Image Processing
Licence Plate Detector

[redacted]

1 Introduction *(0.5 pt)*

We start our program in the `CaptureFrame_Process` with following parameters. The `file_path`, the file path to the video, the `sample_frequency`, how many seconds we have between 2 frames we are going to analyse and the `save_path`, where we're storing the file in the final step. Our program starts by reading in the video, if we cannot find the video we print "No video found!" and terminate the program.

We extract some usefull information from the video (e.g. `total_frame` and `fps_video`) which we will use later on. We loop over all the frames and analyse every frame where the frame number modulus `frames_to_skip` is 0. `Frames_to_skip` is calculated using the `sample_frequency`. From there we first localise the number plate, than optimize the found number plate (e.g. rotation). Next we split the optimized number plate into different images such that every image contains a number/letter/noise. We than filter out the noise and recognize every symbol by comparing it to each symbol in a dataset.

In *figure 1* is the original image from the training video. This we used for training and for this document will be used to work on.

Figure 1: Original Image

2 License plate localization method *(4.0 pt)*

2.1 License plate localization data description *(0.5 pt)*

For the localization we did not need to look up a lot. We looked up the HSI values for yellow and in which range these values are. We also looked up how you can combine an image and a mask such that you only keep the part of the image that is also in the maks. The rest was mainly taken from what we learned from the lectures. We found our information for the localization on stack overflow and in the lecture slides. From there we combined the sources of information to create the `plate_detection` in our localization.

From the data we were given by the course we chose to use "trainingsvideo.avi" as our test video, but we skip the first 36 frames as these frames are already in our training data. Those 36 frames are excatly the "Video10_2.avi" in the the TrainingSet provided by the course. After applying the function we made we printed out the before and after image and the data. This way we could manually label if they were a pass or fail. We could check what the problem was directly and note it down as well.

For the optimization we used "Video10_2.avi" which was 36 frames with the same number plate. As soon as we were satisfied with the result of that number plate we performed a test on 29 number plates from different categories which were not used for training so we skipped the first 36 frames of the "trainingsvideo.avi" video. We counted a test as a pass if and only if at least in 1 frame the full licence plate and potentially some noise was returned by the method.

2.2 License plate localization system *(1.5 pt)*

The localization method takes in a frame and starts working on that until we are only left with the licence plate and maybe a bit of noise. The area of the noise « area of the licence plate. We start by creating a mask for yellow values, because our first focus was Dutch number plates who are always yellow for regular road cars. This mask you can see in *figure 2*. Before clearing out the holes in the mask we first find all corners and save their coordinates for later. Than we dilate 10 times over the mask to remove the holes in our mask where the letters should be and to make sure we get the full licenceplate. We still have the coordinates of the corners and we use those later to cut out the perfect licenceplate, but before we can use those we first need to rotate the plate. Here we return the plate with noise (because we gonna use that noise for rotation) and the 4 corner values to later cut out the licenceplate after rotating. The rotation is done by first finding a line that minimizes the distance from every point in the mask to the line. We calculated this using the least-squares-method. After the line is found it is represented as mx+b=y. If we take the atan of m we can find the angle at which the plate stands.

Figure 2: The mask before and after dilition

2.3 Evaluation metric for license plate localization *(1.5 pt)*

For the evaluation we used the test video that is described above. Checking if a plate was found was quite easy because we only kept the licence plate part. By simply displaying the found licence plate we could just say pass or fail by looking at our definition of a passing licence plate. This gives us also extra insights, if the plate was a fail we could note it down. We can use the gained information to improve our solution. We counted a test as a pass if and only if at least in 1 frame at least the full licence plate and potentially some noise was returned by the method. We know that "some noise" is a bit vague, but we counted it as too much noise, if it is dominant over the found plate. A small border of noise arround it is allowed, which was even very handy to find the rotation later.

We end up with a couple of numbers. The number of plates, the number of correct found and the number of wrongly found plates. To calcualte the pecision we do $precision = \frac{\#correctPlates}{totalplates}$. It is a simple formula, but sufficient enough for this problem.

