License Plate Recognizer
Numerical Calculations for Engineering

International Semester

Pelle Reyniers

May 9, 2019

Due Date:	9 May 2019
Class :	International Semester

Abstract

Matlab is a key example of software used by every engineer, no matter the profession or sector. The versatility of possibilities is so larger that this software is used from pure mathematical calculations up to chemical simulations. This project is perfect to illustrate that fact. The main objective of this project is designing a system that can recognize a license plate from a picture of a car. As a theoretical example it makes a very good bridge to the real world because systems for security and traffic control make constant use of these kind of software. When coupling it back from the real world engineering example to the theoretical, we can see a wide range of interests. The first and most obvious one is the ability to calculate a lot of mathematical problems with software and write custom made functions. More important is the image manipulation what is in fact a certain way of signal processing. By working with image manipulation a lot is learned about how computer systems store and use images.

The objective of recognizing license plates seemed through the process not that easy as first thought. Although it is not difficult to design a system that can recognize certain things in a picture, it is much more complex when these things are not predefined. There is a classic saying referring to this matter: ”Although a computer can calculate a thousand ways to defeat you in a chess game before you know your next move, it cannot distinguish an apple from a banana”. This saying is completely out dated because with the entrance of machine learning and AI, computers can for a while now detect almost everything. But my project is not based on AI or machine learning algorithms and keeps 100% in the world of pure mathematical calculations, therefor the saying is perfectly applicable to my project.

As a result the system described in this report is perfectly capable of recognizing a license plate when following conditions are met: The license plate has to be clearly visible, and the characteristics have to be pre known.[1] The system is not capable of detecting all license plates without pre knowledge. The system does only accept images as input, not a camera with moving images and the system make use of the CPU, not of parallel processing.

This makes it possible to state the following conclusions: The system does for fill it’s purpose but the system can be used as a starting point for a better and more general system.

[1]With characteristics is meant how many characters and how many of those are letters.

Introduction

This document is a report describing a project. The project being reported is part of the course *Numerical Calculations for Engineering*, element of the *International Semester* offered at *Escuela Técnica Superior de Ingeniería y Diseño Industrial*, part of *Universidad Politecnica de Madrid*. This course is presented by *ALBARRACIN SANCHEZ RICARDO* and *CASTANO SOLIS SANDRA*.
The course consists of multiple lectures and examples all related to Matlab and more imported the link between Matlab and the engineering world. Everything in the subject is teached and showed with real life examples.
The evaluation of this course consists of multiple smaller assignments and one large final project. This document describes that final project.

Contents

List of Figures

Listings

1 Problem Introduction

The general information about the project reported in this documented is presented in this section.

1.1 Description

In recent years or decade, a new way of traffic/speed control is introduced. The so called trajectory control [1]. This way of traffic control exists in monitoring traffic at entry and exit points of a given trisect. With monitoring is meant the identification of vehicles. In this case cameras are used with a computer system that recognizes license plates. By comparing times at entry and exit point an average speed can be calculated. In general this method of speed control is better than normal one point control due to the avoidance of the break-and-accelerate syndrome. This system can be used for a lot more than just speed control. By analyzing data, conclusion can be drawn why people change lanes, etc. Maybe it will become possible to predict traffic jams, accidents etc. Both present systems and futuristic extension relay on the basic principle of license plate recognition from images. This is a very interesting and also fundamental subject in the modern world.

1.2 Matlab functionality

The recognizing of license plates is no more than an Image processing problem. This is possible with Matlab because in digital terms a photo is a matrix of values and can be analyzed.[2][3] Matrix manipulation is very straight forward in Matlab, you could say Matlab is build for Matrix manipulation. In this way is this a perfect example as a Matlab project.

1.3 Goals and Objectives

For a project of any size it is very important to clearly define the goals. This gives a clear view off direction, whether the project consists of research, development or even experiments.

- Creating a script that manipulates the images so the license plate becomes clear.
- Creating a script that returns the license plate when (in string format) when a picture is given as input.
- Provided enough test data and results to confirm all stated conclusions.

1.4 Possible Extensions

Further in this report is a larger section devoted to possible extensions, this is only a foretaste.

1.4.1 Moving Images

In real life this system works with cameras so with image processing on frames. An extension can be giving an input of video files instead of just pictures.

1.4.2 CUDA

CUDA is a programming language built on top of C that lets the user control the Graphic processor.[4] Because image processing contains a lot of parallel calculations, it can be interesting to see what the gain is when executing on a GPU.[5]

2 Problem Breakdown

In this section the general problem is brought down to simpler and smaller sub problems. The goals of this section is not making more objectives but making a clear and good dividends in the project. Important in breaking down a problem is the clear definition off the sub problems. It has to be very clear what every part consists off and most important when a sub problem is solved. Out off this last definition follows that it has to be possible to test every different sub problem in a convenient way. To summarize: a project consists off different sub problems which all have the following.

- Title
- Clear and simple objective.
- Orientation within the whole project.
- Test conditions.

Multiple sub problems can be combined into a project phase. When talking about project phases, the following phases can be defined, there is a large resemblance with the goals defined in the previous section.

- Image manipulation: Manipulate the input image to a form where it is easy to start the process of finding the license plate.
- The search: The search off the entire (manipulated) image to the license plate.
- Evaluation: Confirmation or denial.

These project phases each consist of multiple sub problems, these are defined in the following subsections.

2.1 Image manipulation

As earlier described consists this project phase of preparing the image for the search off a license plate. The preparation of this image consists of multiple steps. Each of these steps can be considered a sub problem.

1. **Uniform:** Transform the image to a uniform dimension. In this way all images from this point on have actually the same characteristics.
2. **Greyscale:** Transform the image to a greyscale version off itself.
3. **Noise:** Removal of possible noise in the picture.[6]
4. **Edges:** Detecting off all the possible edges in an image.[7]
5. **Clear edges:** Clear the image to make detected lines the only visible things.
6. **Fill edges:** Detect non straight edges and fill them.

