License Plate Recognition

Group: 9

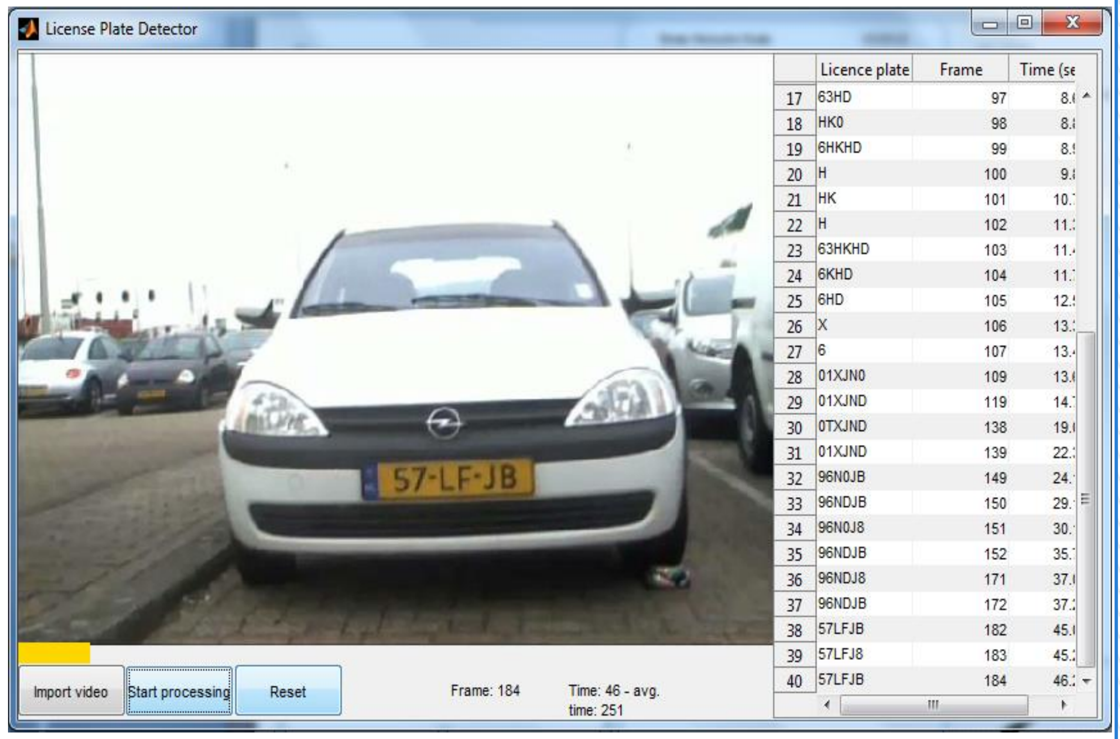
Tomas Heinsohn Huala	4326318
Remi van der Laan	4326156

This iteration

This iteration included a two week holiday, so we managed to do some extra things. We changed the GUI and we started experimenting with the text recognition systems. For this we used OCR (Optical Character Recognition) and we began to segment the license plate from the picture to read every character. As a result of that we are now able to detect the license plates and show the them in the list. There are some problems with this, for example that we don't really compare the license plates to check if they are correct so the list becomes flooded with incorrect license plates. Another problem is that it sometimes crashes because it can't find the license plate, because they are some yellow objects in the background. This error is made by our implementation of our OCR-function when it can't find any characters. But what we worry about the most is that the processing is taking a lot of time. The average time to process a frame is almost 300 ms, which is almost 10 times slower than it should be.

GUI

This is what our GUI looks likes at the moment :



	Licence plate	Frame	Time (se
17	63HD	97	8.
18	HK0	98	8.
19	6HKHD	99	8.
20	H	100	9.
21	HK	101	10.
22	H	102	11.
23	63HKHD	103	11.
24	6KHD	104	11.
25	6HD	105	12.
26	X	106	13.
27	6	107	13.
28	01XJN0	109	13.
29	01XJND	119	14.
30	0TXJND	138	19.
31	01XJND	139	22.
32	96N0JB	149	24.
33	96NDJB	150	29.
34	96N0J8	151	30.
35	96NDJB	152	35.
36	96NDJ8	171	37.
37	96NDJB	172	37.
38	57LFJB	182	45.
39	57LFJ8	183	45.
40	57LFJB	184	46.

We have rearranged the GUI elements and added some new features. The 'Start processing' button now changes to a pause button if the video is playing and the new 'Reset' button will play the video from the start again. The GUI also contains a progress bar which indicates how many frames out of the total have been processed. The list which contains the recognized license plates now also displays the frame and time when it was found. As you can see we've started working on recognizing the text from the license plates, which we will explain further in the 'OCR' section. Furthermore, the window of the GUI is now resizable and if a recognized license plate is the same as the previous entry in the list then it won't be added. This is still something we have to work on because often the recognized text is slightly different than the actual text and we will explain this further in the 'OCR' section.