2.4 Analysis of the license plate localization results *(0.5 pt)*

We get the following results:
We find 28/29 licence plates in Cat I
We find 10/10 licence plates in Cat II
We find 10/10 so 5/5 licence plates in Cat III
We find 0/10 licence plate in Cat IV

Figure 3: An example of a correct found licence plate

Figure 4: An example of a incorrect found licence plate

3 Character recognition method *(4.0 pt)*

3.1 Recognition data description *(0.5 pt)*

For the training of the segment-and-recognize part we used the first number plate of the video provided by the course. The model was validated using a small dataset we made by searching images of number plates on google. This data was used for both the segmenting and recognizing of the number plates/symbols. The actual recognition is done by comparing each symbol to each character in the dataset we got from the course. However we found out this was not sufficient enough so we made our own letters/numbers for most of given letter/numbers. This was made by our method of cutting out letters/numbers of our licence plate and simply saving that result in a bmp file. We used both the course and our dataset. We were happy with our method when we got an output in ascii that exactly matched the symbols in our number plate. We checked this manually by printing the output to the console and showing the input using imshow(). We did not count any partial correct ouput as the eventual goal is to recognize complete number plates.

3.2 Recognition system *(2.0 pt)*

3.2.1 Character localization system *(1.0 pt)*

The Localization of our character recognition process has 5 concrete steps. We first start with rotating the image over the angle given to us by the Localization method. After this step we have an image of a number plate with next-to-no noise surrounding the number plate. The numberplate itself is straight. We first convert the image to a greyscale image. This will help us out later when we convert it to a binary image. The conversion to a binary image is done by first calculating the ISODATA threshold. We then set everything that's below this threshold to 255 and everthing above to 0. This way the dark parts of the image, which are the symbols in the number plate, will be white and the background will be black. Afterwards we dialate and erode the image to both get rid of noise and fill in the gaps in the symbols. The final actual localization step of our characters is done by iterating over the collums of our image. We determine a certain threshold. For each collumn we check if there are less white value than this threshold, if this is the case we can assume this collumn is the end of a character and we split our image on that column.

Figure 5: input vs output

3.2.2 Character recognition system *(1.0 pt)*

Our input is an image of variable size which can contain either a symbol, or a line. We first do some preprocessing on the imput image. This contains normilizing the size of the image and cropping the image so we dont have any completely black rows or collumns. Next we start a for-loop through all the images in our character dataset. For each of these images some preprocessing is needed: they are not complete binary images so we convert them to binary. We also resize them to the same size as our input image. Finally we can start comparing them, we do this by taking the distance between the input image and each of the dataset images. We then take the image with the lowest distance to be our symbol. If this distance isn't low enough we assume the input image to be either noise or a line.

3.3 Evaluation metric for recognition *(1.0 pt)*

We made the method by looking how it worked on the first number plate in the video. After we were happy with that result we switched to the dataset we got from google. We used the metric: $\frac{\#fullplatesRecognized}{\#allplates}$. Like out metric from the localization this is a simple but sufficient metric. We also use this metric because we only have a correct plate when we recognize each symbol. We checked the results manually, by comparing the outputs the method gave us and the number plates themselves. When we were finally happy with the result we moved on to optimization. From the dataset found on google we recognized the full 10 out of 10 number plates.

3.4 Analysis of the recognition results *(0.5 pt)*

Figure 6: input vs output

One case were our method really struggles if there are a lot of shadows. This can best be seen in the number plate shown above. We use the ISODATA threshold to separate the foreground: symbols from the background. If there are a lot of shadows the plate gets too dark to seperate the symbols from background as the color diffrence between the 2 is not there. This causes the binary image to have a lot of noise at the top of our image. Subsequently this causes the splitting of the symbols to not work properly as it cannot find an x-value where the symbols should be split. The examples given above in figure 5 and 6 are a perfect example of this problem. We could improve this method by somehow making the method invariant to illumination. This by for example using HSI values instead of RGB values. Due to time limitations we cannot complete this.

4 Analysis of system *(1.5 pt)*

4.1 Time analysis *(0.1 pt)*

After going over the system with timers we found that the system is the slowest in recognition part. This due to the comparison of 28 letters and 18 numbers. This takes some time. That is why we only analyze with a frame rate 1/2 in stead of the default frame rate of 1/12.

4.2 Successes and failures *(1.0 pt)*

Simple plates that do not have much in common with the car (so no white or black cars) preform the best. Due to the ISODATA we are very vulnerable for these kinds of cars. We are also vulnerable for shadows and see that this gives us problems in the recognition. This is due to the choose to do mostly work on the color image and not for example edge detected images.

4.3 Future improvements *(0.4 pt)*

An improvement would be to use edge detection instead of a color detection to Localize the plates. This way we would also be able to find international plates as well. Another aspect we need to improve is speed. A the moment of writing this we take about 1 - 1.5 seconds for each frame. The way we could do this is by removing a lot of double for loops in the system. The localizing of the symbols could also be improved by using blob detection instead of the current method.