2.2 The Search

Before getting into the search, first a few words of explanation about the divide between the image manipulation and search process. We live in a world were a lot of things are the same in the whole world, a few of them are: the use of license plates, the need for speed control, and things in these categories. When looking at license plates, although the concept of a license plate seems universal, it is easy to recognize that they are different around the world. Some of them have only letters, others have two letter and four numbers, etc. Even inside the European union there are a lot of differences. When building a license plate recognizer it is clear to see that a big part of the process will not change: this is the recognizing of edges and filling possible areas. This will always result in an image with interesting areas coloured in, no matter the composition of the license plate to be found. It is because of this fact that I divided search and image manipulation. No matter which (country) license plate that has to be found. The image manipulation part will always be the same.

1. **Find candidates:** this sub problem is the deliverance of X containers that are possibly part of the license plate.
2. **Test:** testing of the different candidate sets.
3. **Choice:** Make a choice which solution is the most predictable.

These are only three sub problems but they are larger in proportion than the sub problems handled in the Image Manipulation section. The span multiple functions and files, it is possible to say that each of these problems have own sub problems. Therefor are they maybe not in line with how I defined sub problems through this project. Further in this subsection follows a more in detail division for every sub problem.

2.2.1 Find Candidates

This sub problem can be further divided into the following sub problems:

1. **Regions:** Which regions with containers are interesting.
2. **Selection:** Filter every interesting region.

It is difficult to further split all the possible sub problems because of the overlap.

2.3 Evaluation

The evaluation part is straight forward. There is only one goal:

1. **Evaluate:** define a function that really tests all parts of the Matlab script.

This were all the sub problems for this project. In the next section is every problem going to be discussed in a proper way. It is normal that a lot of the causes are not clear and completely understood. The main objective of this section was giving a brief overview and also some structure to relate back to in future sections.

3 Component Overview

3.1 Project Overview

In this subsection follows a description about the structure of the scripts. The main script is called *ReadLicensePlate.m* and can be found in listing 1 on page 19. In this script are a few things defined:

- The name of the input file.
- The amount of characters in a license plate.
- The amount of letters in a license plate.

The three different parts of this project are really visible in this script. The first part: Image Manipulation is done in a different script but started from *ReadLicensePlate.m.* The second part: The search, also started in *ReadLicensePlate.m*, is also mostly done in other scripts. The last part: evaluation is done outside this script. The Image manipulation script can be found in listing 3 on page 20. This script uses only one custom made function, defined in listing 2 on page 20. Part two starts with calling the function findIndices, the script with the Matlab code can be found in listing 4 on page 21. The *findIndices.m* function returns a possible license plate, to get to this license plate the function makes use of an other Matlab function called *eliminateOptions.m.* The code corresponding to this function can be found in listing 5 on page 23. Afterwards everything returns to the main script *ReadLicensePlate.m.* This is in large lines the structure of the project. Of course there are other scripts used for extra functions like matching a container to a letter, but the large parts are explained above.

3.2 Image Manipulation Components

Here follows an implementation of all the sub problems introduced in the previous section. Before starting with discussing the different sub problems, the objective of this phase is transforming the image to a new image were the following thing has happened: edge detection to define possible areas where the license plate can be. To do this, the sub problems defined in the previous section, subsection image manipulation are implemented. This implementation is shown in the current sub section. The corresponding Matlab code can be found in listing 3 on page 20.

3.2.1 Image Preparation

First steps are making the image uniform and removing the color without losing elements in the picture. The color removing is called greyscaling. In the project is a separate script present that reads and greyscales the image. Therefore can sub problems 1 on page 2 and 2 on page 2 be combined when talking about the implementation. The Matlab code corresponding to these sub problems can be found in listing 2 on page 20. These sub problems are solved in the custom made Matlab function called *[rgbImage,greyImage] = greyscale(inputFileName).* This function has an input argument and produces two output products. As an input the name of the file to read can be found. By making this a variables it is easy to change the change the file name, what is only good for the dynamic carater of the application. When looking at the output arguments, both an rgb and greyscale image can be found. This happens to keep the original in the system while executing the program. When looking at the body of this function, the try catch is present to catch any possible error. Because the project consists of a lot of components, it is a very handy feature to always have a more detailed explanation when the program crashes for one reason or another. While the first lines take care of the importing of the picture and assuring that it is an rgb version.[2] Line 15 in the code solves sub problem 1 on page 2. The image is resized using the command *imresize()*, from now on all the images are the same size. The lines 18 to 25 take care of the greyscaling. Greyscaling an image is done by separating the r, g and b channels, multipling each channel by a certain factor and recombining the new value. New value, as in singular value, because in a greyscale image are the r,g and b values equal for each pixel. The factors used can be found in equation 1.

$$Y_{linear} = 0.2126R_{linear} + 0.7152G_{linear} + 0.0722B_{linear} \tag{1}$$

The value of these factors are defined in the *CIE 1931.* [8]. When everything is executed as supposed to, the function now returns both the greyscale and rgb image, otherwise the inputimage is returned twice.
From this point on, a greyscaled image is used in the script. Before starting with edge detection it is use full to remove the noise from a picture.[6] The noise removal is done with the command *medfilt2().* [9] This function also accepts an gpuarray input, this is a better and faster way because the matrix manipulations will be done by the GPU. The GPU has a significant more cores to calculate results. The GPU functionality does rely on the CUDA compatibility of the GPU in the computer it is running on. Because this is not generally supported, I made the decision to not go forward with GPU implementation in the standard version of my project. With removing the noise, sub problem 3 on page 2 is solved.