This is an example of a correct result with its frame number and timestamp.

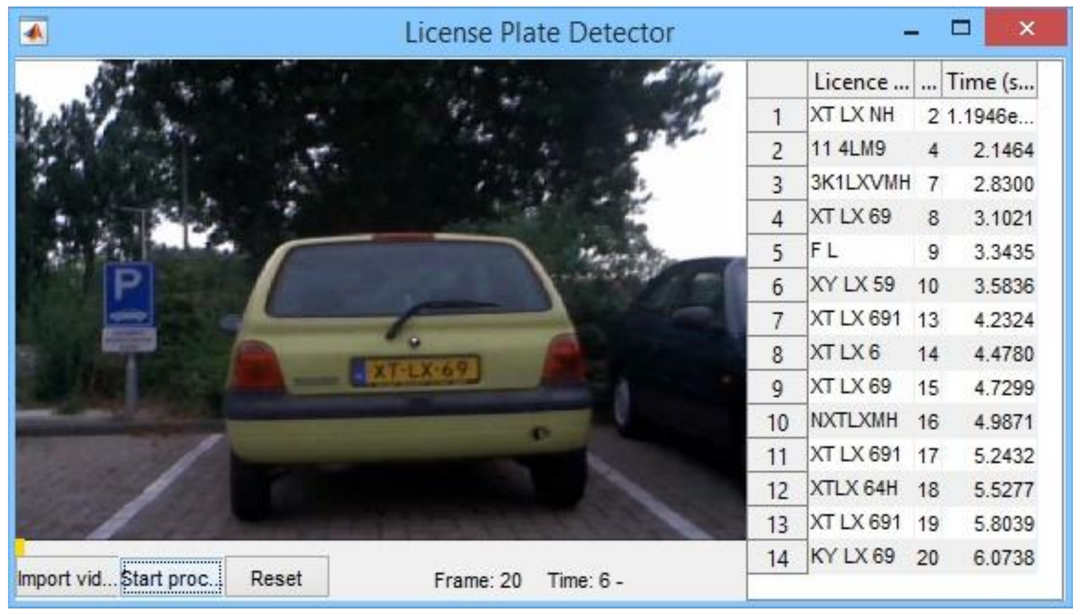
38	57LFJB	182	45.0281

This is an example of an incorrectly read license plate.

39	57LFJ8	183	45.2644

OCR

In this iteration we focused on implementing OCR (Optical Character Recognition) into our program to recognize the characters on the license plates. At first, we used the built-in OCR function of MatLab which we managed to get working, but it was very slow. Here are some results (if the result in the list is the same as the previous one, it's not added):

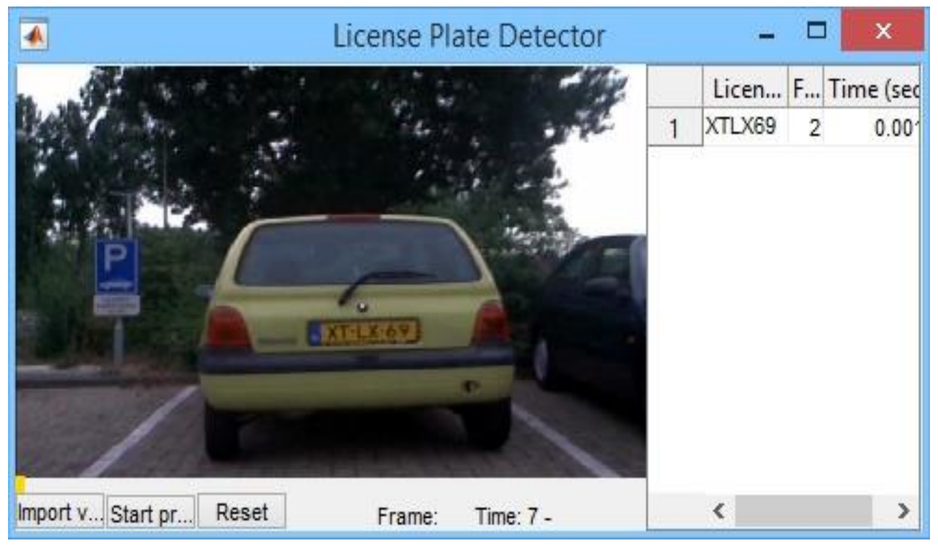


	Licence ...	...	Time (s...
1	XT LX NH	2	1.1946e...
2	11 4LM9	4	2.1464
3	3K1LXVMH	7	2.8300
4	XT LX 69	8	3.1021
5	F L	9	3.3435
6	XY LX 59	10	3.5836
7	XT LX 691	13	4.2324
8	XT LX 6	14	4.4780
9	XT LX 69	15	4.7299
10	NXTLXMH	16	4.9871
11	XT LX 691	17	5.2432
12	XTLX 64H	18	5.5277
13	XT LX 691	19	5.8039
14	KY LX 69	20	6.0738

In many cases the result was incorrect, so we started thinking about other OCR methods. We now use our own OCR function with a library of every possible character on Dutch license plates and find the highest correlation between every object found in the license plate and every character from our library. These are the characters we use:

-0123456789BDFGHJKLNMPRSTVXYZ

After we managed to get it working, the results were surprisingly good but it was even slower than the previous method:



	Licen...	F...	Time (sec
1	XTLX69	2	0.001

For this car, it found the same result for every frame. In some cases however, this method does not work very well because the segmentation is not perfect, which we will explain further in the next section.

