Robotics and Computer Vision 2
University of Southern Denmark
Mandatory 2
Spring 2019

Steven Nygaard Hansen
Syddansk Universitet
steha14@student.sdu.dk

Chris Bruun Mikkelsen
Syddansk Universitet
chmik13@student.sdu.dk

I. Introduction

For this mandatory exercise it is required to analyze a video recorded from a drone. The video which was chosen depicts a curved road with an intersection at the top. The analysis consists of tracking the speed and position of the cars and also putting id's on the cars.

The implementation of the code must be done with ROS and OpenCV. The implementation of the working code was first done offline in order to make the process easier. When tracking the cars on the road, several methods must be used including background subtraction, optical flow and Kalman filtering.

A perspective correction must be implemented in order to be able to estimate the speed of cars travelling on the road.

II. Exercise 1

Create a ROS node structure able to handle the demands of the remainder of the exercises.

The ROS node structure can be seen in figure 1. Using the *rqt_graph*, you get this from the ROS nodes.

Figure 1. This is the node structure of the program running in ROS.

III. Exercise 2

Find a video to work on.

The video that was chosen as the subject of the traffic analysis was the first one on the list named: *https://www.youtube.com/watch?v=UPUQtdMUkoY*. This video was chosen because of the stability of the drone in flight. Because the video does not move around that much, the stability enhancement will be better.

Also the area where the trees hang over the road are quite small, so the cars are easier to track. There are some drawbacks to this image though. There are two bridges where the cars disappear for a moment and then come. Thus gives some issues with the tracking, but this has been taking in to consideration and dealt with. The analysis of the cars first start when the cars are out from under the train track bridge and for the cars going from the bottom of the image towards the top, the analysis stops when they go under the bridge.

Another drawback is that the cars are stopped in a queue and when they are standing still, they are very difficult to track. Then there are also the trees hanging over the road in the direction away from the bottom, this makes it more difficult for the contour function to correctly find a car, since it takes the area of the contour. Another drawback is the big trucks and busses that go by. These are big enough to have multiple contours in them and is therefore counted as several vehicles.

IV. Exercise 3

Locate cars that move in the video, by utilising some kind of background estimation model. Limit the car localisation to the road using an image mask. What advantage does the image mask provide ?

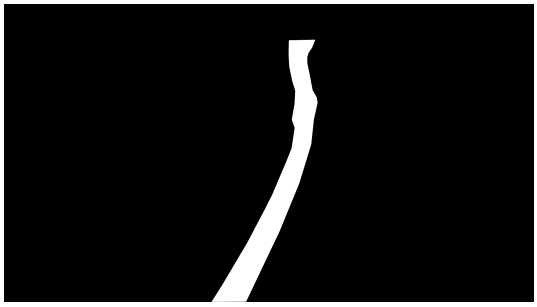

Figure 2. This is the image which was created in GIMP in order to mask the image so only the road is visible. This mask is then bitwise anded with the original image, which means that all black remains black, all white, is then the road.

Using a mask over the road, removes all noise outside the road, this could be birds, trees, bicyclists etc. Without a mask, it would not be possible to do a reasonable contour detection, this would also result in unpredictable behaviour like tracking a bird or a tree blowing in the wind.

With the mask, it was possible to use a decent tracking algorithm. The algorithm is still flawed since when we initialize the algorithm, two non-car objects are being tracked. The algorithm can be seen in figure 3. This could be removed by checking if this object was outside our Region of Interest(ROI).

Tracking of the cars is only taking place in the left lane of the road. This is due to issues when tracking the right lane and removing these issues would take too much time away from some larger concerns that needed taking care of. A box has been created underneath the train bridge, which only covers the left lane. So when a car exits from under the bridge and going towards the bottom of the image, then a tracking function will detect the car and start tracking it to the bottom of the image.

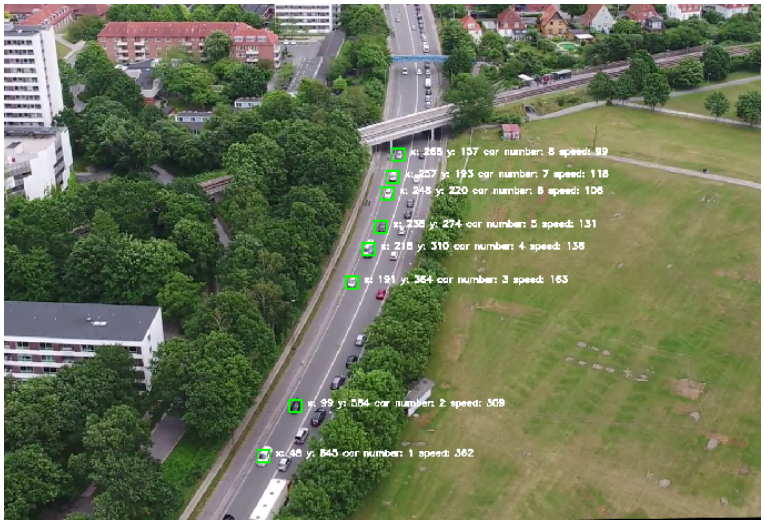

Figure 3. This image shows the result of the tracking algorithm, with kalman filter, it shows the car number, and the speed of the car in pixels. (The speed does not take the perspective in account), in this image, only the left lane is tracked, this also shows the cars until the bottom, if the cars exit the screen, the tracked circles just stop.

