**Computer Vision – Assignment 2 – License Plate Detection**

**Name:** Killian Ronan **Student Number:** 18328687

1. **Declaration:**

This is my own work. Any material taken from other sources has been fully referenced in the text of the work. I have not worked with anyone else on this assignment and have not shown or provided my work to any other person. I have also not made my work accessible to others, both physically and electronically. All sources used in the preparation of this work have been listed in the Bibliography. I have read the statement on plagiarism in the College Calendar and understand that plagiarism is an offence that may result in penalties up to and including expulsion from the University.



1. **Theory:**

I engaged in many different attempts when designing my system in order to produce the best results. My solution followed the same process for the initial and subsequent frames.

I noticed early while running the project, that there was some noise, particularly on the left side and top of the frames with bright objects which were causing problems with my results. For this reason I applied a Gaussian filter onto these areas of the frame border to begin. I wrote a function which applied this filter only to the top and left side of the frame.

The rest of the background had a small amount of noise which was noticeable also. To smooth this noise in the rest of the frame, next, I apply another Gaussian filter with a kernel size of 51x51 to get the desired effect onto the whole image. Then, to remove some of the smaller objects, I apply opening through erosion followed by dilation on the blurred image. This left me with a frame like this:

**Blurred + eroded + dilated image:**

A picture containing text, road, car, outdoor

Description automatically generated

Next step was to convert the frame to greyscale and perform edge detection so that I could detect the edges of the license plate. After some practicing around I decided to use Canny edge detection as it was the most effective method and is widely accepted as the most efficient (more so than Sobel).

**Canny edge detected image:**

Text

Description automatically generated

I noticed the lines appeared quite thin, so I then perform dilation on this edge detected frame which will add pixels to the boundaries of the edges. This is done to ensure the license plate boundary will be clear to see.

**Dilated image:**

Text

Description automatically generated

I noticed in some frames that the license plate detected was not a closed object. So finally, I perform closing on the dilated, edge detected frame to close any small gaps left. This ensured that the perimeter of the license plate edge would be closed.

**Closed image:**

Text

Description automatically generated

The contours represent the connected components in the frame. Contours are generally the boundaries of objects in the each frame. We want this as we want to find the license plate boundary. For this reason, my next goal was to find the contours in the image and then to decipher which of the contours could be the license plate based on a number of metrics which I will discuss. Using the closed image generated above I called the opencv method findContours using the sample code given within the TIPS project as a basis. The first metric which I used to determine the license plate was the aspect ratio. This is calculated by dividing the width by the length of a given rectangle. We were given the dimensions of the license plate which is 465mm wide and 100mm long, meaning the aspect ratio is 4.65. I left a threshold of 0.6 to allow for any minor distortions the real plate may have. This meant any contour with an aspect ratio which was within 4.05 and 5.25, would pass the first test. The contoured image can be seen in section 3).

The next metric I used was a measure of the area minimum area rectangle generated from the contour. I noticed a few large contours being detected which matched the aspect ratio but were clearly too big to be the license plate. To remove these contours, I set a maximum value of 2500 for the area of the rectangle. It took some debugging and trial and error to decide this value. I wrote a simple distance formula to compute the area of the minimum area rectangle using its coordinates. I used this minimum area rectangle to also check the angle of the contour. If the angle is within 20 degrees of upright (90 degrees), then it passed the check. I also added a metric for the contour area. This value would always be lower than the area of the minimum area rectangle so the threshold area had to be lowered here to 1500, with also a minimum value of 100. These values were optimal for me when it came to choosing the correct contour.

The next metric I measured was by approximating each contour to a polygon and ensuring it did not have more than 4 or less than 10 sides. The road markings were contours I was detecting which were causing problems and this helped to mitigate their inclusion. Once a contour managed to pass all checks, I first check if the contour is the first to pass all checks. If so then I save its index, aspect ratio and angle, to compare at a later time if there is another contour which passes all checks. If another contour within the same frame passes all checks, the aspect ratio is compared with the highest contour and if it is a better match, it becomes the highest contour. If no matches are found, the false negatives counter is incremented unless the license plate has already left the frame. In that case the true negative counter is incremented.

Finally the last metric which I incorporated into my solution was ensuring the difference in area between contour and the minimum area rectangle was above a certain maximum. I tested some figures and found the optimal one for my solution was 300. The contour will always have the smaller area. However, the areas of both will be quite similar if the license plate is detected properly. As the license plate is almost a perfect horizontal rectangle, this means that its minimum bounding and minimum area rectangles will be extremely similar. I had an issue with many contours passing all checks bar this one, making it the most important of all.

