# License Plate Recognition


**Group 9**

| | |
|---|---|
| **Tomas Heinsohn Huala** | 4326318 |
| **Remi van der Laan** | 4326156 |


## Introduction

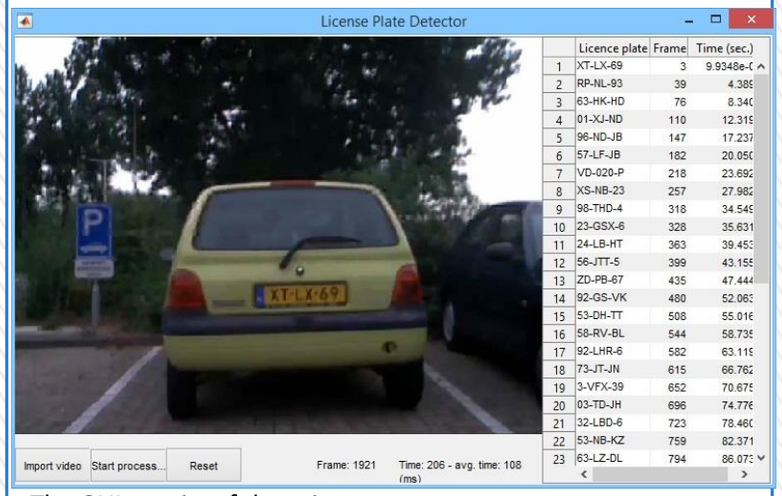
These days it's impossible to write down all license plates you see by hand to find them in a car database. For this purpose we created this application, because we have the technology. We created this application in MatLab and it outputs all license plates it can find in a table.

We used several image processing techniques, such as OCR (Optical Character Recognition), segmentation and made use of other knowledge, like sidecodes, to find the best results possible.

In this poster we will explain how our application was made and how it works and we will end it with an evaluation.

## GUI

Our application required a GUI, which we created with the GUIDE toolbox in MatLab:




The GUI consist of three important parts:

-The area where it displays the video being processed

-The three buttons: To start, import a video and to reset

-The table which contains the license plates as text

The start button is also a pause button when the video is playing and the reset button resets the video and the table.

The GUI also shows which frame is being processed and the average time it has taken to process a frame.

When a license plate is found, it is inserted along with the frame in which it was found and the current timestamp:

| | Licence plate | Frame | Time (sec.) |
|---|---|---|---|
| 1 | XT-LX-69 | 3 | 9.9348e-0 |

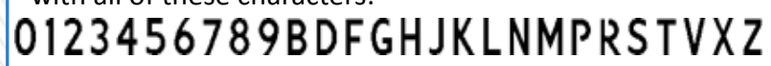
## Character Recognition

We have implemented our character recognition algorithm as follows:

We receive an image from the segmentation algorithm, which looks similar to this:


We calculate the bounding box of every object and check if it could be a character judging by its dimensions. Then we calculate the angle by comparing the width and height between the first and last character, so we can compensate the rotation if necessary.

For every possible character, we calculate the correlation with all of these characters:




The character corresponds with the image which has the highest correlation with the object.

Then we find the two largest gaps between the characters, where we insert the dashes.

We make use of sidecodes to correct any falsely recognized digits and characters, like 'B' and '8'. Sidecodes are the order and amount of characters which could be digits or alphabetic characters, like these ones:

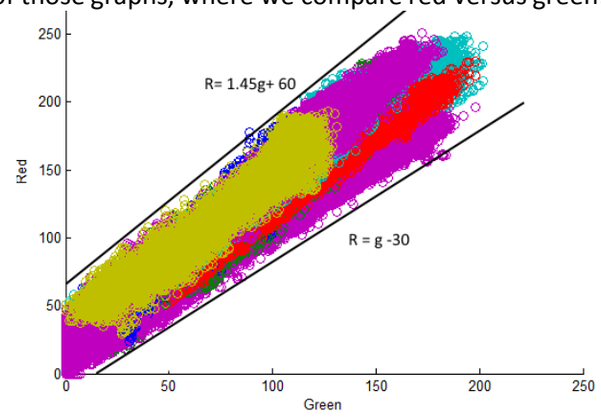
4: NL AA-11-AA
5: NL AA-AA-11
6: NL 11-AA-AA
7: NL 11-AAA-1
8: NL 1-AAA-11
9: NL AA-111-A

If any of these steps fail, we stop the recognition of that frame because we won't get an outcome which could be correct.

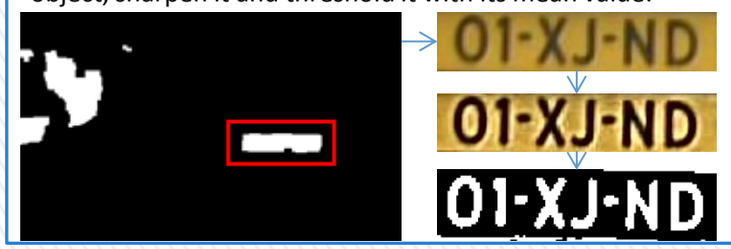
We then count which license plates are recognized and pick the one with the highest count if a new license plate is recognized.

## Segmentation

In our segmentation algorithm we locate the largest yellow object. Therefore we created graphs of the colors of many license plates in the RGB domain. With these graphs we were able to find restrictions for our segmentation function. This is one of those graphs, where we compare red versus green:




If we select these colors from an image, we get something like the image on the left. Then we select the largest rectangular object, sharpen it and threshold it with its mean value:




## Evaluation

After several weeks this project has come to an end. Obviously we made a very good license plate recognition application, but there are always points to improve:

-We made the choice to only look for yellow license plates, but the implementation for other colors would be relatively easy.

-Our program only looks for a single license plate because we did not have enough time to implement it for more.

-In some cases, when license plates are obscured or heavily overexposed, we weren't able to detect them.

-The speed at which it processes can probably be improved, but it's already a lot faster than our first character recognition implementation with the built-in OCR function.