 AT	PK 0108 BA	CT 5910 AK

Figure 1.4 Table of plates 4

LocationPlate function mistake

Segmentation function mistake

Recognition function mistake

FUNCTION	TOTAL IMAGES	ERRORS	% ERROR	% SUCCESS
LocationPlate	200	8	4,0	***96,0***
Segmentation	192	15	7,8	***92,2***
Recognition	177	10	5,6	***94,4***
ANPR	200	33	**16,5**	***83,5***

Table 1.5 Table of results

3.3.2 SUCCESS IMAGES

output location plate

T X 0 6 1 5 X B

Figure 1.14 Image '130.jpg'

output location plate

C A 6 2 5 3 K A

Figure 1.15 Image '13.jpg'

output location plate

C 0 2 6 6 X X

Figure 1.16 Image '9.jpg'

output location plate

K H 0 0 0 2 A X

Figure 1.17 Image '11.jpg'

output location plate

C A 6 9 9 5 H M

Figure 1.18 Image '40.jpg'

output location plate

C A 4 6 5 0 M A

Figure 1.19 Image '77.jpg'

output location plate

B 9 5 2 7 P A

Figure 1.20 Image '97.jpg'

output location plate

E H 1 3 2 3 B H

Figure 1.21 Image '114.jpg'

output location plate

C O 2 0 4 8 A C

Figure 1.22 Image '141.jpg'

3.3.3 ERROR IMAGES

3.3.3.1 LocationPlate mistakes

LOCATIONPLATE
CA 9333 BK
E 8350 BB
C 9432 KA
CO 4268 AK
TX 3746 XB
CA 1841 PK
CA 8863 AB
PB 7640 TM

- Locates the grids of the bonnet and not the plate of a similarity of shape, intensity and color (three images failures).

output location plate

Figure 1.23 Image '111.jpg'

- The mask used as a filter is not suitable (two images failures).

Figure 1.24 Image '43.jpg'

- Locates headlights of the car (three images failures).

Figure 1.25 Image '78.jpg'

3.3.3.2 Segmentation mistakes

SEGMENTATION
CO 4040 AH
CA 7767 MX
PK 3989 AC
E 8350 BB
CA 9015 BX
C 5810 BA
C 2643 HM
CO 5491 PA
CO 8418 AM
CO 3442 MA
KH 5563 BA
X 5884 BM
PB 9763 MX
CA 6947 CB
CA 8294 AT

- Identified as an object start character which is not (five images failures).

output location plate

Figure 1.26 Image '119.jpg'

- Broken character is identified as multiple objects (two images failures).

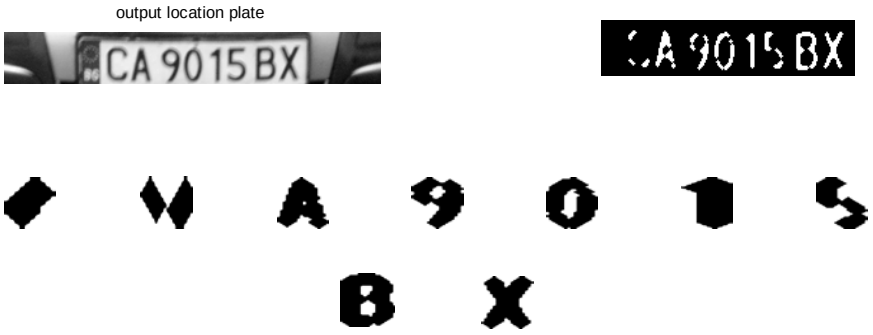


Figure 1.27 Image '46.jpg'

- Character attached to an impurity, the division does not remove invalid objects (three images failures).

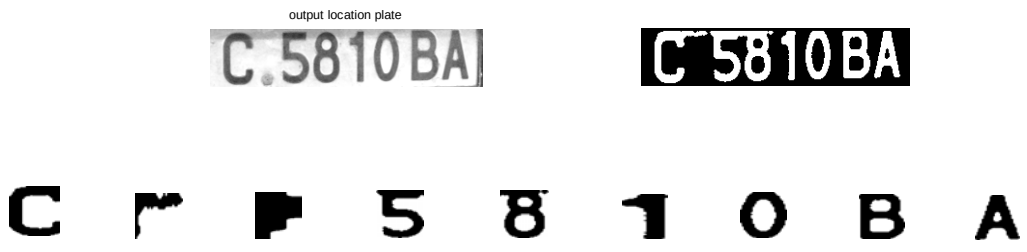


Figure 1.28 Image '56.jpg'

- Upper band of the plate is dark, which makes the characters lose information in that area when the picture is binarized (two images failures).

output location plate

Figure 1.29 Image '81.jpg'

- The input image contains extra information of the plate (three images failures).

output location plate

Figure 1.30 Image '14.jpg'

3.3.3.3 Recognition mistakes

RECOGNITION
B 4507 PP
CA 9652 MP
CA 1357 PM
CO 4259 BB
E 4887 BP
CT 6328 KK
B 3645 PK
CT 8736 AB
BP 3657 AK
EH 6328 BP

- Confuses B by S and 5 or 8 by 6 because of inclination characters (eight images failures).