3.2.2 Edge Detection

After preparing the image, it is now ready to start the edge detection.[7] Before discussion the implementation, a short explanation what edge detection is and how it can be achieved. An edge in an image is an area where the rgb pixel values drastically change. For this reason it was important to remove the color without losing any value of the image.
Imagine not using a greyscaled image but a full colored rgb image.[8] Looking for edges is far more complicated because we have three different channels to take into account. When using a greyscaled image, the r g and b channels have the same value what makes the detection a lot easier. One of the most used detection technique's is both dilating and erroding the image.[10] The difference between the two resulting images will result in a very good edge detection.
To successfully dilate an image, it is necessary to have a neighborhood search area, this can be created using a *strel* function.[11] Imagine to have a neighborhood that looks like a circle with radius of a few pixels. This is shown in figure 1 on the following page. This *strel* is moving over an image, when it covers an area with the same rgb value, the system knows it is in an area without a border. But when the *strel* is in an area that is covered with more than one different value, it is clear that their is a border present. When there is a pixel of the *strel* that has a lighter value [2] than the center pixel, the value of this center pixel becomes this lighter value. When this is performed on an image, the lighter areas on the image will become bigger. This is shown in figure 2 on the next page.
To successfully erode an image, it is necessary to have a neighborhood search area, this can be created using a *strel* function.[12] Imagine to have a neighborhood that looks like a circle with radius of a few pixels. This is shown in figure 1 on the following page. This *strel* is moving over an image, when it covers an area with the same rgb value, the system knows it is in an area without a border. But when the *strel* is in an area that is covered with more than one different value, it is clear that their is a border present.[12] When there is a pixel of the *strel* that has a darker value [3] than the center pixel, the value of this center pixel becomes this darker value.

[2]Lighter in greyscale value means a higher value
[3]Darker in greyscale value means a lower value

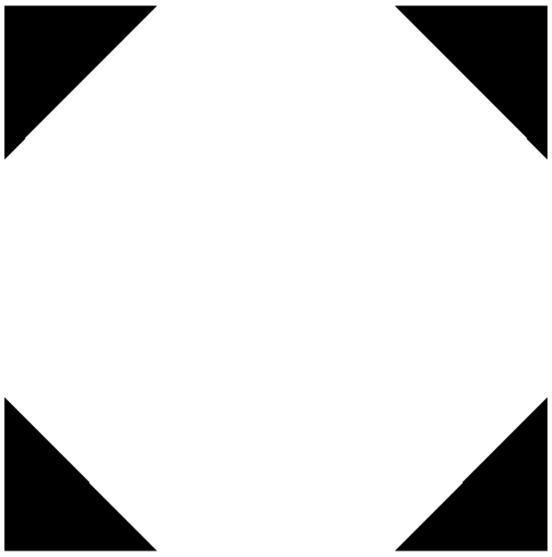

Figure 1: Example of a strel neighborhood.

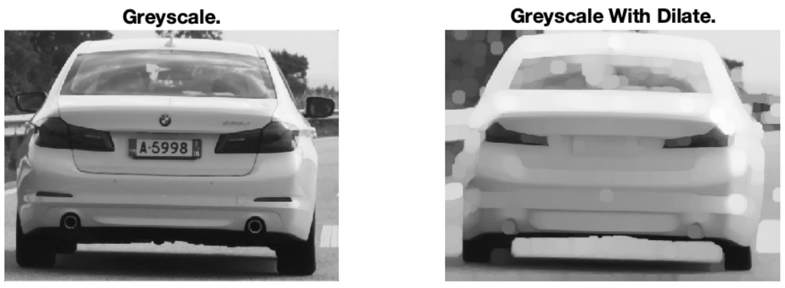

Figure 2: Example of a dilated image.

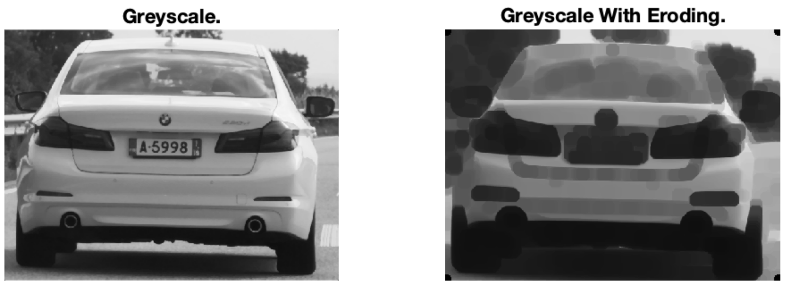

Figure 3: Example of a eroded image.

When this is performed on an image, the lighter areas on the image will become smaller. This is shown in figure 3.

In the previous paragraphs, dilating and eroding is exaggerated. When the radius of the *strel* is reduced to 1, the edges can be exact determent when the the results are subtracted. The result of this is shown in figure 4 on the following page. With this result we can conclude that sub problem 4 on page 2 is solved.

With a quick recap we can recognize the fact that at this point we have an image with recognized edges. However the goal of the image manipulation part is delivering interesting regions where to search for a license plate. To get to that point there are two more necessary sub problems to solve: clear edges and delete non interesting edges, fill the interesting edges. We start with clearing the image. The start point is an image as shown in figure 4 on the following page. To make a difference between the different kind of edges, it is necessary to em brighten the edges and completely en darken the rest of the picture. This is possible by making the contrast bigger and than convert the image to a binary map. By using a binary map, there are only two options: a border or no border, a 1 or a 0. When converting the binary map back to a "normal" image we are now shore we have an image with only the borders/edges present. The result at this point is an image with only clear edges, therefor it is safe to say that the objective of sub problem 5 on page 2 are achieved.

