License Plate Recognizer
Numerical Calculations for Engineering

International Semester

Pelle Reyniers

April 24, 2019

Due Date: 17 May 2019

Class : International Semester

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 mutiple 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 evalution of this course consists of multiple smaller assignments and one large final project. This document describes that final project.

Contents

Introduction **I**

1 Problem Introduction **1**
1.1 Description . . . 1
1.2 Matlab functionality . . . 1
1.3 Goals and Objectives . . . 1
1.4 Possible Extensions . . . 1
1.4.1 Moving Images . . . 1
1.4.2 CUDA . . . 2

2 Problem Breakdown **2**
2.1 Image manipulation . . . 2
2.2 The Search . . . 3
2.3 Evaluation . . . 3

3 Component Overview **3**
3.1 Project Overview . . . 3
3.2 Image Manipulation Components . . . 3
3.2.1 Image Preparation . . . 3
3.2.2 Edge Detection . . . 4
3.2.3 Filling . . . 7

4 Tests and Results **7**

5 Development proces **7**

6 Critical Reflection **7**

7 Conclusion **7**

8 Possible Extensions **7**

9 Bibliography **7**

A Matlab Scripts **7**

Listings

1 The matlabscript corresponding to the custom made greyscale function 7

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. 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 licenseplate 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. 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, wheter 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 licenseplate when (in string format) when a picture is given as input.
- Creating a script capable of recognizing the land code.
- Provided enough test data and results to confirm all stated conclusions.

1.4 Possible Extensions

Further in this report is a larger section devotad to possible extensions, this is only a fortaste.

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. Because image processing contains a lot of parallel calculations, it can be interesting to see what the gain is when executing on a GPU.

2 Problem Breakdown

In this section the general problem is brought down to simpilar and smaller sub problems. The goals of this section is not making more objectives but making a clear and good dividence in the project. Important in breaking down a problem is th 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 defintion 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 whitin 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 defind in the previous section.

- Image manipulation: Manipulate the input image to a form where it is easy to start the proces of finding the license plate.
- The search: The search off the interire (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 considerd a sub problem.

a. **Uniform:** Transform the image to a uniform dimension. In this way all images from this point on have accactly the same characteristics.

b. **Greyscale:** Transform the image to a greyscale version off itself.

c. **Noise:** Removal of possible noise in the picture.

d. **Edges:** Detecting off all the possible edges in an image.

2.2 The Search

2.3 Evaluation

3 Component Overview

3.1 Project Overview

TODO: WRITE SUBSECTION ABOUT OVERVIEW OF THE TOTAL PROJECT.

3.2 Image Manipulation Components

Here follows an implementation of all the subproblems 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 happend: edgde detection to define possible areas where the license plate can be. To do this, the subproblems defined in the previous section, subsection image manipulation are implemntated. This implementation is shown in the current sub section.

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 seperate script present that reads and greyscales the image. Therefore can subproblems 1 on the facing page and 2 on the preceding page be combined when talking about the implemntation. The matlab code corresponding to these subproblems can be found in listing 1 on page 7. These subproblems 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 explenation when the program crashes for one reason or another. While the first lines take care of the importing of the picture and ashoring that it is an rgb version. Line 15 in the code solves subproblem 1 on the preceding page. 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 seperating the r, g and b channels, multipling each channel by a certain factor and recombining the new value. New value, as in singluar 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*. 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 usefull to remove the noise from a picture. The noise removal is done with the command *medfilt2()*. This function also accepts an gpuarray input, this is a better and faster way because

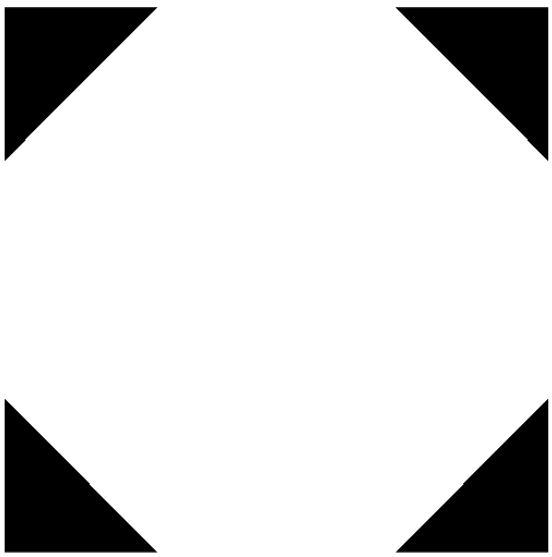

Figure 1: Example of a strel neighborhoud.

the matrix manipulations will be done by the GPU. The gpu has a significant more cores to calculate results. The gpu functionality does relie on the CUDA compatability 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 standerd version of my project. With removing the noise, subproblem 3 on page 2 is solved.

3.2.2 Edge Detection

After preparing the image, it is now ready to start the edge detection. Before discussion the implementation, a short explenation what edge detection is and how it can be achieved. An edge in an image is an area where the rgb pixel values drasticly change. For this reason it was important to remove the color without losing any value of the image.
Imagin not using a greyscaled image but a full colored rgb image. 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 technics is both dilating and erroding the image. The difference between the two resulting images will result in a very good edge detection.
To succesfull dilate an image, it is necesairy to have a neighborhoud search area, this can be created using a *strel* function. Imagine to have a neighborhoud that looks like a cicrle with radius of a few pixels. This is shown in figure 1. This *strel* is moving over an image, when it covers an aerea with the same rgb value, the system nows 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 [1] than the center

[1] Lighter in greyscale value means a higher value

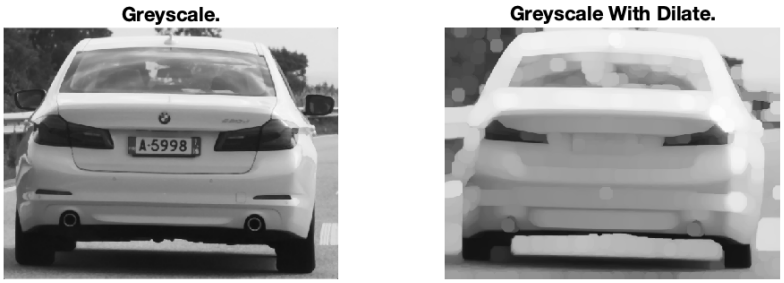

Figure 2: Example of a dilated image.

pixel, the value of this center pixel becomes this lighter value. When this is done on an image, the lighter areas on the image will become bigger. This is shown in figure 2.

To succesfull erode an image, it is necesairy to have a neighborhoud search area, this can be created using a *strel* function. Imagine to have a neighborhoud that looks like a cicrle with radius of a few pixels. This is shown in figure 1 on the facing page. This *strel* is moving over an image, when it covers an aerea with the same rgb value, the system nows 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 darker value [2] than the center pixel, the value of this center pixel becomes this darker value. When this is done on an image, the lighter areas on the image will become smaller. This is shown in figure 3 on the next page.

In the previous paragraphs, dilating and eroding is exagerated. When the radius of the *strel* is reduced to 1, the edges can be exact deteremt when the the results are subtracted. The result of this is shown in figure 4 on page 7. With this result we can conclude that subproblem 4 on page 2 is solved.

[2] Darker in greyscale value means a lower value

Figure 3: Example of a eroded image.

Figure 4: Example of an all edge detection

3.2.3 Filling

4 Tests and Results

5 Development proces

6 Critical Reflection

7 Conclusion

8 Possible Extensions

9 Bibliography

References

A Matlab Scripts

Listing 1: 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
```