Figure 1.31 Image '2.jpg'

- Confuses C by Z and 1 by 7 because of impurities (two images failures).

Figure 1.32 Image '28.jpg'

3.3.3.4 ANPR mistakes

A rare problem arises in the main program, if you write on the main screen of MatLab the function body ANPR, everything works correctly, is showing the outputs of each of the functions, but if we call the function directly ANPR (`[List s]=ANPR('image')`) cannot read the directory of images and actually we are doing the same.

CHAPTER 4 – REVIEWS

4.1 CONCLUSIONS

- Each independent function does not exceed an error 8%.
- For all 200 images the error is 17%.
- The total elapsed time of recognition is 2161.36 seconds.
- The average time of recognition of each image is 10.80 seconds.
- The plate status, environmental conditions and the hardware used to catch of pictures are deterministic important factors for the proper functioning program.
- A good image preprocessing almost guarantees a successful recognition.

4.2 FUTURE WORKS

To improve the success of program is needed small improvements at each stage.

The image must be centered, fixed and evenly illuminated during the catch.

Differentiate car color of image under study, i.e. to adapt the preprocessing at car color because of several problems appear in the plate location when the cars are white and silver. Also is possible to do an adaptive mask depending of picture.

Improve the choice of level to threshold and not lose information about the shape of the characters found. Through an adaptive threshold that divides the image into subimages and chooses the most appropriate level in each case, this solution is associated with a significant increase in execution time.

Once characters are segmented the main mistake is that these are distorted or incomplete. Adding a process of reconstruction and the calculation of Hough transform increases the success rate.

There are several solutions that can be applied but keep in mind what you want to sacrifice, if the run time, the quality of image objects, the degree of difficulty of implementation or the hardware and quality cost, between other.

BIBLIOGRAPHY

- William K. Pratt, '*Digital Image Processing: PIKS Inside, Third Edition*', Copyright © 2001 John Wiley & Sons, Inc.
- Rafael C. Gonzalez, Richard E. Woods, '*Digital Image Processing, Instructor's Manual*' Prentice Hall, Third Edition, 2008.
- R. C. Gonzalez, R. E. Woods, S. L. Eddins, '*Digital Image Processing Using MatLab*', Prentice Hall, Second Edition, 2009.
- Regis C. P. Marques, Fátima N. S. Medeiros, Jilseph Lopes Silva and Cassius M. Laprano, *'License Vehicle Plates Localization Using Maximum Correlation'*, Structural, Syntactic, and Statistical Pattern Recognition Lecture Notes in Computer Science, Springer Berlin, 2004.
- N. Otsu, '*A Threshold Selection Method for Gray Level Histogram*', IEEE Transactions on System, Man and Cybernetics, 1979.
- Héctor Oman Ceballos, '*Desarrollo de Tecnología Computacional para la Identificación de Vehículos Automotores Mediante la Visión Artificial*', Thesis, 2003.
- Ondrej Martinsky, '*Algorithmic and Mathematical Principles of Automatic Number plate Recognition Systems*', B.SC. Thesis, 2007.
- Erik Bergenudd, *'Low Cost Real Time License Plate Recognition for a Vehicle PC'*, Master's Degree Project, 2006.
- L. M. Pechuán, *'Uso de Ontologías Para Guiar el Proceso de Segmentación de Imágenes'*, Final Project, 2007.

Automatic Number Plate Recognition
ANPR

PRESENTATION

- Lourdes Jiménez Zozaya
- Spanish student of Public University of Navarre
- Telecommunication engineering specializing in sound and image
- Since October 2011 in Bulgaria doing Final Project and tourism