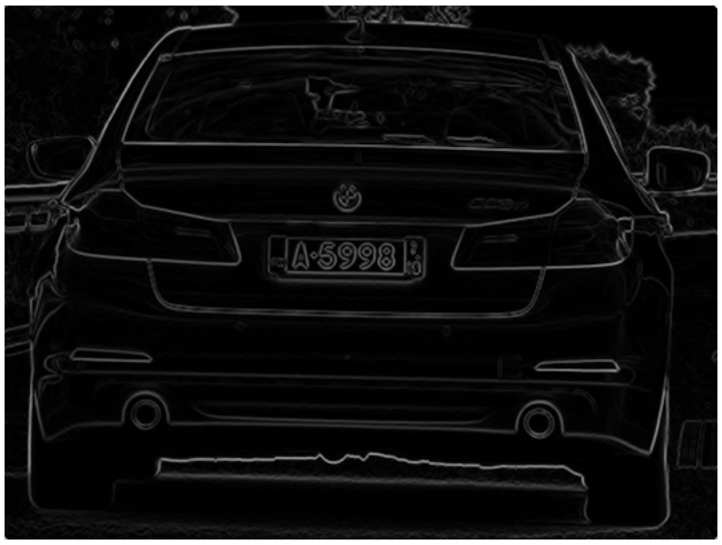

Figure 4: Example of an all edge detection

3.2.3 Filling

As stated in the previous paragraph, the objective of the image manipulation is the presentation of interesting regions. What are interesting regions in the picture? Regions that possible contain a number or letter. A property of numbers and letters is that they mostly do not have a lot of straight lines. especially no single straight lines. With this property in mind we can use another *strel* function to remove straight lines. Instead of using a disk size strel, it is possible to use a linear strel, this will help detect the non interesting edges. With the function *imfill()*, it is possible to fill enclosed regions in the image. This is applied to the image, in theory, every enclosed region can be a letter or a number. Non filled edges are thinned out with the result that the difference between them becomes bigger. Results off these manipulations are shown in figure 5 on the facing page. These manipulations are combined with the earlier talked about linear edge removal. An example of the final result after image manipulation is shown in figure 6 on the next page. Before starting with the search part, one addition is done to the image manipulation. The function *Iprops()* is called on the final result from the image manipulation. This function produces containers over all the the different interesting parts in the pictures. Two parts of this function are very interesting for the use in this case. The tag 'Bounding Box', this produces practical information about the container in the form of four elements:

- Starting X coordinate.
- Starting Y coordinate.
- Height.
- Width.

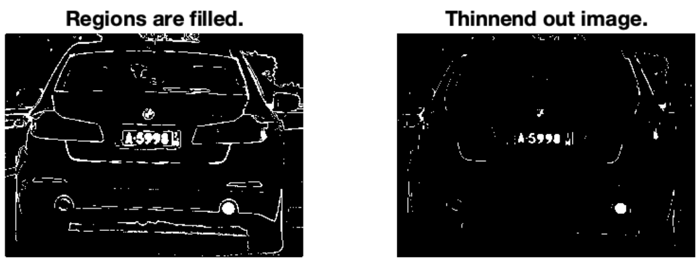

Figure 5: Example of region filling and thinning out the image.

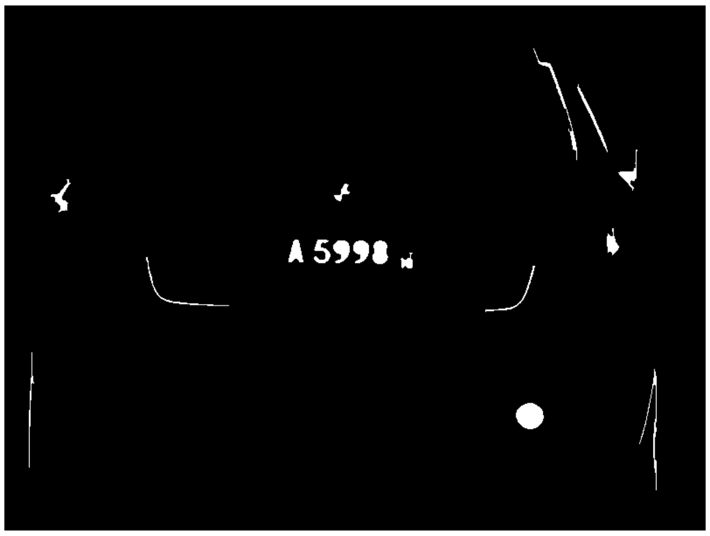

Figure 6: Example of the final result after image manipulation.

In that way are the complete dimensions of the container given. The second tag is 'Image', this produces the image corresponding to the container dimensions. Now all the elements are there to start the search operation.

3.3 The Search

The use of making a difference between image manipulation and the search is explained in the "Problem Breakdown" section. In this subsection follows a detailed explanation of how the search works and how to adapt the search to different license plate layouts.

3.3.1 Find Candidates

The main objective is also the biggest question, how do start searching in the picture. We know we have a picture with the regions selected but how can we confident say which regions to skip? The answer to this question can partly be found in the knowledge we have about the containers. We can assume the following: the containers corresponding to the different characters of the license plate will be in the same Y dimension. So if it is possible to filter the containers on Y dimension it will be clear which regions are interesting and have possibly the license plate.
From the main script *ReadLicensePlate.m*, listing 1 on page 19, the function *findIndices* is called. In this function, the sorting in regions start. This is possible using the *hist()* function, with parameter the Y starting point of the different containers. The hist function makes a histogram, standard dividing the complete range into 10 parts. This function uses a BucketSort mechanism.[4] After executing this function it is possible to see how much containers there are in every region. Because the specifics of the license plate are given in the main script, it is justified to say that all regions with less containers than total license plate characters are not interesting. In this way we only keep the regions with a lot of containers.
Only filtering on Y coordinate is not enough. It is safer to make a safety mechanism with another factor. By multiplying the Y coordinate with the width of the container we get a new factor to filter. This second factor is used as a backup when the first filter is unable to give use full results. After the filtering, all the regions with at least as many items as there have to be characters in the license plate, are selected. Every region is looked into. The histogram corresponding to the example used in the image manipulation can be found in figure 7 on the next page. The next major challenge is that regions can have more containers than that there are characters. A selection procedure is necessary to select which of these containers are not part of the license plate. This selection is done in the function *eliminateOptions*. The function gets the selection of containers as input as well as the images corresponding to these containers, the amount of characters in a license plate and the amount of letters in a license plate. This function is run in a for-loop on every interesting region.