After all contours have been checked and a highest contour has been determined, the DICE coefficient is computed. I wrote a function to handle this. It uses the min of maxes method for finding the intersection rectangle. The DICE coefficient is then calculated by multiplying two times the intersection of the ground truth and the located object divided by the sum of the two areas. If the returned result is above 0.8, meaning it is a good match, then the true positive count is incremented and the detected plate is drawn onto the current frame. If the DICE coefficient is below 0.8, the false positive counter is incremented.

**Contoured image:**

Graphical user interface, text, application

Description automatically generated

I calculated the distance from the contour in the image, to the camera by using the following formula:

Distance (mm) = (Focal length (mm) x Width (mm)) / Size in Pixels

The distances travelled between frames was calculated by simply subtracting the distance from the camera by the previous distance.

To calculate the speed of the vehicle, I first divided the number of frames by the given FRAMES\_PER\_SECOND constant to get me the amount of seconds in the gap between frames. Next I used this value to calculate the millimetres travelled per second by dividing the distance travelled between the two frames by the time between the two frames. Next I converted this millimetres per second value to kilometres per hour by first converting it to millimetres per hour then kilometres per hour. These results were logged to the console.

Next up, to calculate the precision and recall of my system I used the following formulas:

Recall = True Positives / (True Positives +False Negatives)

Precision = True Positives / (True Positives +False Positives)

The values for TP, TN, FP and FP are displayed on the current frame, along with the Recall and Precision. The distance to the camera is also displayed. The true positive rectangle is displayed in blue. The false positive rectangle is displayed in black.

The final result I was left with for a true positive prediction was:

Cars on a road

Description automatically generated with low confidence

The final result I was left with for a false positive prediction was:

Cars on a road

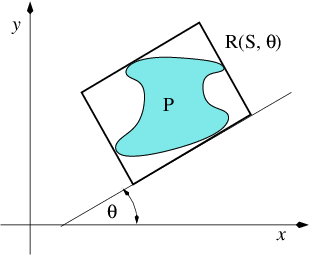
Description automatically generated with low confidence

1. **My Solution + Problems or Issues:**

There are a couple of design flaws with my system which most likely have more optimal solutions. My solution is quite general and doesn’t rely on any complicated image differencing. It involves smoothing the frame, converting to greyscale, along with some edge detecting and morphing of edges through the use of combinations of opening, closing, dilation and erosion to yield the best results. There may be better checks on contours for similarities to a license plate which I may have missed also. Eroding an dilating the initial image helped with the results of my model, but this may not have been the best course of action to begin. I will mention again that sharpening the license plate may have helped in the later stages as the detection method seemed to fail towards the last 40 frames.

Initially my model only had an precision of around 54%. I was having problems with my contour checking process as it seemed to be missing an important part. I found a solution which increased this precision to 81.9%. This was with ensuring the difference between the area of the contour and the area of the minimum area rectangle was not over 300. Many contours which were appearing matched the aspect ratio and matched both the bounding and minimum area rectangles area maximums and minimums. I could see that the detected contours for the license plate of my chosen solution were shaped like the picture below. They are rectangular shaped and I knew that the area of both the contour and minimum area rectangle of contours like these would be similar values. I chose a maximum difference between these two areas of 300 which optimised my results. This check helped me see the largest improvements. As we can see in the image below the minimum area rectangle is allowed to be an angle which allowed me to measure the angle of some contours. The minimum area rectangle being allowed to be rotated allows it to always be smaller or at the very least equal to the bounding rectangle. This also means if the contour is shaped like a rectangular license plate, then it will have an area very close to the minimum area rectangle.

**Minimum area rectangle:**  **Bounding rectangle:**

Shape, polygon

Description automatically generated

**Contour of located license plate:**

Graphical user interface, text, application

Description automatically generated

Towards the last 40 frames, my system fails to predict any true positives. This came as a result of the edge detected image not recognizing the license plate as a closed object. My solution involves closing the frame but this did not do enough to create a contour which would pass my checks. This could possibly be solved if I figured out a better method for detecting the license plate. A possible solution for this which I did not have time to implement, could be sharpening the image. This might help highlight the edges of the license plate further which would help the edge detection improve at the detriment of maybe having more noise to deal with.

Another solution for this which I attempted was with using a difference image. We are provided with a static background image, which I used first to see if the static background modelling would yield a better image for parsing the correct contour. I compared the current frame to the static background and obtained a difference image which I tried to smooth and perform morphologic operations on to optimise. However I could not get better results than with my chosen solution. There was too much noise in the resulting image using the static background model. Another idea similar to this method which I tried was using the current frame and previous frame to create a difference image. I thought this might be a more optimal solution but ended up with the same problem as with the static background model. I could not remove enough of the noise from the difference image to yield any better results than my chosen model.