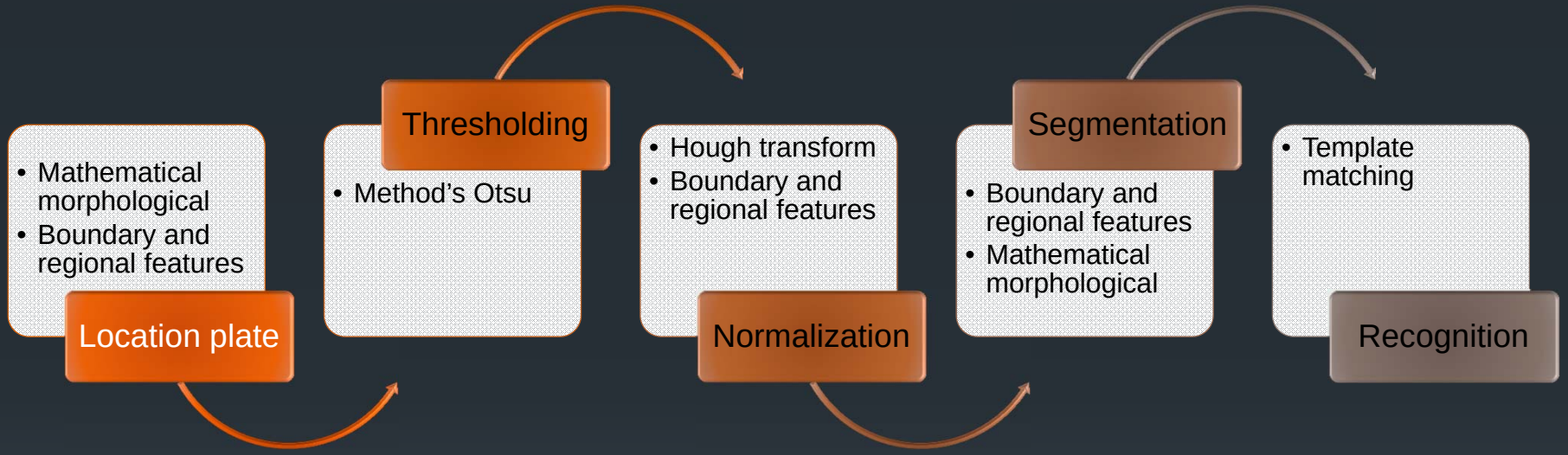
OBJECTIVES

The project's objective is to develop a character recognition system for a license plate of a car using MatLab tool:

More exactly the project´s objectives are:

- Find a method with acceptable results for the correct location of the area of the license plate.
- Build a system that given an image region where plate can be found, to determine the character of the same.
- Recognize each character extracted above by OCR system.

SCHEMATIC OF THE SYSTEM



STAGES

The first processing step sequence aims at finding and cutting a ROI, which is assumed to contain the license plate.

The next step is to divide the image ‘ImgPlate’, into subimages as many as characters are recognized.

Finally, the Optical Character Recognition (OCR) algorithm works by comparing the picture ‘ImgChar’, with the predefined templates of the typeface used in the license plates.

CHARACTERISTICS

The program is designed for automatic number plate recognition of the following characteristic:

- Bulgarian cars
- Rectangle plate
- Single plate (one line of characters)
- White background and black characters
- Arrangement of letters and numbers, LL NNNN LL
- Different ambient conditions (garage, street, snow…)

RESULTS

FUNCTION	TOTAL IMAGES	ERRORS	% ERROR	% SUCCESS
LocationPlate	200	8	4,0	96,0
Segmentation	192	15	7,8	92,2
Recognition	177	10	5,6	94,4
ANPR	200	33	16,5	83,5

SUCCESS IMAGES

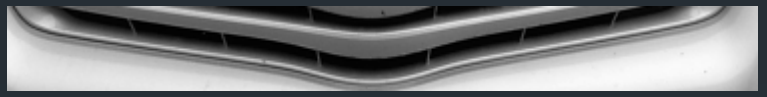
output location plate

output location plate

ERROR IMAGES

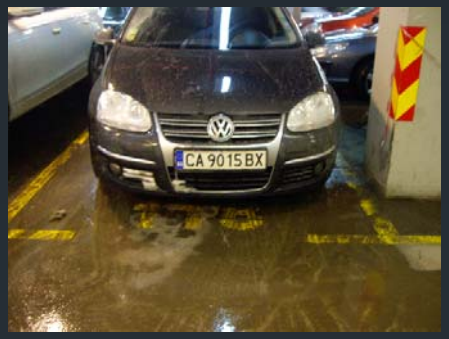
LOCATIONPLATE

output location plate

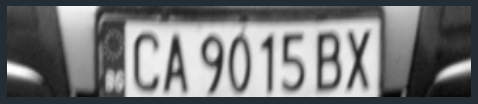


output location plate

SEGMENTATION



output location plate



output location plate

output location plate

RECOGNITION

- The function confuses ‘B’ by ‘S’ and ‘5’ by ‘6’

output location plate

'S' *'4‘* *'6‘* *'0‘* *'7'* *'P‘* *'P'*

CONCLUSIONS

- Each independent function does not exceed an error 8%.
- For all 200 images the error is 17%.
- The total elapsed time of recognition is 2161.36 seconds.
- The average time of recognition of each image is 10.80 seconds.
- The plate status, environmental conditions and the hardware used to catch of pictures are deterministic important factors for the proper functioning program.
- A good image preprocessing almost guarantees a successful recognition.

SOLUTIONS

- Better hardware
- Adaptive mask (LocationPlate)
- Adaptive threshold level (Segmentation)
- Transform’s Hough (Recognition)

PROBLEMS FOUND

- Have to do the project in other different language.
- Take photographs in the street, in public garages for to complete database.
- Select the different methods to apply in functions.
- Make changes in the code when the end was near.
- Execute ANPR function with a call of function.