3.3.2 Testing

The testing happens in the function *eliminateOptions*, can be found in listing 5 on page 23. All containers in the region are matched to a character, by the function *readLetter*. This function not only returns the letter that is matched to the corresponding container but also the correlation factor. This correlation factor is very important because after the matching, the filtering has to start. The filtering of the containers is done in a while loop and continuous as long as there are more characters left than that there are in a license plate. In every iteration of the while loop will the character with the lowest correlation be tracked down and deleted. In this way it is safe to say that the best option for this region will come out of this function.

[4]BucketSort is a fast but not so precise sorting algorithm.

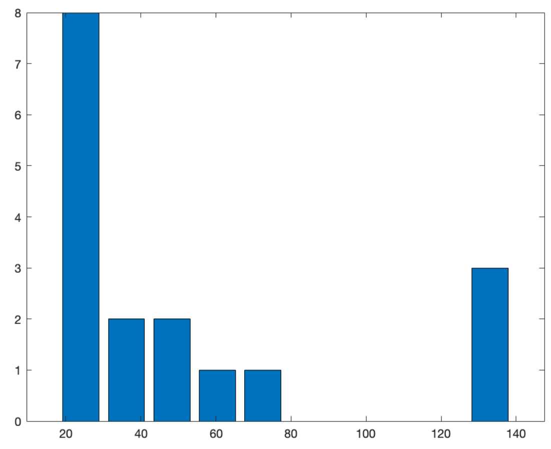

Figure 7: Histogram corresponding to the picture used in the image manipulation example.

3.3.3 Choice

This sub problem is solved in the function *ReadLicensePlate*. After the for-loop done on all possible regions, we have a possible license plate for every interesting region. Except for the license plate, the function *findIndices* also returned an average correlation factor for the complete license plate. This average factor is used to select the license plate with the highest possibility. At this point the system returns the license plate found in the original picture and this is therefor the end of the system. The system returns the license plate in ASCII code, what means that these are not the character values but this is easier if the Matlab scripts are ever going to be used in another software program. Also most languages support an easy conversion from ASCII code to character value.

4 Tests and Results

The main objective of this section is proving my system works. The explenation of how it works is done in previous sections. With testing the system the following things happend: The system is very good at identifying text in the image. When all boxes are matched to letters, the text is always translated correctly. The problem there is: there is more text on a car than only the licenseplate.
Pictures of cars where the license plate is

1. Clearly visible
2. Sharp enough
3. No text present on the car, or not in the same Y area.

The system always detects the right license plate.
Problems begin to occur when there is a lot of text present in the picture. Especially when this text is very clearly visible and sharp. The system releis completely on edge detection and not on machine learning to predict in which part of the image the license plate will be. This means that when the other text in the picture is of about the same size than the license plate and more clearly visible, the system is going to make mistakes.

5 Development process

In this section the development process is discussed. Before getting into the details, first a short description on why this is important. Although *Matlab* is a scientific software tool, making projects with *Matlab* can be compared to a software development process. In software development processes, a very important part is planning and way of developing. The way of developing has a direct impact on the chances to succeed, or chances to meet the objectives. Personally I have some experience with (small) software projects so I started this project with my experience in mind.
The first and very import stage is the preparation and planning. This is a big stage in the project because the better this is done, the easier it is to make and develop the project. With better is not meant more into detail, with better realistic and thoroughly are the real goals. The proper way of planning can easily be found in section 2 Problem Breakdown. There a dividends in different parts with each own sub problems is clearly visible. This way of working is necessary to keep track of the complete project and state clear objective goals.
After reading a lot online and in a very interesting book: TODO link image manipulation book, I fastly came to the conclusion that the image manipulation was the best way to start working on this project. In the case of libraries and concepts, this was also the most new part for me. Before working on this project, my matrix manipulation mostly concentrated on signal processing or audio filtering. Without having the exact necessary knowledge, it was still possible to define an end goal for this part, this because the end goal is straight forward: An image with all parts detected.
The next part, referred to as the search, is not so much based on new knowledge but more implementation of my personal ideas how this would work. By defining the sub problems as stated, it gives the developer (me) clear small objectives, what is most of all important in parts were the projects builds on own knowledge.
The same planning tactic can also be used for writing the report. Write objectives for your report: list all necessary items, design a template, make the base structure. Then write the first version, puzzled into the designed structure.
To keep track of changes and as a safety procedure, I did put the project (code and report) on *GitHub*. Working with a Git server not only provides a back up but the luxury of going back in time when certain mistakes are made.

6 Critical Reflection

To wright a critical reflection it is important to stretch the difference between a critical reflection and a conclusion. The conclusion will focus on the stated objectives and the results. An evaluation is made and in this evaluation also is room for a critical session. However the conclusion mostly focuses on the beginning and ending.
In this section, the critical reflection, the whole process is discussed en evaluated. From making the objectives to the developing process to failures or non met objectives.

6.1 The objectives

The objectives of this project were straight forward: recognize the license plates based on an input picture. To this objective a few other cases were added. Look into the possibility of moving images and CUDA (parallel processing). When looking critical to these objectives it is safe to say that there didn't went a lot of thinking in it. The main objective is clear and also realistic. But the objective of looking into moving images isn't. This objective was stated without a lot of pre knowledge in image processing and especially working with film. If I had done my research before starting the development of this project. I would have made the conclusion that this is a very difficult extension. This extension is probably a complete project on his own. More about the what and why in the section "possible extensions".
The second secondary objective was a good choice to state. Unfortunately the CUDA development didn't work out but it was a good thing to state it as an extra objective. If the original scope of the project was a little different it might have been possible to start with parallel processing but this wasn't the case. In my scope I developed a program that is machine independent. This will mean it will work on any machine that can execute *Matlab/Octave* code. This makes it very difficult to start with parallel processing because this depends on the GPU and direct control of the GPU is different on every machine combination. [5] For an example on a *Linux* operating system with a NVidia CUDA enabled processor it is very easy to start developing parallel processing, keeping in mind that on Linux octave is used instead of Matlab.