V. Exercise 4

Locate four reference points in one frame of the video and match these four points with their real coordinates (UTM) as they are given.

Four points was located in UTM coordinates, the problem here is to find four points which is both easy to locate on google maps/krak, and in the drone images. we have located four points receding in a somewhat rectangular pattern, but luckily the homography transform doesn't really care about if the points are in a rectangle or not. On figure 5 it is possible to see what perspective transform would do to the image. The image seems warped and corrected slightly, but a problem occurs when the cars move down, because the scale is still a bit off. This means that when the car is in the top of the image, it is larger than in the bottom.

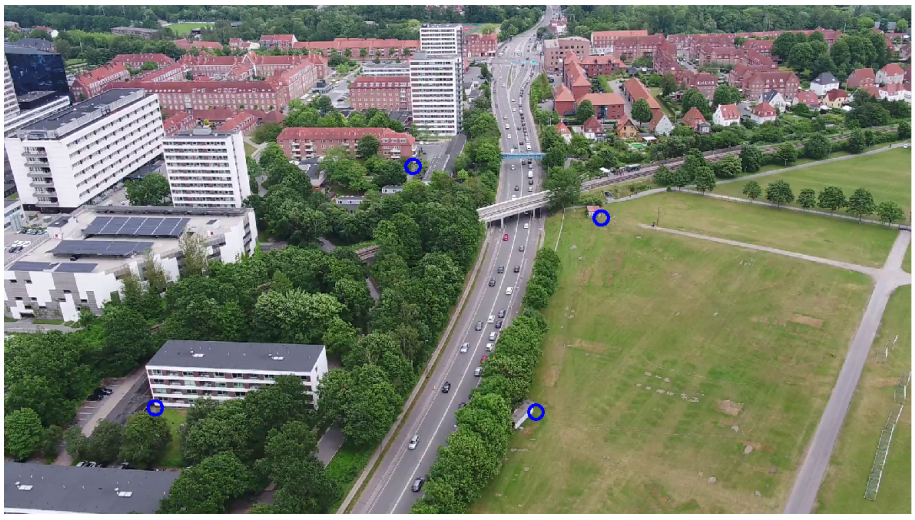

Figure 4. Four points found and mapped for UTM conversion and perspective transform. the UTM coordinates was found on Krak.dk

Figure 5. The warped image have made the ROI not totaly comparable, meaning that the cars in the bottom of the image, seems to be not as large as the ones in the top of the ROI, this is a unwanted feature because this makes it hard to do a speed estimate of each car.

VI. Exercise 5

Estimate the position and speed in x and y coordinates over time by using Kalman filters. Use the locations from exercise 3 projected down on the groundplane.

The Kalman filter showed to be more difficult to implement than first anticipated and therefore it has first been properly implemented late in the process. The kalman seems to track the cars in a decent fashion, when the program is running, a box i created around the car, and the x,y coordinates of that car is printed besides the car, with the ID and the speed of the car. The speed of the car is unfortunately not in km/h and is actually the

difference in position from the previous image. To correct this, one would do a perspective transform of the image / the coordinates of the car. send it to the kalman filter, find the displacement from the previous image, calculate the length of one pixel from the image versus the real world, this would give meters per frame. It would then be possible to find out how long time one frame is, and convert it to meters/second or km/h.

VII. Exercise 6

To ensure a Kalman filter is always tracking the same car a matching method is needed. It could be beneficial to look at optical flow or maximum likelihood, for matching objects from one frame to the next.

The only matcher that has been implemented is one which uses the euclidean distance to match if it is the same car in the next frame. Preferably extra matchers would also have been implemented, so that a more stable result could have been achieved. Right now the tracker loses the cars when they are almost out of the image and this might be because of the matcher lacking the ability to match each car to the next frame. One could implement a matcher which would look at the size, shape and color of each car. and then take the object which matches the car the best.

VIII. Exercise 7

Add tracking information to a visualisation output of the original video(image sequence).

The tracking information has been added to the original image, this can be seen in figure 3 where you can see the ID of the cars, the position and also the speed of the cars. Though the speed of the cars is not yet properly calculated, there is still something being in printed. The current speed implementation tell the difference in pixels from one image to the other. this is desired to be km/h instead, to do this, one would measure approximately how many pixels 100 meters is, and then do a approximation of the cars speeds in relation to the time difference from one image to the next.

IX. Conclusion

This project seemed to be a bit large, and we have probably not spend enough time on the project, every part of exercise is flawed and could use a update. Nevertheless this project was exciting and fun to program and gave a great intro to the Robot Operating System, with packages forwards and backwards between systems and nodes
A launch file was also made, which made running the program much easier.
We also got to look into some trackers for example lucas kanade and optical flow.

References