Below you can find my attempt to use a difference image with the current and previous frame. I included the thresholded difference image and the dilated, thresholded difference image. I found it too hard to parse the license plate from this.

**Thresholded difference: Dilated thresholded difference:**

Shape

Description automatically generated with medium confidenceText

Description automatically generated

I couldn’t optimally select the contour of the license plate for this solution but maybe with the use of image sharpening to show the edges of the license plate better, before dilating the image, it would have yielded better results. I could have used closing also to close any gaps within the plate.

Below you can find my attempt to use a difference image with the current and static background frame. I included the thresholded difference image and the eroded, thresholded difference image and again. I eroded this time as I was to reduce the shapes and highlight the gap between the car and the license plate, but I found it too hard to parse the license plate from this.

**Thresholded difference: Eroded thresholded difference:**

A picture containing diagram

Description automatically generatedText

Description automatically generated

I couldn’t optimally select the contour of the license plate for this solution but maybe with the use of image erosion and some form of sharpening, it would have yielded better results.

1. **Performance Metrics:**

True positives were incremented when a correct match was found, true negative when no match was found when there shouldn’t be a match, false positives were found when an incorrect match was found and false negatives were found when no match is found when there should be one.

A picture containing text, device, gauge, meter

Description automatically generated

The final number of true positives was 100, true negatives was 45, false positives was 22 and false negatives was 53. This calculated from all the 221 frames in the video.

Using these values I calculated and printed the precision and recall of my system.

Graphical user interface

Description automatically generated

The Recall of my system after the 176 frames was 0.653 and the Precision was 0.82. The value for the recall isn’t terrible, but it’s not exactly great. It means that 65.3% of all of the relevant results were classified well. The system is precise, with only 82% of my results being relevant.

The results I obtained for the distances travelled in millimetres which I logged to the console can be seen below. Frames 143, 158 and 172 produced a false negative with my system so I was unable to obtain the distances or speeds for those 3 of the frames.

Text

Description automatically generated

I was able to estimate the distance travelled to within 655 millimetres of the ground truth provided, and the speed in km/h to within 4km/h. I estimated within 524 millimetres of the ground truth for the second frame gap (70 -> 86) and the same speed as before. The next frame (86 -> 101) improved immensely upon this, estimating the distance travelled to within 119 millimetres and the speed to within 0.7km/h difference. The next estimation (101 -> 115) was the most accurate of them all. It estimated the distance travelled to within 36 millimetres and the speed difference to within 0.3km/h. Finally, the last estimation (115 -> 129) which I was able to obtain got the distance travelled to within 296 millimetres of the ground truth and the speed within 2.5km/h.

1. **Results + Problems or Issues:**

With around 30 frames to go, the precision had a value close to its final value, however the recall had a much higher value of 0.82. There is a significant drop in the subsequent 30 frames due to the many false negatives being predicted. This is a problem with my model as no contours were recognized to have passed the checks at this point. If I could have found a better solution for the last few frames, my model would not have yielded such a low final value for recall.

Below frame 141 can be seen:

A picture containing text

Description automatically generated

I was pleased with the improvement in precision as initially, before I added one of the more important contour checks, the precision sat at 0.54. This was an increase of almost 30%.

My estimations for the speed and distance travelled were relatively accurate with the first estimation (54 -> 70) being the worst one. This can be expected as with the earlier frames, the license plate is further away meaning it is more than likely more difficult to detect accurately. The differences between the estimates gets smaller and smaller for the next two frame gaps as the vehicle edges closer to the camera. We can see clearly our estimation slowly become better and better as the car approaches. Early estimates had bigger differences like with (54 -> 70) and (70 -> 86). Then the next two estimates (86 -> 101) and (101 -> 115) yielded the most accurate results. The estimates then worsen slightly for the final frame (115 -> 129), indicating my model does not work as well as the car comes within a certain distance of the camera. This can be seen further as I was unable to obtain estimates for the proceeding frames. I was disappointed that I could not gather estimates for the final frames 143, 158 and 172, as they produced false negatives by my system.

I tried lots of different methods as explained in some of the sections above such as using image differencing, background models and using different morphological operations to improve the recognition of the license plate. The results I obtained using my system was the best I got. I would have liked to improve my model for the last few frames as it brought down my performance results, maybe I could have used image differencing for the last 40 frames with sharpening to help outline the license plate better. Overall I am happy with how my results turned out though.