6.2 Development process

To reflect the development process it is necessary to introduce a time scope. The start of the development lays in February after the introduction of the objective. The finish and most of the reporting came after Semana Santa. This wide spread time period with the combination of other project and causes results in a lot of wasted time. Although the project is not very complex, it also not straight forward. This results in a start-up and work-in time needed after every break in development. Fortunately I already had the pre knowledge that I described in the previous section, this helped me to make a strategic plan and minimize the loss of time. This problems and loss of time had in no way to do with the way this course is designed or scheduled, it is only a result of the combination of the different works and dividing of my own attention between different causes.
The research done for the different part went very smooth because I was in the possession of a very good book on the matter.[12] Also online are a lot of examples of image processing and container regions available. At the end the most difficult job was connecting the dots.

[5] Machine combination means operation system in combination with type of GPU.

7 Conclusion

To make a general conclusion it is again necessary to look back at the stated objectives. **Main objective:**

- Design a system that recognizes license plates.

Side Objectives:

- Moving Images
- CUDA: parallel processing.

The main objective is met. The system designed and described is capable of distracting a number plate from a picture. It is possible to identify different kind of number plates and the system is transparent so it easy to implement in a larger or other system. The results make it clear that the system is not perfect. There can be made mistakes. The most mistakes are made when there is a lot more text in the pictures and specifically wen the font size is similar to the license plate. For the main objective the following conclusion can be made:
The objective is met but is not 100% effective. Therefor this system is good for academic purposes but not ready to be used in the street. Actual traffic control camera's have a way more complex system en probably determine the license plate in a few ways to be absolutely certain. Also the fact that the specifics of the license plate have to be in the script rules this out as a real world system. In a real world system all license plates have to be recognized not knowing which country region or design they have. To achieve this, it is possible that a complete different approach is necessary.
The side objectives are both not met. Processing moving images proves to be too complex to do in the span of this course. Maybe this can be done as a secondary project using the findings I've become. It is also possible that the clue to find the solution for the trade offs of the main objective can be found in the use of moving images. The fact why CUDA programming was not an option is already discussed in this report. An interesting study would be implementing my solution with parallel processing and observe what the time difference would be. In this way it should become clear in which way a custom designed system with parallel processor is necessary to make a system conforming the needs. Of course the conclusion of that research can be different when using images or moving images.

8 Possible Extensions

In this section I will discuss some products, developments, etc.. that can be made using my system. Later in this section follows a little more research and a short explanation about Moving Images and CUDA, this because these terms are well talked about during this report and maybe deserve a more in detail explanation.

8.1 Products

With making a general system like this, there are a lot of side products possible. There are some interesting ideas.

8.1.1 The extra Objectives

The extra objectives discussed in the previous sections are not met. This means that the first logical extension to the project is meeting these objectives. Meeting both extra objectives can mean a total change in strategy and project structure.

8.1.2 EU - proof

With making it EU proof is meant designing a system that can possibly detect all the different types of license plates in the European union. In this way the system will not only return the license plate but also the country of residence and maybe even the region. This extension is completely software side and can be in combination with the extension of the extra objectives.

8.1.3 Custom made products

While the previous extensions are mostly based on the software side, this is an integration of the software into custom hardware. Designing custom products int he way of designing a stand alone system. This can go from simple designs like a a camera attached to a raspberry pi for ”home” use, to a fully developed system. With fully developed meaning, multiple cameras connected to a heavy computer system or GPU, maybe even combined with a server.
With the ”home” system, or lightweight system, a lot of options are available. Opening the gate from your loan when your car is detected, opening a barrel of private corporate parking. Using the system to drop down road poles when certain vehicles arrive, there are a lot of options with the lightweight system. There is a security concern though, you are not controlling who enters but only which license plate so in theory everybody can copy the plate to enter so this always needs to be kept in mind while developing such a system.
Fully developed systems are bigger projects end are far more used in traffic control or security measurements. This means that the complexity of the system increases exponentially. Also because the professional usage requires a lot more fail proof system. Because of these reasons, a fully developed system lays maybe way out of reach of a student project.

8.2 Extra Research

The extra research is most of all an explanation for a two terms constantly used in this report.

8.2.1 Moving Images

No matter in which way this system or concept is described, one thing is always going to be true. It is a digital system. This means that moving images are in theory side by side images, produced in a certain rate. A very common rate is 25frames/second. This means that 25 pictures are produced every second. When developing a system it is going to have an input of 25 pictures a second. Analysing all 25 with the same intensity is not a good idea. This means that the system has to make a rational choice which frames can have a license plate in them and which frames to drop from the start.

8.2.2 CUDA

CUDA is a programming language, based on C, developed by Nvidia. Together with the right plugins, it allows a system to directly make use of the GPU. The GPU has to be CUDA enabled and of course from Nvidia. The direct advantage of GPU programming is that a GPU consists of more cores. The difference with a CPU is that a lot the GPU cores have to execute the same instruction. This is not good for general usage but it is good to do repetitive tasks. A common repetitive task is the manipulation of pixels or the processing of signals. Therefor it would be a very good extension to this project.
Toolboxes in *Matlab* make use of CUDA to enable parallel programming. This does mean that parallel code only works on computer systems that have a CUDA enabled GPU.

References

[1] DXC Technologies, "Intelligent transport systems." `https://www.dxc.technology/nl/offerings/136123-dxc_intelligent_transportation_systems`, January 2019. Last checked on apr 25, 2019.

[2] Wikipedia, "Rgb color model." `https://en.wikipedia.org/wiki/RGB_color_model`, April. Last checked on apr 26, 2019.

[3] Mathworks, "Imread." `https://nl.mathworks.com/help/matlab/ref/imread.html`, April. Last checked on apr 26, 2019.

[4] Nvidia, "About cuda." `https://developer.nvidia.com/about-cuda`, April. Last checked on apr 28, 2019.