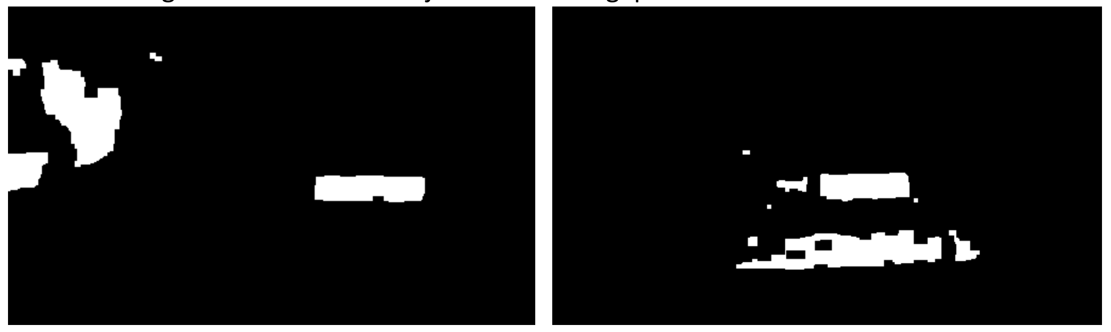
We had some problems with recognizing the dashes between characters, so we disabled that they can be recognized for the time being.

This process is quite costly, mostly because we have to resize every character to the size of every object for every frame to calculate the correlation. This could be precomputed but we would have to have a huge library of every character in every size. Another way to speed up the process would be to skip some frames if we are confident that the correct license plate has been found.

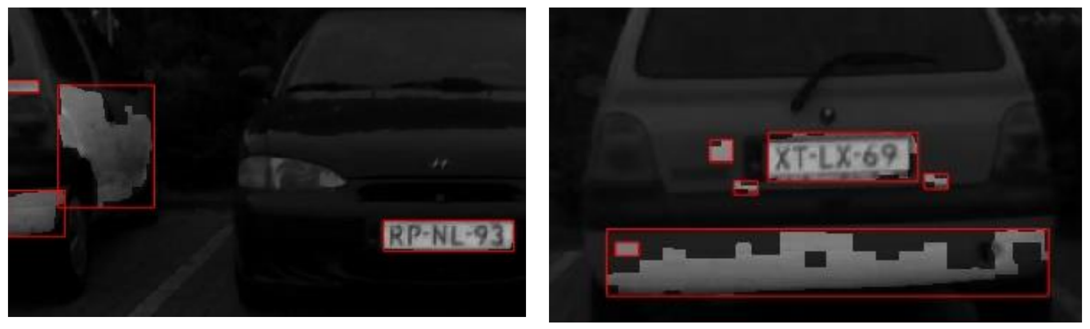
Segmentation

We improved our segmentation algorithm in a few different ways. Here are the steps in which we currently segment every frame:

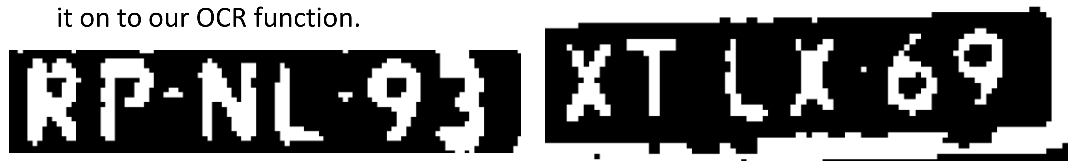
1. We segment the image by only selecting the yellow colors and then open and close the image to remove small objects and to fill gaps:



2. Then we label the image and calculate the area and bounding box for each object.



3. We then loop through every object , starting with the largest area, and check if its bounding box has an aspect ratio similar to a license plate.
4. Once we find an object which meets that condition, we stop the loop, crop the area of the original frame in the bounding box , threshold it with its mean value and pass it on to our OCR function.



Next iteration

We're satisfied with the progress we made this iteration, and hope we will finish product for the next iteration which will pass the first two categories.

Here are the tasks which we will focus on for the next iteration:

- Speed up the processing to play and process the video at the specified frame rate.
- Improve the segmentation algorithm to completely separate the characters from each other and the background.
- Find and correct the rotation of the video to improve the OCR algorithm.