[5] Mathworks, "Matlab gpu computing support for nvidia cuda-enabled gpus." `https://nl.mathworks.com/solutions/gpu-computing.html`, April. Last checked on apr 28, 2019.

[6] Mathworks, "Noise removal." `https://nl.mathworks.com/help/images/noise-removal.html`, April. Last checked on apr 24, 2019.

[7] Mathworks, "Edge detection methods for finding object boundaries in images." `https://nl.mathworks.com/discovery/edge-detection.html`, April. Last checked on apr 24, 2019.

[8] S. C. R. M. Michael Stokes, Matthew Anderson, "A standard default color space for the internet - srgb," *w3.org*, vol. 1.10, 1996.

[9] Nvidia, "Supported gpus." `https://www.geforce.com/hardware/technology/cuda/supported-gpus`, April. Last checked on apr 28, 2019.

[10] J. G. Proakis and D. K. Manolakis, *Digital Signal Processing (4th Edition).* Prentice Hall, 4 ed., 2006.

[11] R. van den Boomgaard and R. van Balen, "Methods for fast morphological image transforms using bitmapped images," *CVGIP: Graphical Models and Image Processing*, vol. 54, pp. 252–258, 05 1992.

[12] R. C. Gonzalez and R. E. Woods, *Digital image processing.* Upper Saddle River, N.J.: Prentice Hall, 2008.

A Matlab Scripts

Listing 1: The matlabscript corresponding main matlab script.

```matlab
% Pelle Reyniers
% This script is used to extract the license plate out of a picture.
% This is done by following a lot of steps, general control of the
    whole
% process is done in this script.
clear all;

% Part 1 ———————————————————— Image Manipulation ————————————————————

% image name is loaded into the workspace.
imageName='pic4.jpg';
% information about the license plate
totalCharacters = 6;
letters= 3;
% image manipulation script is used.
imageManipulation;

% The funciton regionprops returns information about the containers
    with
% capturing interesting regions in the picture.
% Boundingbox: info about the container
% Image: image for all the containers
ImageContainers=regionprops(final,'BoundingBox','Image');
ICMatrix=cat(1,ImageContainers.BoundingBox); % Matrix conversion

% Part 2 —————— Starting the search for parts off the license plate
    ——————
Images={ImageContainers.Image};
[plates,corrs]=findIndices(ICMatrix, totalCharacters, letters,Images)
    ;

if ~isempty(plates)

    [M, I]=max(corrs);
    licensePlateInt=plates(:,I);
    licensePlateInt=licensePlateInt';

    fid = fopen('licensePlate.txt', 'wt');
    fprintf(fid,'%s\n',licensePlateInt);
    fclose(fid);

else
    fprintf('Operation failed, the characters could not be matched to
         make a correct license plate.\n');
end

% Part 3 —————— End of the script, evaluation of result manually
    ——————
```

Listing 2: The matlabscript corresponding to the custom made greyscale function.

```
% Pelle Reyniers
% Function used to greyscale an rescale and greyscale an image.
% Funciton returns resized colored and greyscaled image.

function [rgbImage,greyImage] = greyscale(inputFileName)
try
    % read input image
    [X,map] = imread(inputFileName);
    % transform input image to rgb if needed.
    if ~isempty(map)
        rgbImage = ind2rgb(X,map);
    else
        rgbImage = X;
    end
    rgbImage=imresize(rgbImage,[400 NaN]); % Resizing the image.
    [rows, columns, numberOfColorChannels] = size(rgbImage);
    if numberOfColorChannels  == 3 % this doesn't work with RGBa
        % split the different color channels
        redChannel = rgbImage(:, :, 1);
        greenChannel = rgbImage(:, :, 2);
        blueChannel = rgbImage(:, :, 3);
        % make grayscale image
        greyImage = .299*double(redChannel) + ...
                    .578*double(greenChannel) + ...
                    .114*double(blueChannel);
        % Backscale using unint8 for readebility.
        greyImage = uint8(greyImage);
    else
        % image isn't RGB -> return input image.
        greyImage = rgbImage;
    end
catch ME
    errorMessage = sprintf('Error in function %s() at line %d.\n\
       nError Message:\n%s', ...
    ME.stack(1).name, ME.stack(1).line, ME.message);
    fprintf(1, '%s\n', errorMessage);
    uiwait(warndlg(errorMessage));
end
end
```

Listing 3: The matlabscript corresponding to the image manipulation.

```
% Pelle Reynierss
% Part 1: imageManipulation
% This script is ustrelDisk1d to manipulated the input image.
% After this script the image will be in a form that can be read and
% interpreted by the other Matlab scripts later in the project.

% Image manipulation consists of the following steps:
% - Read the image into the system
% - Greyscale the image
% - Remove possible noise
% - Dilate and Errode the image
% - Detect regions of interest in the picture

% -- Image preparation
% Greyscale the image -> custom made greyscale function
[rgbImage,greyImage] = greyscale(imageName);
% Remove noise -> cpu removal (universal okay), consider gpu removal
    when
% using a dedicated system.
greyImage=medfilt2(greyImage,[3 3]);

% -- Edge detection
% The use of strelDisk1el is explained in the report. High level:
    used for edge
% detection.
strelDisk1=strel('disk',1);
% Image dilation and eroding, use is explained in the report. Both
    are
% combinend to detect all edges.
greyImageImdilate = imdilate(greyImage,strelDisk1);
greyImageImErode = imerode(greyImage,strelDisk1);
% Substraction of the two previous items.
initialEdgeDetection=imsubtract(greyImageImdilate,greyImageImErode);

% -- Clear edges
% Converting the class to double -> increase brightness -> convert to
% logical values -> become a perfect only edges image.
initialEdgeDetectionDoubleValues=mat2gray(initialEdgeDetection);
initialEdgeDetectionDoubleValues=conv2(
    initialEdgeDetectionDoubleValues,[1 1;1 1]);
initialEdgeDetectionDoubleValues=imadjust(
    initialEdgeDetectionDoubleValues,[0.5 0.7],[0 1],0.1);
logicalEdges=logical(initialEdgeDetectionDoubleValues);

% -- Filter edges and fill areas
% Clearing the image from non interesting edges and fill interesting
% regions.
er=imerode(logicalEdges,strel('line',50,0));
out1=imsubtract(logicalEdges,er);
% Filling
F=imfill(out1,'holes');
% Thinning the image to ensure character isolation.
H=bwmorph(F,'thin',1);
H=imerode(H,strel('line',3,90));
% strelDisk1lecting all the regions that are of pixel area more than
    100.
final=bwareaopen(H,100);
```

Listing 4: The matlabscript corresponding to the function used to find the correct containers.

```
% Pelle Reyniers
function [plates, corrs]=findIndices(input, totalCharacters, letters,
     Images)
```

```
% Function findIndices gets as an input a vector with container
    information.
% This information contains location and size of the containers
    containing
% the interesting regions in the image. This function will make an
    educated
% guess on which containers are probably part of the license plate an
    which
% containers probably aren't.
% The output values is an array with indices of the corresponding
% intersting containers.

[n,xout]=hist(input(:,4));
bar(xout,n);
ind=find(n>=totalCharacters);
for a=1:length(input)
    combo(a)=input(a,2) * input(a,4);
end
input2=cat(2,input,combo');
[n2,xout2]=hist(input2(:,5),20);
ind2=find(n2>=totalCharacters);

plates=[];
corrs=[];

if length(ind)>=1
    for x=1:length(ind)
        % find middle point, find width, make box, add result to
            output.
        MP=xout(ind(x));
        binsize=xout(2)-xout(1);
        container=[MP-(binsize/2) MP+(binsize/2)];
        temp=takeboxes(input,container,2);
        [p,c] = eliminateOptions(temp, totalCharacters, letters,
            Images);
        plates = cat(2,plates,p');
        corrs = cat(2,corrs,c');
    end

elseif length(ind2)>=1
    for x=1:length(ind)
        % find middle point, find width, make box, add result to
            output.
        MP=xout2(ind2(x));
        binsize=xout2(2)-xout2(1);
        container=[MP-(binsize/2) MP+(binsize/2)];
        temp=takeboxes(input,container,2);
        [p,c]=eliminateOptions(temp, totalCharacters, letters, Images
            );
        plates = cat(2,plates,p');
        corrs = cat(2,corrs,c');
    end
end
```

```
end
```

Listing 5: The matlabscript corresponding to the function used to eliminate unlikely options.

```
% Pelle Reyniers
function [licenseOut, corrOut]=eliminateOptions(candidate,
    totalCharacters, letters, Images)
% This function is designed to guess every character in a license
    plate and
% reduce the license plate to the size previously stated.
% return the possible license plate and average corr.
tempPlateCh= [];
tempPlateCo= [];
for v=1:length(candidate)
    N=Images{1,candidate(v)};
    [letter corr]=readLetter(N);
    while letter==79 || letter==30
        if v<=letters
            letter=79;
        else
            letter=30;
        end
        break;
    end
    tempPlateCh = [tempPlateCh letter];
    tempPlateCo = [tempPlateCo corr];
end
% eliminate characters with lowest corr until restrictions are met
while length(tempPlateCh)>totalCharacters
    [M,I] = min(tempPlateCo);
    tempPlateCh(I) = [];
    tempPlateCo(I) = [];
end
licenseOut = tempPlateCh;
corrOut = sum(tempPlateCo)/length(tempPlateCo);
end
```

Listing 6: The matlabscript corresponding to the function used to match a container to a letter.

```
function [letter, maxcor]=readLetter(snap)
%READLETTER reads the character fromthe character's binary image.
%   LETTER=READLETTER(SNAP) outputs the character in class 'char'
    from the
%   input binary image SNAP.

load NewTemplates % Loads the templates of characters in the memory.
snap=imresize(snap,[42 24]); % Resize the input image so it can be
    compared with the template's images.
comp=[ ];
for n=1:length(NewTemplates)
    sem=corr2(NewTemplates{1,n},snap); % Correlation the input image
        with every image in the template for best matching.
    comp=[comp sem]; % Record the value of correlation for each
        template's character.
end
vd=find(comp==max(comp)); % Find the index which correspond to the
    highest matched character.
maxcor=max(comp);
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-
% Accodrding to the index assign to 'letter'.
% Alphabets listings.
if vd==1 || vd==2
    letter=65;
elseif vd==3 || vd==4
    letter=66;
elseif vd==5
    letter=67;
elseif vd==6 || vd==7
    letter=68;
elseif vd==8
    letter=69;
elseif vd==9
    letter=70;
elseif vd==10
    letter=71;
elseif vd==11
    letter=72;
elseif vd==12
    letter=73;
elseif vd==13
    letter=74;
elseif vd==14
    letter=75;
elseif vd==15
    letter=76;
elseif vd==16
    letter=77;
elseif vd==17
    letter=78;
elseif vd==18 || vd==19
    letter=79;
elseif vd==20 || vd==21
    letter=80;
elseif vd==22 || vd==23
    letter=81;
elseif vd==24 || vd==25
    letter=82;
elseif vd==26
    letter=83;
elseif vd==27
    letter=84;
elseif vd==28
    letter=85;
elseif vd==29
    letter=86;
```

```
elseif vd==30
    letter=87;
elseif vd==31
    letter=88;
elseif vd==32
    letter=89;
elseif vd==33
    letter=90;
    %*-*-*-*-*
% Numerals listings.
elseif vd==34
    letter=31;
elseif vd==35
    letter=32;
elseif vd==36
    letter=33;
elseif vd==37 || vd==38
    letter=34;
elseif vd==39
    letter=35;
elseif vd==40 || vd==41 || vd==42
    letter=36;
elseif vd==43
    letter=37;
elseif vd==44 || vd==45
    letter=38;
elseif vd==46 || vd==47 || vd==48
    letter=39;
else
    letter=30;
end
end
```